Linearized Acousto-Electric Impedance Tomography

Christina Berndt Hildebrandt

M.Sc.Thesis

Kongens Lyngby 2015

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Linearized Acousto-Electric Impedance Tomography

Author:
Christina Berndt Hildebrandt, s103234

Supervisor:
Associate Professor Kim Knudsen

Co-supervisor:
PhD Student Henrik Garde

Project period: January 26th 2015 - June 19th 2015
Workload: 30 ECTS
Degree: M.Sc.in Engineering
Programme: Matematical Modelling and Computation (Honors)

Summary

The purpose of the thesis is to investigate the linearisation of Acousto-Electric Impedance Tomography as well as explaining the behaviour and characteristics of the reconstructions by use of singular value decomposition.

The thesis consists of an introduction to the model of Acousto-electric Impedance Tomography followed by a linearisation of the model by use of Fréchet derivatives. In order to do analysis of the system for the reconstruction the linearisation is then decoupled. The system is implemented numerically in Python using FEniCS, [1], to create reconstructions. To improve the reconstructions Tikhonov regularization is introduced as well as truncated singular value decomposition to get rid of artefacts appearing in the reconstructions. The concept of noisy measurements is also investigated by adding Gaussian noise to the electrical power density distribution, where we again get improved reconstructions by use of regularization.

To examine the effectiveness, when access to the boundary is restricted, we investigate what happens with the reconstructions when part of the boundary is set to have a homogeneous Dirchlet boundary condition. Here it is clear that areas close to these boundaries gives unreliable reconstruction and we need regularization to get better solutions. However when doing regularization we lose information about the unreliable areas and when we have elements hidden behind each other these are lost before the artefacts are removed.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Engineering: Mathematical Modelling and Computing. It represents the completion of my honors master program at DTU and a work load of 30 ECTS points. The thesis was conducted in the Spring 2015 with Associate Professor Kim Knudsen as supervisor and PhD student Henrik Garde as co-supervisor.

The thesis examines the model for the hybrid tomography method known as Acousto-Electric Impedance Tomography. It treats the linearisation of the model and the implementation to do reconstructions. Furthermore it investigates the solution using singular value decomposition as well as improving reconstructions by means of regularization. Lastly the model of Limited-View is treated.

The prerequisites for reading this thesis is a basic understanding of inverse problems and partial differential equations as well as a more profound understanding of functional analysis including Sobolev spaces.

One thing to note is that several of the images of reconstructions looks better in the electronic copy than when printed. The reader is encouraged to check out the images on a computer if one wants to see the best images.

Christina Berndt Hildebrandt
Kongens Lyngby, June 19th 2015

Acknowledgements

First and foremost I would like to thank my supervisor and honors mentor, Associate Professor Kim Knudsen for providing guidance and help when needed and taking the time to meet with me on a regular basis as well as providing me with the subject of this thesis.

I would also like to thank PhD student Henrik Garde for help with Python and FEniCS, when my knowledge was lacking, and Postdoc Kaloyan Dimitrov Marinov for taking the time to talk to me about the subject of the thesis.

Lastly I would like to thank my fellow student Katrine Wittenhoff Kristensen and my boyfriend Kasper Lüthje Jørgensen for listening to me ramble on about subjects they knew nothing about while still providing helpful inputs and helping me keep my focus.

Contents

CHAPTER 1

Introduction

Acousto-Electric Impedance Tomography (AET) also known as a Ultrasound Modulated EIT (UMEIT) is a hybrid inverse problem. A hybrid inverse problem is a combination of multiple modalities, that complement each other to increase the contrast, resolution and stability of the reconstruction.

Medical ultrasound is a well-known imaging method. Here ultrasonic waves penetrate the medium. As they propagate through the medium some are scattered and some are reflected at the interior interfaces. The wave response is measured and provides information about the structure of the medium. This is an inverse scattering problem, which is generally ill-posed.

In Electrical Impedance Tomography (EIT) one applies voltage to the boundary of a domain and then measure the resulting current. The conductivity is then recovered from a boundary data mapping, which relate the applied voltage to the resulting current a so called Dirichlet-Neumann-mapping. One thing to note is that the inverse problem of EIT is severely ill-posed.

AET uses the acousto-electric effect that describes the conversion of acoustical energy into electrical energy. High intensity acoustic pressure waves create a local deformation

of the electronic structure which changes the electrical properties.

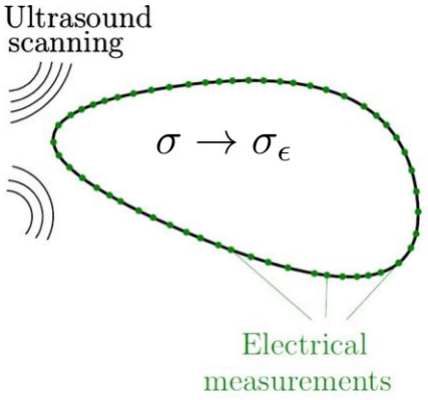


Figure 1.1: Ultrasound scanning causes change in the conductivity and this causes a change in boundary measurements, [9]

The idea is to conduct classical EIT measurements, while a known ultrasonic waves travels through the object. Due to the acousto-electric effect changes in the electrical properties will be recorded. This is then done for a family of ultrasonic waves, where we have different interior conductivity perturbations.

The boundary value problem for EIT resulting from Maxwell's equation under the assumption of low frequency is given as

$$\begin{cases} \nabla \cdot \sigma \nabla u &= 0 \quad \text{in } \Omega \\ u &= f \quad \text{on } \partial\Omega \end{cases}, \tag{1.1}$$

where $u \in H_0^1(\Omega)$ is the electrical potential, $\sigma \in L_+^\infty(\Omega)$ the conductivity and $f \in C^\infty(\partial\Omega)$ boundary potential i.e. the voltage applied to the boundary on a domain Ω.

Definition 1.1 $(L_+^\infty(\Omega))$
$L_+^\infty(\Omega)$ consists of the functions $\sigma \in L^\infty(\Omega)$, which are each bounded below by a some positive constant, K_σ i.e.

$$K_\sigma \leq \sigma(x) \text{ for a.e. } x \in \Omega,$$

so K_σ is the essential infimum of σ.
$\|\sigma\|_{L^\infty(\Omega)}$ i.e. the essential supremum is denoted by M_σ.

This is a very simplified model of the real experiment. Another model is the complete

electrode model, where it is taken in to account that we have a finite amount of electrodes and that we have a surface impedance between the medium and the electrodes.

1.1 Interior data

The ultrasonic signal is assumed to be a plane wave with a chosen amplitude and with constant wave speed. In accordance with [2], [8] one can determine a mathematical expression of the interior from boundary measurements.
The acousto-electric effect can be modelled by the relation (See for example [8], [9])

$$\sigma_\epsilon(x) = \sigma(x)\left(1 + \epsilon e^{i(k\cdot x)}\right), \tag{1.2}$$

where $\epsilon \ll 1$ is a product of a measure of the acousto-electric coupling between the wave and the electrical conductivity and the amplitude of the wave. $\sigma_{-\epsilon}$ can be found by shifting the phase of the wave (i.e. changing k).

u_ϵ is defined as the solution to

$$\begin{cases} \nabla \cdot \sigma_\epsilon \nabla u_\epsilon &= 0 \quad \text{in } \Omega \\ u_\epsilon &= f \quad \text{on } \partial\Omega \end{cases}. \tag{1.3}$$

Now using the integration by parts we get

$$\begin{aligned} &\int_\Omega (\sigma_\epsilon - \sigma_{-\epsilon}) \nabla u_\epsilon \cdot \nabla u_{-\epsilon} dx \\ &\qquad = \int_\Omega \sigma_\epsilon \nabla u_\epsilon \cdot \nabla u_{-\epsilon} dx - \int_\Omega \sigma_{-\epsilon} \nabla u_\epsilon \cdot \nabla u_{-\epsilon} dx \\ &\qquad = \int_{\partial\Omega} \sigma_\epsilon u_{-\epsilon} \partial_{\boldsymbol{\nu}} u_\epsilon ds - \int_\Omega u_{-\epsilon} \underbrace{\nabla \cdot \sigma_\epsilon \nabla u_\epsilon}_{=0} dx \\ &\qquad\quad - \int_{\partial\Omega} \sigma_{-\epsilon} u_\epsilon \partial_{\boldsymbol{\nu}} u_{-\epsilon} ds + \int_\Omega u_\epsilon \underbrace{\nabla \cdot \sigma_{-\epsilon} \nabla u_{-\epsilon}}_{=0} dx \\ &\qquad = \int_{\partial\Omega} \sigma_\epsilon u_{-\epsilon} \partial_{\boldsymbol{\nu}} u_\epsilon - \sigma_{-\epsilon} u_\epsilon \partial_{\boldsymbol{\nu}} u_{-\epsilon} ds, \end{aligned} \tag{1.4}$$

where $\boldsymbol{\nu}$ is the unit outer normal and $\partial_{\boldsymbol{\nu}} u = \boldsymbol{\nu} \cdot \nabla u$.

The right-hand side can be calculated by the measured EIT boundary data for different

values of ϵ and therefore we can make a polynomial expansion in ϵ

$$\int_{\partial\Omega} \sigma_\epsilon u_{-\epsilon} \partial_\nu u_\epsilon - \sigma_{-\epsilon} u_\epsilon \partial_\nu u_{-\epsilon} ds = \epsilon J_1(k) + \epsilon^2 J_2(k) + \mathcal{O}(\epsilon^3), \tag{1.5}$$

by changing the amplitude of the wave. Here $J_1(k)$ and $J_2(k)$ are the coefficients of the polynomial expansion.

We want to determine an expression for the first order coefficients $J_1(k)$.
We look at $\sigma_\epsilon \nabla u_\epsilon$ and use the definition of σ_ϵ from (1.2) and $u_\epsilon = u + d_\epsilon$, where d_ϵ is the difference between the two solutions

$$\begin{aligned}\sigma_\epsilon \nabla u_\epsilon &= \sigma \left(1 + \epsilon e^{i(k\cdot x)}\right) \nabla \left(u + d_\epsilon\right) \\ &= \sigma \nabla u + \sigma \nabla d_\epsilon + \sigma \epsilon e^{i(k\cdot x)} \nabla u + \sigma \epsilon e^{i(k\cdot x)} \nabla d_\epsilon,\end{aligned}$$

which gives us

$$\sigma \nabla u = \sigma \left(1 + \epsilon e^{i(k\cdot x)}\right) \nabla u_\epsilon + \mathcal{O}(\epsilon), \tag{1.6}$$

if $d_\epsilon = \mathcal{O}(\epsilon)$.
Thus we take a look at the PDE for d_ϵ,

$$\begin{aligned}\nabla \cdot \sigma \nabla d_\epsilon &= \nabla \cdot \sigma \nabla \left(u_\epsilon - u\right) \\ &= \nabla \cdot \sigma \nabla u_\epsilon \\ &= \nabla \cdot \left(\sigma - \sigma_\epsilon\right) \nabla u_\epsilon \\ &= -\nabla \cdot \sigma \epsilon e^{i(k\cdot x)} \nabla u_\epsilon.\end{aligned}$$

This gives me a boundary value problem

$$\begin{cases} \nabla \cdot \sigma \nabla d_\epsilon &= -\nabla \cdot \sigma \epsilon e^{i(k\cdot x)} \nabla u_\epsilon \quad \text{in } \Omega \\ d_\epsilon &= 0 \quad \text{on } \partial\Omega \end{cases}.$$

By Theorem A.2 one can show that

$$\|d_\epsilon\|_{H_0^1(\Omega)} \leq C \frac{\|\sigma \epsilon e^{i(k\cdot x)}\|_{L^\infty(\Omega)}}{K_\sigma} \|u_\epsilon\|_{H^1(\Omega)} \leq \epsilon \tilde{C} \|u_\epsilon\|_{H^1(\Omega)}, \tag{1.7}$$

where the constant, $\tilde{C}$, depends on the bounds of σ.
Thus to have $d_\epsilon = \mathcal{O}(\epsilon)$ we need to ensure that u_ϵ does not depend on ϵ reciprocally. To do this we start by analysing the boundary value problem for u (1.1) by use of the Theorem A.2 (See Appendix A for the analysis). In (A.4) we get

$$\|u\|_{H_0^1(\Omega)} \leq c(\Omega) \left(\frac{M_\sigma}{C K_\sigma} + 1\right) \|f\|_{H^{1/2}(\partial\Omega)},$$

for

$$K_\sigma < \sigma(x) < M_\sigma, \text{ for a.e. } x \in \Omega$$

and some positive constant C independent of σ.
For $\epsilon \ll 1$ we can do similar calculations for u_ϵ since the perturbation is assumed to be small

$$\|u_\epsilon\|_{H_0^1(\Omega)} \leq c(\Omega) \left(\frac{\tilde{M}_\sigma}{C\tilde{K}_\sigma} + 1 \right) \|f\|_{H^{1/2}(\partial\Omega)},$$

for $\tilde{M}_\sigma, \tilde{K}_\sigma > 0$, which are the bounds on σ_ϵ, which means that $d_\epsilon = \mathcal{O}(\epsilon)$.

From this we get that $u_\epsilon = u + \mathcal{O}(\epsilon)$. Now using (1.6) and $u_\epsilon = u + \mathcal{O}(\epsilon)$ implies that

$$\nabla u_\epsilon = \nabla u - \epsilon e^{i(k\cdot x)} \nabla u + \mathcal{O}(\epsilon). \tag{1.8}$$

Looking at the left-hand of (1.4) we can plug in the expression for σ_ϵ and $\sigma_{-\epsilon}$

$$\begin{aligned} &\int_\Omega (\sigma_\epsilon - \sigma_{-\epsilon}) \nabla u_\epsilon \cdot \nabla u_{-\epsilon} dx \\ &\qquad = \int_\Omega \left(\sigma \left(1 + \epsilon e^{i(k\cdot x)} \right) - \sigma \left(1 - \epsilon e^{i(k\cdot x)} \right) \right) \nabla u_\epsilon \cdot \nabla u_{-\epsilon} dx \\ &\qquad = \int_\Omega \left(2\sigma\epsilon e^{i(k\cdot x)} \right) \nabla u_\epsilon \cdot \nabla u_{-\epsilon} dx. \end{aligned} \tag{1.9}$$

Using the expression from (1.8) we can rewrite $\nabla u_\epsilon \cdot \nabla u_{-\epsilon}$

$$\begin{aligned} \nabla u_\epsilon \cdot \nabla u_{-\epsilon} &= \left(\nabla u + \epsilon e^{i(k\cdot x)} \nabla u + \mathcal{O}(\epsilon) \right) \left(\nabla u - \epsilon e^{i(k\cdot x)} \nabla u + \mathcal{O}(\epsilon) \right) \\ &= \nabla u \cdot \nabla u - \underbrace{\epsilon^2 e^{i(k\cdot x)} \nabla u \cdot \nabla u}_{=\mathcal{O}(\epsilon^2)} + \mathcal{O}(\epsilon) \\ &= \nabla u \cdot \nabla u + \mathcal{O}(\epsilon). \end{aligned} \tag{1.10}$$

Plugging (1.10) in to (1.9) we get that the first order term of (1.5) is given by

$$\int_\Omega 2\sigma\epsilon e^{i(k\cdot x)} \nabla u \cdot \nabla u dx = \epsilon J_1(k).$$

Thus $\frac{1}{2} J_1(k)$ is the Fourier transform of $\sigma|\nabla u|^2$ extended to zero outside Ω. In theory we can measure $J_1(k)$ for all values of k and then by the inverse Fourier transform this will correspond to interior knowledge of $\sigma|\nabla u|^2$.

The inverse problem of AET is then to reconstruct σ from the knowledge of the electrical power density distribution, $\sigma|\nabla u|^2$ for a chosen set of boundary conditions $\{f_j\}_{j=1}^J$.

1.2 Model

The hybrid inverse problem for AET can be formulated as

Let Ω be open, bounded subset of $\mathbb{R}^2$ with a sufficiently smooth boundary $\partial\Omega$ and let $\sigma \in L^\infty(\Omega)$ and bounded from below by a positive constant in Ω. Consider the set of PDEs

$$\begin{cases} \nabla \cdot \sigma \nabla u_j &= 0 \quad \text{in } \Omega \\ u_j &= f_j \quad \text{on } \partial\Omega \end{cases} \tag{1.11}$$

and the interior data of the type

$$H_j = \sigma |\nabla u_j|^2 \text{ in } \Omega. \tag{1.12}$$

For a chosen set of boundary conditions $\{f_j\}_{j=1}^J$ and knowledge of the corresponding interior data $\{H_j\}_{j=1}^J$ we want to reconstruct σ.

It is also possible to use focused and spherical waves instead of plane waves to recover the electrical power density distribution by a similar construction, [10].

By [4, Theorem 3.2] it is known that for the interior power density of the form $H_{ij} = \sigma \nabla u_i \cdot \nabla u_j$ there is global uniqueness and stability in dimension $n = 2$ for $J = 2$.
For $J = 2$ we will then get the measurements H_{11}, H_{22} and H_{12}, however in our set-up we will not directly have knowledge of H_{12}. By use of the polarization identity for real-valued entries on H_{12} we get that

$$H_{12} = H_{21} = \sigma \nabla u_1 \cdot \nabla u_2 = \frac{1}{4} \left(\sigma |\nabla (u_1 + u_2)|^2 - \sigma |\nabla (u_1 - u_2)|^2 \right).$$

Hence by use of 4 measurements ($J = 4$) in our set-up it will be possible to get the 3 measurements needed for the H_{ij}-model. Thus it is possible to find a unique and stable solution to the non-linear AET model.

1.3 Reading guide

In Chapter 2 we will explain the concept of linearising the model to get a linear system of PDE problems. After linearising the model we explain what results there exists for uniqueness and stability after which we decouple the system.

After this Chapter 3 introduces how the system is handled and implemented numerically and we look in to one of the errors we make in the implementation.

We then do reconstructions in Chapter 4 for a selected variety of phantoms. We inspect the reconstructions visually as well as numerically, where we compare it to the true solution by use of their difference in 2-norm. We then add gaussian noise to the interior data to see the effects of noisy data and then lastly try to improve the solutions by Tikhonov regularization.

In Chapter 5 we see what happens when we only have access to do measurements on part of the boundary. We use the boundary condition on either half or a quarter of the boundary and the other part of the boundary will have a homogeneous Dirichlet condition.

Lastly in Chapter 6 we turn our attention to singular value decomposition. We analyse, whether the systems are ill-posed by use of the singular values as well as examining the structure of the reconstructions by looking at the singular vectors. After this we use truncated singular value decomposition to get rid of the artefacts in the solutions.

CHAPTER 2

Linearization

To solve and analyse the problem we want to linearise the boundary value problem. We consider the non-linear interior data map defined by

$$\mathcal{H}_j : \sigma \mapsto \sigma |\nabla u|^2. \tag{2.1}$$

$\mathcal{H}_j$ can be considered as a mapping

$$\mathcal{H}_j : L^\infty_+(\Omega) \to L^1(\Omega).$$

$\sigma \in L^\infty_+(\Omega)$ and observing that $u \in H^1_0(\Omega)$ we get that $\nabla u \in L^2(\Omega)$, which means that $\nabla u \cdot \nabla u \in L^1(\Omega)$.

We want to derive a linear approximation of this operator to arrive at a linear inverse problem for AET expressed as a linear system of PDE problems.

For the linearisation we need the definition of a Fréchet derivative.

Definition 2.1 (Fréchet derivative)
Let V and W be Banach spaces and $U \subset V$ be open. A map $\mathcal{F} : U \to W$ is called Fréchet

differentiable at $k \in U$ if there exists a bounded linear operator $\mathrm{d}\mathcal{F}|_k : V \to W$ such that

$$\lim_{h\to 0} \frac{\|\mathcal{F}(k+h) - \mathcal{F}(k) - \mathrm{d}\mathcal{F}|_k(h)\|_W}{\|h\|_V} = 0.$$

The linear map $\mathrm{d}\mathcal{F}$ is called the Fréchet derivative of $\mathcal{F}$ at k and its value in the direction $h \in V$ is denoted by $\mathrm{d}\mathcal{F}|_k(h)$. $\mathcal{F}$ is Fréchet differentiable in an open domain, if it is differentiable at every point in this domain [8].

Definition 2.2 ($C^\infty_+(\bar{\Omega})$)
$C^\infty_+(\bar{\Omega})$ consists of the functions $\sigma \in C^\infty(\bar{\Omega})$, which are each bounded below by a some positive constant, K_σ i.e.

$$K_\sigma \leq \sigma(x) \text{ for a.e. } x \in \Omega,$$

i.e. K_σ is the essential infimum of σ.

Remark *Note that for functions $\sigma \in C^\infty_+(\bar{\Omega})$, we also have that $\sigma \in L^\infty_+(\Omega)$*

We denote $\sigma^0 \in C^\infty_+(\bar{\Omega})$ as the *reference* conductivity and the corresponding *reference* potential by $u^0_j \in H^1(\Omega)$, which is the solution to (1.11) when σ is replaced by σ^0. The *reference* interior data is defined as $H^0_j = \sigma^0|\nabla u^0_j|^2$.

The map (2.1) relates the difference between $\sigma - \sigma^0$ to the corresponding difference in the interior data

$$\mathcal{H}_j(\sigma) - \mathcal{H}_j(\sigma^0) = H_j - H^0_j.$$

The Fréchet derivative (Definition 2.1) of $\mathcal{H}_j$ at σ^0 evaluated at $\sigma - \sigma^0$ is an approximation to $\mathcal{H}_j(\sigma) - \mathcal{H}_j(\sigma^0)$ to the first order.

We consider $\mathcal{H}_j$ as a two-step mapping

$$\mathcal{H}_j : \sigma \mapsto \{\sigma, u_j\} \mapsto \sigma|\nabla u_j|^2.$$

We start by analysing the first step, where we linearise the solution operator

$$\mathcal{U}_j : \sigma \mapsto u_j(\sigma),$$

for the problem (1.11) at the reference conductivity $\sigma^0 \in C^\infty_+(\bar{\Omega})$.

Lemma 2.4 *The solution operator $\mathcal{U}_j : \sigma \mapsto u_j(\sigma)$ for the problem (1.11) is Fréchet differentiable as an operator from $L^\infty_+(\Omega) \to H^1_0(\Omega)$ at $\sigma^0 \in C^\infty_+(\bar{\Omega})$.*

The Fréchet derivative in the direction $\delta\sigma \in L^\infty(\Omega)$ is given by

$$\mathrm{d}\mathcal{U}_j|_{\sigma^0}(\delta\sigma) = \delta u_j, \tag{2.2}$$

where $\delta u \in H_0^1(\Omega)$ is the unique weak solution to

$$\begin{cases} \nabla\cdot\sigma^0\nabla\delta u_j &= -\nabla\cdot\delta\sigma\nabla u_j^0 \quad \textit{in } \Omega \\ \delta u_j &= 0 \quad \textit{on } \partial\Omega \end{cases}, \tag{2.3}$$

where $u_j^0 \in H^1(\Omega)$ is the solution to (1.11)*, when σ is replaced by σ^0.*

PROOF. We need to show that $\mathrm{d}\mathcal{U}_j|_{\sigma^0}$ given by (2.2) is a bounded linear operator from $L_+^\infty(\Omega) \to H_0^1(\Omega)$ satisfying

$$\lim_{\delta\sigma\to 0} \frac{\|\mathcal{U}_j(\sigma^0+\delta\sigma) - \mathcal{U}_j(\sigma^0) - \mathrm{d}\mathcal{U}_j|_{\sigma^0}(\delta\sigma)\|_{H^1(\Omega)}}{\|\delta\sigma\|_{L^\infty(\Omega)}} = 0, \tag{2.4}$$

Linearity of $\mathrm{d}\mathcal{U}_j|_{\sigma^0}$ comes from (2.3), where it can be seen that δu_j is linear in $\delta\sigma$. This can be proven by taken the conductivities $\delta\sigma_A$ and $\delta\sigma_B$ and the corresponding solutions $\delta u_{A,j}$ and $\delta u_{B,j}$

$$\begin{aligned} -\nabla\cdot(\delta\sigma_A+\delta\sigma_B)\nabla u_j^0 &= -\nabla\cdot\delta\sigma_A\nabla u_j^0 - \nabla\cdot\delta\sigma_B\nabla u_j^0 \\ &= \nabla\cdot\sigma^0\nabla\delta u_{A,j} + \nabla\cdot\sigma^0\nabla\delta u_{B,j} \\ &= \nabla\cdot\sigma^0\nabla(\delta u_{A,j}+\delta u_{B,j}). \end{aligned}$$

To prove boundedness, note that $\delta\sigma \in L^\infty(\Omega)$, $\sigma^0 \in C_+^\infty(\bar{\Omega})$ and $\tilde{u} \in H^1(\Omega)$. By Theorem A.2 this is sufficient to conclude that $\delta u_j \in H_0^1(\Omega)$ i.e. $\mathrm{d}\mathcal{U}_j|_{\sigma^0}$ is bounded.

We are left to show that (2.4) is satisfied. We consider the weak form of (2.3)

$$\int_\Omega \sigma^0\nabla\delta u_j\cdot\nabla\phi dx = -\int_\Omega \delta\sigma\nabla u_j^0\cdot\nabla\phi dx, \ \ \forall\phi\in H_0^1(\Omega).$$

Choosing $\delta u_j \in H_0^1(\Omega)$ as the test function gives

$$\int_\Omega \sigma^0|\nabla\delta u_j|^2 dx = -\int_\Omega \delta\sigma\nabla u_j^0\cdot\nabla\delta u_j dx.$$

We take the absolute value on both sides. Note that the left-hand side consists of strictly

positive terms

$$\begin{aligned}\int_\Omega \sigma^0 |\nabla \delta u_j|^2 dx &= \left| \int_\Omega \delta\sigma \nabla u_j^0 \cdot \nabla \delta u_j dx \right| \\ &\leq \int_\Omega |\delta\sigma| \left| \nabla u_j^0 \cdot \nabla \delta u_j \right| dx \\ &\leq \|\delta\sigma\|_{L^\infty(\Omega)} \|\nabla u_j^0 \cdot \nabla \delta u_j\|_{L^1(\Omega)}.\end{aligned}$$

Since $\sigma^0 \in C_+^\infty(\bar{\Omega})$ we know that σ_0 is bounded from below by K_{σ^0} (See Definition 2.2) thus

$$\frac{1}{K_{\sigma^0}} \left| \int_\Omega \sigma^0 |\nabla \delta u_j|^2 dx \right| \geq \|\nabla \delta u_j\|_{L^2(\Omega)}^2.$$

Thus we have that

$$\|\nabla \delta u\|_{L^2(\Omega)}^2 \leq \frac{1}{K_{\sigma^0}} \cdot \|\delta\sigma\|_{L^\infty(\Omega)} \|\nabla u_j^0 \cdot \nabla \delta u_j\|_{L^1(\Omega)}$$

for some positive constant K_{σ^0}.
Using Hölder inequality and that $u_j^0 \in H_0^1(\Omega)$ we get that

$$\begin{aligned}\|\nabla \delta u\|_{L^2(\Omega)}^2 &\leq \frac{1}{K_{\sigma^0}} \cdot \|\delta\sigma\|_{L^\infty(\Omega)} \|\nabla u_j^0\|_{L^2(\Omega)} \|\nabla \delta u_j\|_{L^2(\Omega)} \\ &\leq C \cdot \|\delta\sigma\|_{L^\infty(\Omega)} \|\nabla \delta u_j\|_{L^2(\Omega)}\end{aligned}$$

for some positive constant C i.e.

$$\|\nabla \delta u_j\|_{L^2(\Omega)} \leq C \cdot \|\delta\sigma\|_{L^\infty(\Omega)}. \tag{2.5}$$

By definition of $\mathcal{U}_j$ we get the following relation in the weak form (where we use that $\sigma^0 + \delta\sigma = \sigma$)

$$\begin{aligned}\nabla \cdot (\sigma^0 + \delta\sigma) \nabla \big(\mathcal{U}_j(\sigma^0 + \delta\sigma) - \mathcal{U}_j(\sigma^0) - \mathrm{d}\mathcal{U}_j|_{\sigma^0}(\delta\sigma) \big) \\ = \nabla \cdot (\sigma^0 + \delta\sigma) \nabla \left(u_j - u_j^0 - \delta u_j \right) \\ = \underbrace{\nabla \cdot \sigma \nabla u_j}_{=0} - \underbrace{\nabla \cdot \sigma^0 \nabla u_j^0}_{=0} - \underbrace{\nabla \cdot \delta\sigma \nabla u_j^0}_{=-\nabla \cdot \sigma^0 \nabla \delta u_j} \\ - \nabla \cdot \sigma^0 \nabla \delta u_j - \nabla \cdot \delta\sigma \nabla \delta u_j \\ = - \nabla \cdot \delta\sigma \nabla \delta u_j\end{aligned}$$

Here u_j is the solution to (1.11) with $\sigma = \sigma^0 + \delta\sigma$. Thus we get a boundary value

problem where $\mathcal{E}_j = \mathcal{U}_j(\sigma^0 + \delta\sigma) - \mathcal{U}_j(\sigma^0) - \mathrm{d}\mathcal{U}_j|_{\sigma^0}(\delta\sigma)$ is the weak solution

$$\begin{cases} \nabla \cdot (\sigma^0 + \delta\sigma)\nabla\mathcal{E}_j &= -\nabla \cdot \delta\sigma\nabla\delta u_j \quad \text{in } \Omega \\ \mathcal{E}_j &= 0 \quad \text{on } \partial\Omega \end{cases}. \tag{2.6}$$

By the remark after Theorem A.2 we get

$$\|\mathcal{E}_j\|_{H_0^1(\Omega)} \leq C\frac{\|\delta\sigma\|_{L^\infty(\Omega)}}{K_\sigma}\|\nabla\delta u_j\|_{L^2(\Omega)}$$

with $\sigma = \sigma^0 + \delta\sigma$. Using (2.5) we get

$$\|\mathcal{U}_j(\sigma^0 + \delta\sigma) - \mathcal{U}_j(\sigma^0) - \mathrm{d}\mathcal{U}_j|_{\sigma^0}(\delta\sigma)\|_{H_0^1(\Omega)} \leq \tilde{C}\|\delta\sigma\|^2_{L^\infty(\Omega)}$$

where $\tilde{C}$ is independent of $\delta\sigma$, which proves (2.4).

□

We now turn out attention to $\mathcal{H}_j$.

Theorem 2.5 *Let $\sigma^0 \in C_+^\infty(\bar{\Omega})$ and denote $u_j^0 \in H^1(\Omega)$ as the solution to* (1.11)*, when σ is replaced by σ^0. Also let $|\nabla\tilde{u}|$ be bounded from below by a positive constant in Ω.*

$$\mathcal{H}_j : \sigma \mapsto \sigma|\nabla u_j|^2$$

is then Fréchet differentiable as an operator from $L_+^\infty(\Omega) \to L^1(\Omega)$ at σ^0 and the Fréchet derivative in the direction $\delta\sigma \in L^\infty(\Omega)$ is given by

$$\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma) = \delta\sigma|\nabla u_j^0|^2 + 2\sigma^0\nabla u_j^0 \cdot \nabla\delta u_j, \tag{2.7}$$

where $\delta u \in H_0^1(\Omega)$ is defined as in Lemma 2.4.

PROOF. We need to show that $\mathrm{d}\mathcal{H}_j|_{\sigma^0}$ given by (2.7) is a bounded linear operator from $L_+^\infty(\Omega) \to L^1(\Omega)$ satisfying

$$\lim_{\delta\sigma\to 0}\frac{\|\mathcal{H}_j(\sigma^0 + \delta\sigma) - \mathcal{H}_j(\sigma^0) - \mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma)\|_{L^1(\Omega)}}{\|\delta\sigma\|_{L^\infty(\Omega)}} = 0. \tag{2.8}$$

The linearity of $\mathrm{d}\mathcal{H}_j|_{\sigma^0}$ is obvious since by Lemma 2.4, δu_j is linear in $\delta\sigma$.

We now take the L^1-norm of $\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma)$ and use the triangle and Hölder inequality

$$\begin{aligned}
\|\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma)\|_{L^1(\Omega)} &= \left\|\delta\sigma|\nabla u_j^0|^2 + 2\sigma^0\nabla u_j^0\cdot\nabla\delta u_j\right\|_{L^1(\Omega)} \\
&\leq \left\|\delta\sigma|\nabla u_j^0|^2\right\|_{L^1(\Omega)} + 2\left\|\sigma^0\nabla u_j^0\cdot\nabla\delta u_j\right\|_{L^1(\Omega)} \\
&\leq \|\delta\sigma\|_{L^\infty(\Omega)}\left\|\nabla u_j^0\cdot\nabla u_j^0\right\|_{L^1(\Omega)} + 2\|\sigma^0\|_{L^\infty(\Omega)}\left\|\nabla u_j^0\cdot\nabla\delta u_j\right\|_{L^1(\Omega)} \\
&\leq \|\delta\sigma\|_{L^\infty(\Omega)}\underbrace{\left\|\nabla u_j^0\right\|_{L^2(\Omega)}\left\|\nabla u_j^0\right\|_{L^2(\Omega)}}_{=c_1} \\
&\quad + \underbrace{2\|\sigma^0\|_{L^\infty(\Omega)}\|\nabla u_j^0\|_{L^2(\Omega)}}_{=c_2}\|\nabla\delta u_j\|_{L^2(\Omega)} \\
&\leq c_1\|\delta\sigma\|_{L^\infty(\Omega)} + c_2\left\|\nabla\delta u_j\right\|_{L^2(\Omega)}.
\end{aligned}$$

Using the inequality from (2.5) we get that

$$\|\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma)\|_{L^1(\Omega)} \leq c\|\delta\sigma\|_{L^\infty(\Omega)}$$

for some constant c independent of $\delta\sigma$ and i.e. $\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma)$ is bounded.

We now look at $\mathcal{H}_j(\sigma^0+\delta\sigma) - \mathcal{H}_j(\sigma^0)$

$$\begin{aligned}
\mathcal{H}_j(\sigma^0+\delta\sigma) - \mathcal{H}_j(\sigma^0) &= (\sigma^0+\delta\sigma)|\nabla(u_j^0+\delta u_j)|^2 - \sigma^0|\nabla u_j^0|^2 \\
&= \sigma^0|\nabla u_j^0|^2 + \delta\sigma|\nabla u_j^0|^2 + \sigma^0|\nabla\delta u_j|^2 + \delta\sigma|\nabla\delta u_j|^2 \\
&\quad + 2\sigma^0\nabla u_j^0\cdot\nabla\delta u_j + 2\delta\sigma\nabla u_j^0\cdot\nabla\delta u_j - \sigma^0|\nabla u_j^0|^2 \\
&= \delta\sigma|\nabla u_j^0|^2 + 2\sigma^0\nabla u_j^0\cdot\nabla\delta u_j + \mathcal{O}(|\delta\sigma|^2).
\end{aligned}$$

Here $\mathcal{O}(|\delta\sigma|^2)$ means that the remainder converges to 0 faster than $\|\delta\sigma\|^2_{L^\infty(\Omega)}$. The result is evident by use of (2.5), since we have by this that $\nabla\delta u_j = \mathcal{O}(\delta\sigma)$.

Thus

$$\|\mathcal{H}_j(\sigma^0+\delta\sigma) - \mathcal{H}_j(\sigma^0) - \mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma)\|_{L^1(\Omega)} \leq \|\delta\sigma\|^2_{L^\infty(\Omega)},$$

which proves (2.8).

□

From the Theorem 2.5 we have that the Fréchet derivative in the direction $\delta\sigma \in L^\infty(\Omega)$ is given by

$$\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma) = \delta\sigma|\nabla u_j^0|^2 + 2\sigma^0\nabla u_j^0\cdot\nabla\delta u_j,$$

where $\delta u \in H_0^1(\Omega)$ is the unique weak solution to

$$\begin{cases} \nabla \cdot \sigma^0 \nabla \delta u_j &= -\nabla \cdot \delta\sigma \nabla u_j^0 \quad \text{in } \Omega \\ \delta u_j &= 0 \quad \text{on } \partial\Omega \end{cases} \tag{2.9}$$

By definition of the Fréchet derivative it follows that the solution $\delta\sigma \in L^\infty(\Omega)$ to

$$\mathrm{d}\mathcal{H}_j|_{\sigma^0}(\delta\sigma) = H_j - H_j^0 \tag{2.10}$$

is a first order approximation to $\sigma - \sigma^0$.

The equations (2.9) and (2.10) form a system of linear PDE's for $\left\{\delta\sigma, \{\delta u_j\}_{j=1}^J\right\}$. In matrix form this can be expressed as a boundary value problem

$$\begin{cases} \boldsymbol{\mathcal{L}}\boldsymbol{u} &= \boldsymbol{f} \quad \text{in } \Omega \\ \boldsymbol{\mathcal{B}}\boldsymbol{u} &= \boldsymbol{0} \quad \text{on } \partial\Omega \end{cases} \tag{2.11}$$

Here $\boldsymbol{\mathcal{L}}$ is a $2J \times (J+1)$ matrix defined as

$$\boldsymbol{\mathcal{L}} = \begin{bmatrix} \nabla \cdot ([\cdot]\nabla u_1^0) & \nabla \cdot (\sigma^0 \nabla [\cdot]) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla \cdot ([\cdot]\nabla u_J^0) & 0 & \cdots & \nabla \cdot (\sigma^0 \nabla [\cdot]) \\ |\nabla u_1^0|^2 & 2\sigma^0 \nabla u_1^0 \cdot \nabla [\cdot] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ |\nabla u_J^0|^2 & 0 & \cdots & 2\sigma^0 \nabla u_J^0 \cdot \nabla [\cdot] \end{bmatrix}.$$

$\boldsymbol{\mathcal{B}}$ is a boundary operator and defined as

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

The solution and the right-hand side is defined as

$$\boldsymbol{u} = \begin{bmatrix} \delta\sigma \\ \delta u_1 \\ \vdots \\ \delta u_J \end{bmatrix} \quad \text{and} \quad \boldsymbol{f} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ H_1 - H_1^0 \\ \vdots \\ H_J - H_J^0 \end{bmatrix}.$$

We then have a boundary value problem (2.11) that is a PDE formulation of the linearised inverse problem.

2.1 Uniqueness and stability of the system

In [3] Bal investigates the conditions to ensure stability and uniqueness in the linear system.

First off he concludes, when the system will be elliptic

Theorem 2.6 *In dimension $n = 2$, $J \geq 2$ is necessary for $\boldsymbol{\mathcal{A}}$ to be elliptic. $J = 2$ is sufficient for $n = 2$ if ∇u_1 and ∇u_2 are nowhere parallel or orthogonal.*
Moreover $J = 3$ is sufficient for $\boldsymbol{\mathcal{A}}$ to be elliptic in all dimensions $n \geq 2$ by choosing the boundary conditions $(f_1, f_2, f_1 + f_2)$ provided that ∇u_1 and ∇u_2 are nowhere parallel.

When a system is elliptic it is stable in the sense that there will not be more singularities in the solution than in the data i.e. the system is "good". This can be seen in relation to for example hyperbolic systems, where we might have propagation of singularities. If the reader wants to learn more about elliptic systems [6, Chapter 6] would be a place to start.

Next of Bal looks into the question of when the system will be injective i.e. when it will have a unique solution

Theorem 2.7 *A system $\boldsymbol{\mathcal{A}}$ is injective if the system is elliptic.*

2.2 Decoupling the system

We look at the system for a Dirichlet boundary condition f_j. This gives us the coupled system of equations

$$\begin{cases} \nabla \cdot \sigma^0 \nabla \delta u_j &= -\nabla \cdot \delta\sigma \nabla u_j^0 \quad \text{in } \Omega \\ \delta u_j &= 0 \quad \text{on } \partial\Omega \end{cases} \tag{2.12}$$

$$\delta\sigma |\nabla u_j^0|^2 + 2\sigma^0 \nabla u_j^0 \cdot \nabla \delta u_j = H_j - H_j^0, \tag{2.13}$$

where everything but $\delta\sigma$ and δu_j are theoretically known quantities.

We now want to decouple the system such that we only get a system for $\delta\sigma$. This is done for a reference conductivity $\sigma^0 = 1$.

Looking at (2.12) for $\sigma^0 = 1$

$$\begin{cases} \Delta \delta u_j &= -\nabla \cdot \delta\sigma \nabla u_j^0 \quad \text{in } \Omega \\ \delta u_j &= 0 \quad \text{on } \partial\Omega \end{cases}.$$

By [11, Section 7.3, Theorem 2] we know that this has the solution

$$\delta u_j(\boldsymbol{x}_0) = -\int_\Omega G(\boldsymbol{x}, \boldsymbol{x}_0) \nabla \cdot \delta\sigma(\boldsymbol{x}) \nabla u_j^0(\boldsymbol{x}) dx,$$

where $G(\boldsymbol{x}, \boldsymbol{x}_0)$ is the Green's function for the operator Δ on the domain Ω.

Using Green's first identity and the fact that G is 0 on the boundary $\partial\Omega$ we get that

$$\delta u_j(\boldsymbol{x}_0) = \int_\Omega \nabla G(\boldsymbol{x}, \boldsymbol{x}_0) \cdot \delta\sigma(\boldsymbol{x}) \nabla u_j^0(\boldsymbol{x}) dx. \tag{2.14}$$

Defining this relation between δu_j and $\delta\sigma$ as $\delta u_j = \boldsymbol{\mathcal{M}}_j \delta\sigma$, for some matrix $\boldsymbol{\mathcal{M}}_j$. We can plug this in to (2.13) and get a system only depending on $\delta\sigma$ (Again for $\sigma^0 = 1$)

$$\delta\sigma |\nabla u_j^0|^2 + 2\nabla u_j^0 \cdot \nabla \boldsymbol{\mathcal{M}}_j \delta\sigma = H_j - H_j^0.$$

So to analyse this we write it in the weak form for test functions $v \in L^\infty(\Omega)$

$$\int_\Omega \left[\delta\sigma |\nabla u_j^0|^2 + 2\nabla u_j^0 \cdot \nabla \boldsymbol{\mathcal{M}}_j \delta\sigma\right] \cdot v dx = \int_\Omega \left[H_j - H_j^0\right] \cdot v dx.$$

By use of the basis $\{\boldsymbol{\phi}_k\}$ of the space we can get the problem on a matrix form

$$\boldsymbol{\mathcal{A}}_j \boldsymbol{\delta\sigma} = \boldsymbol{b}_j,$$

where we have that

$$\boldsymbol{\mathcal{A}}_j[i,k] = \int_\Omega \left[\boldsymbol{\phi}_k |\nabla u_j^0|^2 + 2\nabla u_j^0 \cdot \nabla \boldsymbol{\mathcal{M}}_j \boldsymbol{\phi}_k\right] \cdot \boldsymbol{\phi}_i dx \tag{2.15}$$

and

$$\boldsymbol{b}_j[i] = \int_\Omega \left[H_j - H_j^0\right] \cdot \boldsymbol{\phi}_i dx. \tag{2.16}$$

Now for each j we will get a system and all of these can be combined to one big system for $\delta\sigma$

$$\boldsymbol{\mathcal{A}\delta\sigma} = \boldsymbol{b} \tag{2.17}$$

with

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} \boldsymbol{\mathcal{A}}_1 \\ \vdots \\ \boldsymbol{\mathcal{A}}_J \end{bmatrix}$$

and

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_J \end{bmatrix}.$$

CHAPTER 3

Setting up numerical problem

For the numerical implementation the problem is implemented in FEniCS, [1], and Python. The FEniCS Project is a collection of free software with an extensive list of features for automated, efficient solution of differential equations, which uses the Finite Element Method.
We choose the domain, Ω, to be the unit circle and here we generate a mesh. The mesh will be generated differently for each conductivity such that the boundary of internal discontinuities will be well defined. To compute the mesh we define the domain as a unit circle by

```
domain = Circle(Point(0, 0), 1, 150)
```

Here 150 means that the boundary of the circle is to be approximated by 150 points.

Subsets of the domain where we want sharp edges can then be defined as

```
domain.set_subdomain(1, Rectangle(Point(-0.2,-0.2), Point(0.2, 0.2)))
```

which then means that we define the first subdomain as a rectangle with the lower left corner as $(-0.2, -0.2)$ and the upper right corner as $(0.2, 0.2)$.

Running the following line of code

```
mesh = generate_mesh(domain, N)
```

for $N = 20$ gives the following mesh, see Figure 3.1, where it is clear that the boundary of the inner domain is clearly defined.

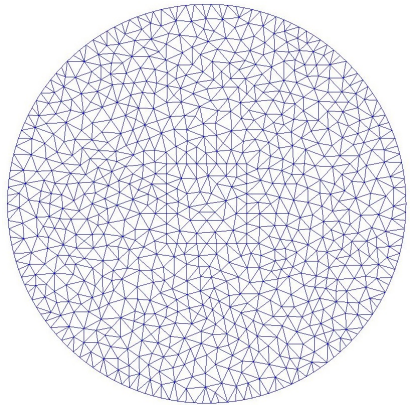


Figure 3.1: Example of mesh

In the implementation the Green's function for the domain is needed.

3.1 Green's function on the unit disk

To construct $\boldsymbol{\mathcal{M}}_j$ we need to find the Green's function for the domain. Now for the unit disk we get from [11, Section 7.4] that

$$G(\boldsymbol{x}, \boldsymbol{x}_0) = \frac{1}{2\pi} \left[\log \rho - \log \left(r_0 \rho^*\right)\right] \tag{3.1}$$

with $\rho = |\boldsymbol{x} - \boldsymbol{x}_0|$, $r_0 = |\boldsymbol{x}_0|$, $\rho^* = |\boldsymbol{x} - \boldsymbol{x}_0^*|$ and $\boldsymbol{x}_0^* = \boldsymbol{x}_0 / |\boldsymbol{x}_0|^2$.

Clearly there will be a problem for $\boldsymbol{x}_0 = 0$, since then $\boldsymbol{x}_0^*$ is not well-defined. We start by looking at the case where $\boldsymbol{x}_0 \neq 0$

For the implementation we need the gradient of the Green's function. To find this we look at

$$\rho^2 = |\boldsymbol{x} - \boldsymbol{x}_0|^2.$$

Differentiating this w.r.t. x gives

$$2\rho\nabla\rho = 2|\boldsymbol{x} - \boldsymbol{x}_0|,$$

which gives us

$$\nabla\rho = \frac{|\boldsymbol{x} - \boldsymbol{x}_0|}{\rho}. \tag{3.2}$$

For ρ^* we can do the same thing and get

$$\nabla\rho^* = \frac{|\boldsymbol{x} - \boldsymbol{x}_0^*|}{\rho^*}. \tag{3.3}$$

Thus using (3.2) and (3.3) the gradient of the Green's function is defined as

$$\begin{aligned}\nabla G(\boldsymbol{x}, \boldsymbol{x}_0) &= \frac{1}{2\pi}\left[\frac{\boldsymbol{x} - \boldsymbol{x}_0}{\rho^2} - \frac{\boldsymbol{x} - \boldsymbol{x}_0^*}{(\rho^*)^2}\right] \\ &= \frac{1}{2\pi}\left[\frac{\boldsymbol{x} - \boldsymbol{x}_0}{|\boldsymbol{x} - \boldsymbol{x}_0|^2} - \frac{\boldsymbol{x} - \boldsymbol{x}_0^*}{|\boldsymbol{x} - \boldsymbol{x}_0^*|^2}\right].\end{aligned} \tag{3.4}$$

For $\boldsymbol{x}_0 = 0$ we need to check what happens as $\boldsymbol{x}_0 \to 0$.

The first term of (3.1) is well defined for $\boldsymbol{x}_0 = 0$ so we focus on the second term

$$\log(r_0\rho^*) = \log\left(|\boldsymbol{x}_0| \cdot \left|\boldsymbol{x} - \frac{\boldsymbol{x}_0}{|\boldsymbol{x}_0|^2}\right|\right) = \log\left(\left|\boldsymbol{x} \cdot |\boldsymbol{x}_0| - \frac{\boldsymbol{x}_0}{|\boldsymbol{x}_0|}\right|\right)$$

Looking at the interior part of the logarithm we can make an upper and a lower bound

$$1 - |\boldsymbol{x}| \cdot |\boldsymbol{x}_0| = \frac{|\boldsymbol{x}_0|}{|\boldsymbol{x}_0|} - |\boldsymbol{x}| \cdot |\boldsymbol{x}_0| \leq \left|\boldsymbol{x} \cdot |\boldsymbol{x}_0| - \frac{\boldsymbol{x}_0}{|\boldsymbol{x}_0|}\right| \leq |\boldsymbol{x}| \cdot |\boldsymbol{x}_0| + \frac{|\boldsymbol{x}_0|}{|\boldsymbol{x}_0|} = |\boldsymbol{x}| \cdot |\boldsymbol{x}_0| + 1$$

As $\boldsymbol{x}_0 \to 0$ both the upper and lower bound tends to 1 thus

$$\log(r_0\rho^*) \to 0 \text{ as } \boldsymbol{x}_0 \to 0.$$

Thus the Green's function for $\boldsymbol{x}_0 = 0$ is defined as

$$G(\boldsymbol{x}, 0) = \frac{1}{2\pi}\log|\boldsymbol{x}|, \tag{3.5}$$

which gives us that

$$\nabla G(\boldsymbol{x}, 0) = \frac{1}{2\pi} \frac{\boldsymbol{x}}{|\boldsymbol{x}|^2}. \tag{3.6}$$

3.1.1 Green's function - numerically

Green's function and the gradient is singular as $\boldsymbol{x} \to \boldsymbol{x}_0$ so this need to be handled in the implementation.

In Python the gradient is set to 0 whenever $\boldsymbol{x} = \boldsymbol{x}_0$ to avoid any problems in the implementation.

We need to ensure that this is not a big error to make, so the singularity needs to be investigated. If we look at the gradient, it goes like $\frac{1}{|\boldsymbol{x}-\boldsymbol{x}_0|}$, thus we integrate this

$$\int_\Omega \frac{1}{|\boldsymbol{x} - \boldsymbol{x}_0|} d\boldsymbol{x}.$$

To integrate we make a small epsilon ball around the singularity $B(\boldsymbol{x}_0, \epsilon)$, see Figure 3.2,

$$\int_\Omega \frac{1}{|\boldsymbol{x} - \boldsymbol{x}_0|} d\boldsymbol{x} = \int_{B(\boldsymbol{x}_0,\epsilon)} \frac{1}{|\boldsymbol{x} - \boldsymbol{x}_0|} d\boldsymbol{x} + \int_{\Omega \setminus B(\boldsymbol{x}_0,\epsilon)} \frac{1}{|\boldsymbol{x} - \boldsymbol{x}_0|} d\boldsymbol{x}.$$

The area away from the singularity is no problem so we look at the small ball and

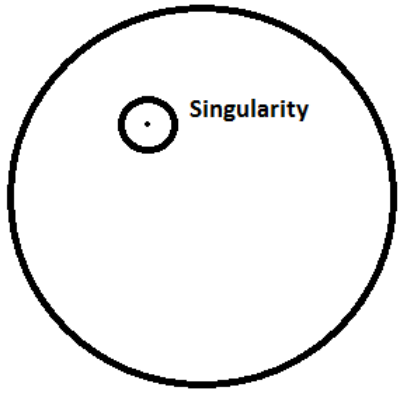


Figure 3.2: Small ball around the singularity

switch to polar coordinates

$$\int_{B(\boldsymbol{x}_0,\epsilon)} \frac{1}{|\boldsymbol{x}-\boldsymbol{x}_0|} d\boldsymbol{x} = \int_0^{2\pi} \int_0^{\epsilon} \frac{1}{r} r dr d\theta = 2\pi\epsilon,$$

thus

$$\int_{\Omega} \frac{1}{|\boldsymbol{x}-\boldsymbol{x}_0|} d\boldsymbol{x} = 2\pi\epsilon + \int_{\Omega\setminus B(\boldsymbol{x}_0,\epsilon)} \frac{1}{|\boldsymbol{x}-\boldsymbol{x}_0|} d\boldsymbol{x}.$$

Thus the contribution of the small ball to the integral converge to 0 as $\epsilon \to 0$, thus the finer the mesh the better the implementation

Thus in implementation $\nabla G(\boldsymbol{x}, \boldsymbol{x}_0)$ is computed by setting the singularities to 0. The code is seen below

```
cord = mesh.coordinates()
K = mesh.num_vertices()
NabGarray = np.zeros((K,K,2))

# Compute x-x_0/|x-x_0|
def xx0absgrad(x,x0):
    y = (x-x0)
    return y/(y[0]**2+y[1]**2)

# Compute |x-x_0|
def xx0abs(x,x0):
    y = (x-x0)
    return (y[0]**2+y[1]**2)

# Compute the gradient of G
for i in range(K):
    x1 = cord[i]
    for j in range(K):
        if (j<>i):
            x2 = cord[j]
            if (x2[0]**2+x2[1]**2==0):
                NabGarray[i,j,:] = (1/(2*math.pi)*
                x1/(x1[0]**2+x1[1]**2))
            else:
                x2star =x2/(x2[0]**2+x2[1]**2)
                NabGarray[i,j,:] = (1/(2*math.pi)*
                (xx0absgrad(x1,x2)-xx0absgrad(x1,x2star)))
```

3.2 Problem Data

In FEniCS we need to compute all the needed data for the linearised problem. Thus we need H_j, u^0 and H_j^0.
This is easily done in FEniCS seeing as we can solve (1.1) in the variational form for a given conductivity, σ

$$\int_\Omega \sigma \nabla u \nabla v = 0$$

with the given Dirichlet boundary condition.
This is solved both for the reference conductivity $\tilde{\sigma} = 1$ and a chosen conductivity σ.
To solve this we use finite element method, so we need a function space to solve it on. Here the choice falls on "CG", which stands for Continuous Galerkin, which is the standard Lagrange family of elements. This is the standard choice when doing finite element method, where we approximate everything by use of continuous basis functions.

The function space, given the mesh, is defined as

```
V = FunctionSpace(mesh,"CG",1)
```

Initialy the problem is solved for the given reference and the given conductivity on the function space

```
u = TrialFunction(V)
v = TestFunction(V)

# lhs of the weak formulation
a = sigma*inner(grad(u),grad(v))*dx # sigma given earlier
aref = inner(grad(u),grad(v))*dx #reference sigma = 1

# rhs of weak formulation
uleft = Constant("0.00")
L = uleft*v*dx

# Boundary condition
f = Expression("x[0]") #cos(theta)
bcs = DirichletBC(V,f,"on_boundary")

u = Function(V)
uref = Function(V)

# Solve weak formulation
solve(a == L,u,bcs)
solve(aref == L,uref,bcs)
```

Here the weak formulation has a left hand-side however this is just an integral over 0 i.e. it is the 0 functional. The reason for doing this, is that we need everything in the solver on functional form, so we can't just write

```
a == 0
```

From this we can compute the interior data by

```
# Interior data
H = sigma*inner(grad(u),grad(u))
Href = inner(grad(uref),grad(uref))
```

Now using the computed interior data and the reference potential one can set up the system by first making a function that computes the matrix $\mathcal{M}_j$, see (2.14)

```
# For handling gradient, vector space
parameters.reorder_dofs_serial = False
Vvec = VectorFunctionSpace(mesh,"CG",1)
dof0 = Vvec.sub(0).dofmap().dofs()
dof1 = Vvec.sub(1).dofmap().dofs()

# Function for computing M
def Gint(dsigma):
    I = Function(V)
    NabG = Function(Vvec)
    Iarray = np.zeros((K))
    for i in range(K):
        nabarray = NabG.vector().array()
        nabarray[dof0] = NabGarray[:,i,0]
        nabarray[dof1] = NabGarray[:,i,1]
        NabG.vector()[:] = nabarray
        Iarray[i] = assemble(inner(NabG,dsigma*grad(uref))*dx)
    I.vector()[:] = Iarray
    return I
```

Here the first line of code ensures that the numbering of vertices will be the same for both the computed matrix for ∇G and the mesh. The vector space, Vvec, is used for ∇G and dof0 and dof1 are pointers to the vertices in the mesh.

Now we can compute $\mathcal{A}_j$ and $\boldsymbol{b}_j$ by use of the basis vectors, see (2.15) and (2.16)

```
A = np.zeros((K,K))
b = np.zeros((K))
```

```
# Compute A and b
for i in range(K):
    ibasis = Function(V)
    ibasis.vector()[i] = 1
    delu = Gint(ibasis)
    for j in range(K):
        jbasis = Function(V)
        jbasis.vector()[j] = 1
        A[j,i] = (assemble((ibasis*inner(grad(uref),grad(uref))+
        2*inner(grad(uref),grad(delu)))*jbasis*dx))
    b[i] = assemble(inner(H-Href,ibasis)*dx)
```

$\boldsymbol{\mathcal{A}}_j$ and $\boldsymbol{b}_j$ is then saved for later use and they are also computed for several other boundary conditions.

The entire code put together can be seen in Appendix B.

3.2.1 Check of Green's function solution

We want to check that the solution we get using (2.14) is actually a close representation of the solution to (2.12) with $\sigma^0 = 1$.

We do this for three different $\delta\sigma$'s, one smooth and two discontinuous,

$$\delta\sigma_1(\boldsymbol{x}) = \begin{cases} 3e^{\frac{1}{|\boldsymbol{x}|^2-1/2}} & \text{for } |\boldsymbol{x}|^2 < 1/2 \\ 0 & \text{elsewhere} \end{cases}$$

and

$$\delta\sigma_2(x,y) = \begin{cases} 0.5 & \text{for } (x,y) \in [-0.2,0.2] \times [-0.2,0.2] \\ 0 & \text{elsewhere} \end{cases}$$

$$\sigma_3(x,y) = \begin{cases} 0.9 & \text{for } (x,y) \in [-0.2,0.1] \times [-0.5,-0.2] \\ 0.6 & \text{for } (x,y) \in [-0.5,-0.2] \times [0.2,0.4] \\ -0.3 & \text{for } (x,y) \in [-0.5,-0.2] \times [-0.2,0.0] \\ 0 & \text{elsewhere} \end{cases}$$

We compute the solution using (2.14) and using FEniCS we compute the solution to (2.12) for $f = \cos\theta$. This is done on an circular mesh with no subdomains.

In Figure 3.3, 3.4 and 3.5 the solutions are plotted. First the one using the Green's function and then the true one computed using FEniCS.

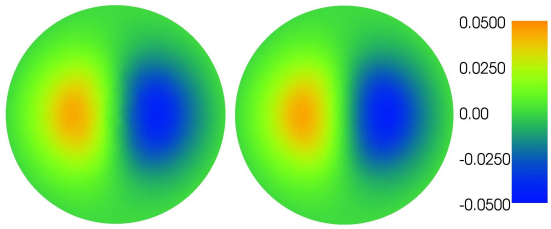


Figure 3.3: Green's function solution and FEniCS solution for $\delta\sigma_1$

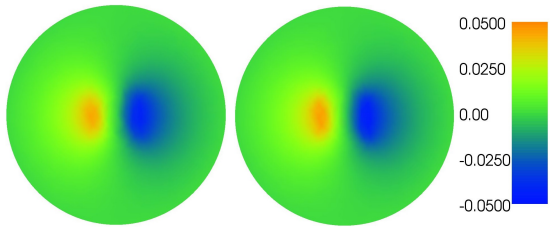


Figure 3.4: Green's function solution and FEniCS solution for $\delta\sigma_2$

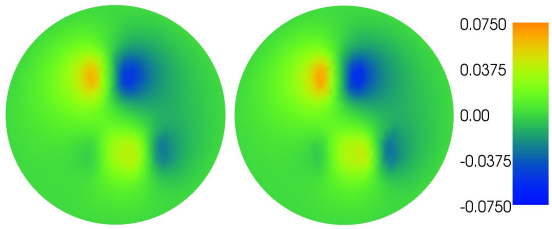


Figure 3.5: Green's function solution and FEniCS solution for $\delta\sigma_3$

The difference

$$\frac{\|\boldsymbol{\delta u}_{green} - \boldsymbol{\delta u}\|_2}{\|\boldsymbol{\delta u}\|_2}$$

is computed both for $f = \cos\theta$, $f = \sin\theta$ and $f = \cos\theta + \sin\theta$

	$\cos\theta$	$\sin\theta$	$\cos\theta + \sin\theta$
$\delta\sigma_1$	0.0182	0.0185	0.0188
$\delta\sigma_2$	0.0378	0.0396	0.0403
$\delta\sigma_3$	0.0529	0.0536	0.0499

It is clear that there is an error however it is not that big, less than 6%. It is clear that the more discontinuous and complex $\delta\sigma$ gets the bigger the error is.

The two solutions are both finite element solutions however one is approximated by use of a Green's function, where we set the gradient to 0 whenever $\boldsymbol{x} = \boldsymbol{x}_0$, which results in a difference. The difference between the two are however not that big, so we can assume that our numerical method is descent.

The code for this can be seen in Appendix B.

CHAPTER 4

Numerical results in FEniCS

After having computed $\boldsymbol{\mathcal{A}_j}$ and $\boldsymbol{b}_j$ for $j \in [1, J]$, where each j corresponds to a certain boundary condition the problem, (2.17) can be solved to find $\boldsymbol{\delta\sigma}$ using the least squares method.

Here we find the solution $\boldsymbol{\delta\sigma}$ that minimizes

$$\|\boldsymbol{\mathcal{A}\delta\sigma} - \boldsymbol{b}\|_2^2 \tag{4.1}$$

For $J \geq 2$ we will have an overdetermined system and thus we will have a unique least squares solution. Seeing as our discretization is evenly spaced one area will not be weighted more than another, thus the least squares method will be a reasonable to use.

4.1 Phantoms

We start by finding the linear solution for different types of conductivity. As stated earlier the mesh generated will depend on the conductivity to be able to generate sharp

edges around discontinuities. The code for all the meshes and the conductivities can be seen in Appendix B.

Also it should be worth nothing that the color bar is fitted to each example to get the best visualisation, but the color bar however will be consistent throughout one example.

We start by looking at a simple discontinuous conductivity

$$\sigma_1(x,y) = \begin{cases} 1.5 & \text{for } (x,y) \in [-0.2, 0.2] \times [-0.2, 0.2] \\ 1 & \text{elsewhere} \end{cases}.$$

For this we use the mesh, see Figure 4.1, that has a sharp edge at the boundary of the discontinuity (This is the same mesh as generated earlier), which results in the following conductivity plot

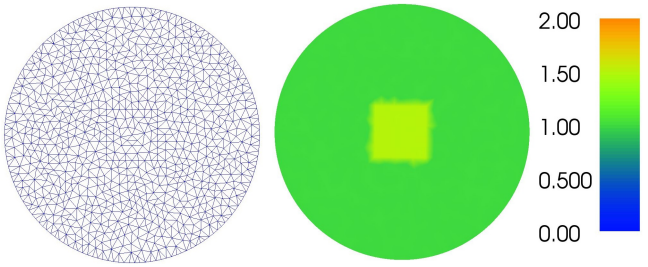


Figure 4.1: Mesh and simple discontinuous conductivity

Next we turn our attention to a continuous conductivity a so called mollifier. Here the choice falls on

$$m(\boldsymbol{x}) = \begin{cases} e^{\frac{1}{|\boldsymbol{x}|^2-1}} & \text{for } |\boldsymbol{x}| < 1 \\ 0 & \text{elsewhere} \end{cases}. \tag{4.2}$$

Seeing as we want the background to be 1 and the area different from 1 being away from the boundary the mollifier is changed to the following conductivity

$$\sigma_2(\boldsymbol{x}) = \begin{cases} 1 + 3e^{\frac{1}{|\boldsymbol{x}|^2-1/2}} & \text{for } |\boldsymbol{x}|^2 < 1/2 \\ 1 & \text{elsewhere} \end{cases}.$$

The mollifier is multiplied with 3 to get a larger contrast. The mesh choosen is just a circular domain with no subdomain and this gives the following plots, see Figure 4.2

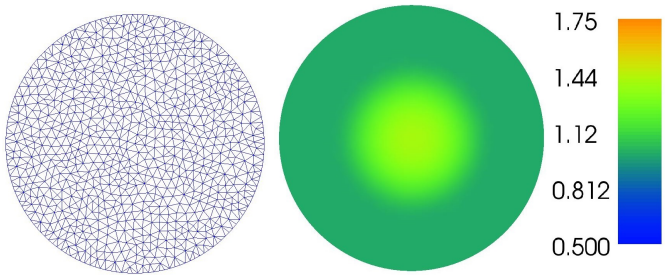


Figure 4.2: Mesh and conductivity for mollifier

Then we look at a conductivity similar to the one in the first example. Here however we use a lower contrast to see if we will have trouble if the difference between background and discontinuity is small

$$\sigma_3(x,y) = \begin{cases} 1.1 & \text{for } (x,y) \in [-0.2, 0.2] \times [-0.2, 0.2] \\ 1 & \text{elsewhere} \end{cases},$$

which plotted looks like, see Figure 4.3.

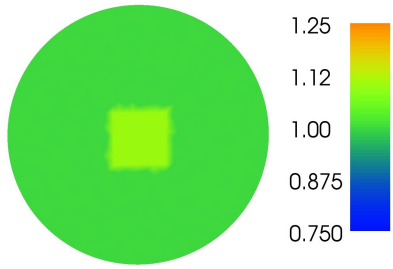


Figure 4.3: Conductivity with low contrast

After this we look at a conductivity which has a discontinuity that is close to the boundary to see if there will be any difference, when we place our discontinuity here.

$$\sigma_4(x,y) = \begin{cases} 1.5 & \text{for } (x,y) \in [0.6, 0.9] \times [-0.1, 0.1] \\ 1 & \text{elsewhere} \end{cases},$$

which gives the following plot, see Figure 4.4

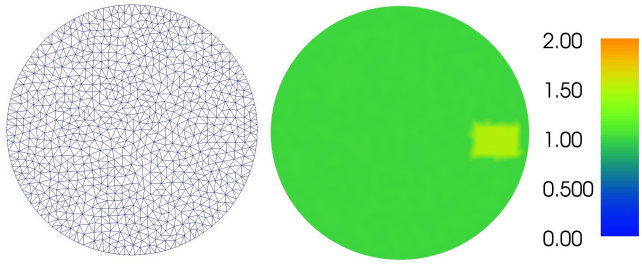


Figure 4.4: Mesh and conductivity with discontinuity near the boundary

As the final example we try to compute the solutions for a conductivity with several different discontinuities to check if we will have some difficulties with the areas in-between elements.

$$\sigma_5(x,y) = \begin{cases} 1.9 & \text{for } (x,y) \in [-0.2, 0.1] \times [-0.5, -0.2] \\ 1.6 & \text{for } (x,y) \in [-0.5, -0.2] \times [0.2, 0.4] \\ 0.7 & \text{for } (x,y) \in [-0.5, -0.2] \times [-0.2, 0.0] \\ 1 & \text{elsewhere} \end{cases},$$

which visually looks like, see Figure 4.5.

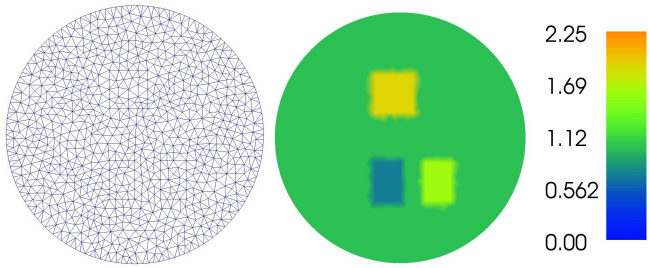


Figure 4.5: Mesh and conductivity with 3 discontinuities

4.2 Solutions

We then start computing the solutions for the different phantoms.
Other than inspecting the solutions visually we investigate the norm difference between

the true, $\boldsymbol{\sigma}$ and the approximative solution, $\tilde{\boldsymbol{\sigma}}$

$$\frac{\|\tilde{\boldsymbol{\sigma}} - \boldsymbol{\sigma}\|_2}{\|\boldsymbol{\sigma}\|_2} = \frac{\|\boldsymbol{\delta\sigma} + \boldsymbol{\sigma}^0 - \boldsymbol{\sigma}\|_2}{\|\boldsymbol{\sigma}\|_2}.$$

This is normalized by dividing with the norm of the true solution to be able to compare all the phantoms.

4.2.1 Simple discontinuous conductivity

We start by computing the solution using combinations of the following 4 different boundary conditions

$$f_1 = \cos\theta = x$$
$$f_2 = \cos\theta + \sin\theta = x + y$$
$$f_3 = \sin\theta = y$$
$$f_4 = \cos(2\theta) = x^2 - y^2$$

If we only use one boundary measurement the result is very unstable and does not give a valid approximation. This is also in accordance with the results in Section 2.1, where it is clear that the solution will not be stable for just one measurement. An example of the solution can be seen in Figure 4.6, where the used boundary condition is f_1.

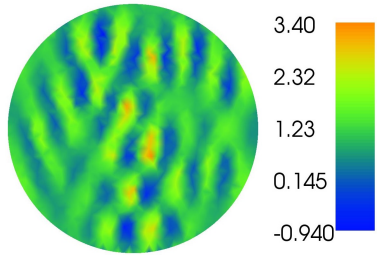
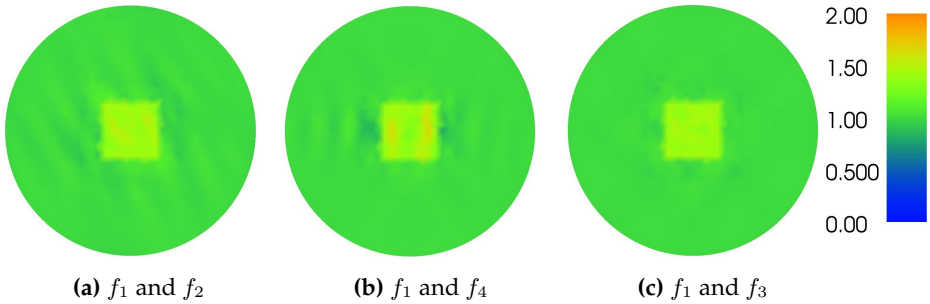


Figure 4.6: Solution with f_1 as the boundary condition

Next we compute the solution for different combinations of the boundary conditions for $J = 2, 3, 4$, see Figure 4.7 and 4.8

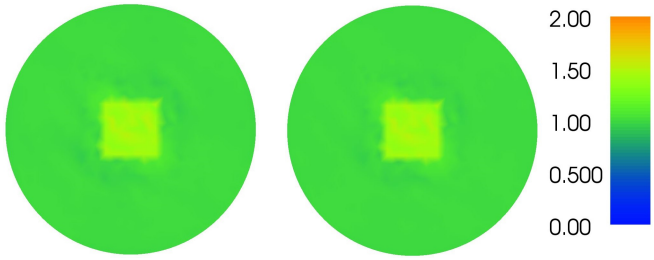
The computed norm differences are



(a) f_1 and f_2 **(b)** f_1 and f_4 **(c)** f_1 and f_3

(d) f_2 and f_3 **(e)** f_2 and f_4 **(f)** f_3 and f_4

Figure 4.7: Solutions for $J = 2$



(a) f_1, f_2 and f_3 **(b)** f_1, f_2, f_3 and f_4

Figure 4.8: Solutions for $J = 3, 4$

Norm difference		
$J = 2$	f_1, f_2	0.0323
	f_1, f_3	0.0242
	f_1, f_4	0.0360
	f_2, f_3	0.0343
	f_2, f_4	0.0364
	f_3, f_3	0.0375
$J = 3$	f_1, f_2, f_3	0.0262
$J = 4$	f_1, f_2, f_3, f_4	0.0253

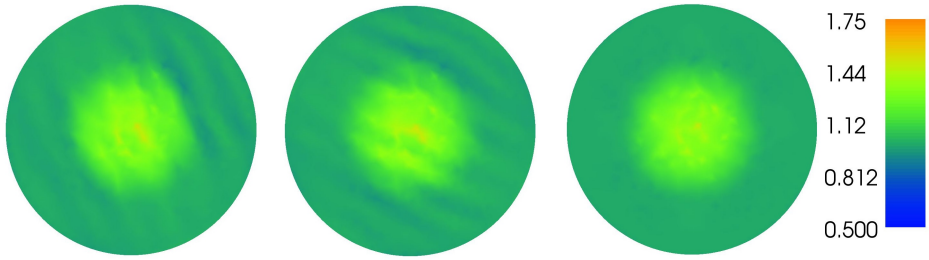
From the norm differences it is clear that f_4 is the one resulting in the worst reconstructions for $J = 2$. Thus this boundary condition will be ignored in the following examples also as it seems that the reconstruction does not get much better from using a 4th boundary conditions.
Surprisingly the best reconstruction is at $J = 2$ with f_1 and f_3, since these boundary conditions gives rise to perpendicular solutions, which according to Theorem 2.6, should not be the optimal for $J = 2$, if we want stability and uniqueness.

4.2.2 Mollifier

The used boundary conditions for this and the following examples will be

$$f_1 = \cos\theta = x$$
$$f_2 = \cos\theta + \sin\theta = x + y$$
$$f_3 = \sin\theta = y$$

as said in the previous example. This results in the following solutions, see Figure 4.9



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.9: Solutions for $J = 2, 3$

with the following norm differences

	Norm difference	
$J = 2$	f_1, f_2	0.0232
	f_2, f_3	0.0240
	f_1, f_3	0.0146
$J = 3$	f_1, f_2, f_3	0.0164

Again we get the same tendency as in the previous example. The best one is the one with f_1 and f_3 closely followed by the one with $J = 3$, while both f_1, f_2 and f_2, f_3 give somewhat the same results in norm difference. However visually they look different seeing as the parallel artefacts are at different angles.

4.2.3 Low contrast

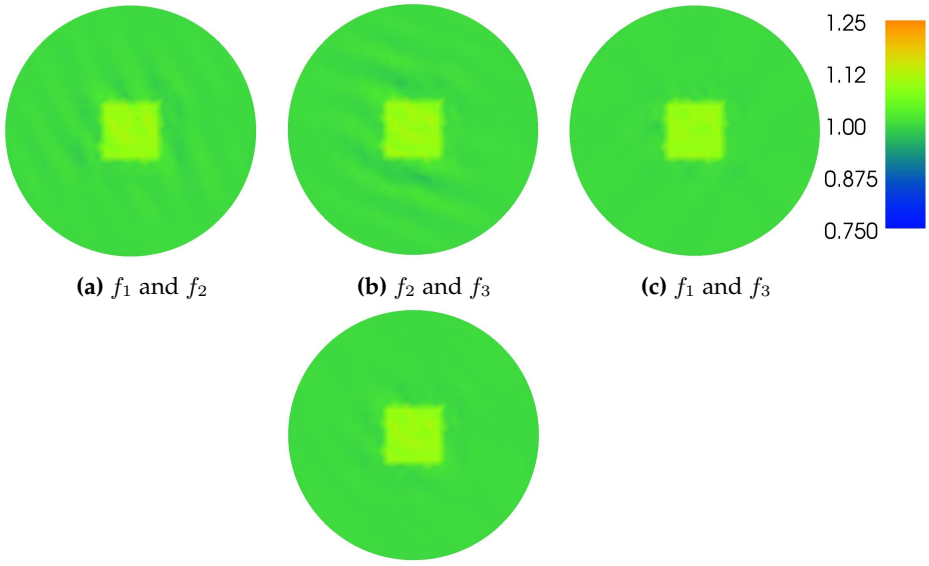
Again here we use the first three boundary conditions as stated in the previous example and this results in the following solutions, see Figure 4.10

Visually the solutions look better in the sense that the parallel anomalies are not as dominant even though we fitted the colour bar to have a very limited range. Thus it seems that the intensity of the anomalies depend on the intensity of the reconstruction. By intensity we mean the difference between the maximum and the minimum value i.e. the intensity of Figure 4.3 is $1.1 - 1 = 0.1$ whereas the one in Figure 4.1 will be $1.5 - 1 = 0.5$.

The computed norm difference is then

	Norm difference	
$J = 2$	f_1, f_2	0.00490
	f_2, f_3	0.00546
	f_1, f_3	0.00253
$J = 3$	f_1, f_2, f_3	0.00325

Again the same pattern arises where the best solution is the one with f_1 and f_3. Also as we normalise the norm difference by dividing by the norm of the true reconstruction



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.10: Solutions for $J = 2, 3$

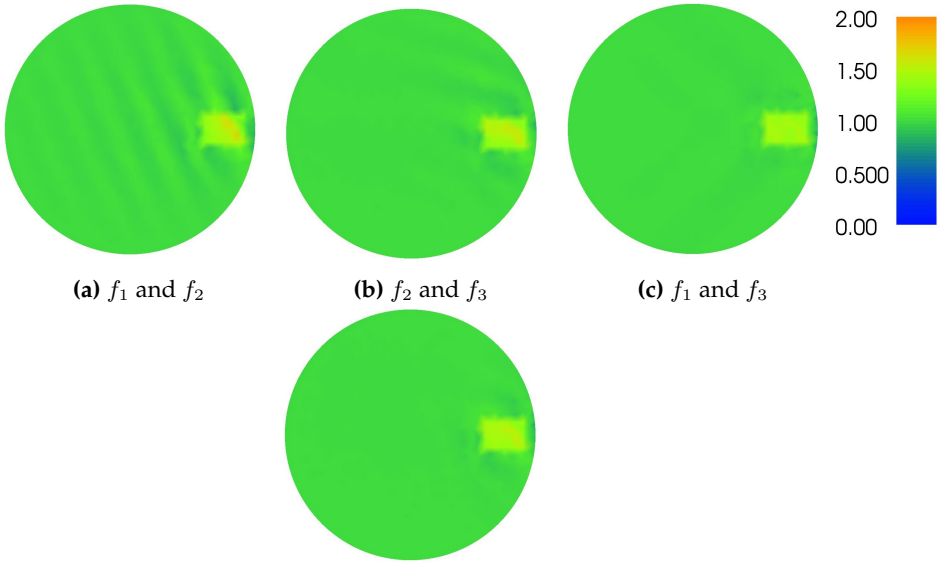
we can also see that our visual observation is evident in the norm difference as well. The anomalies are less dominant.

4.2.4 Near boundary

Computing the different solutions gives the following reconstructions, see Figure 4.11, and the following norm differences

Norm difference		
$J = 2$	f_1, f_2	0.0359
	f_2, f_3	0.0238
	f_1, f_3	0.0229
$J = 3$	f_1, f_2, f_3	0.0218

As with all the other examples the best is with f_1 and f_3 but it is also clear that using f_3 is way better than f_1 so the idea is that this has something to do with the placement



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.11: Solutions for $J = 2, 3$

of the discontinuity compared to the boundary condition. Thus to check this we do a similar computation but where the discontinuity is then placed at the top of the domain instead, which gives the following expression for the conductivity

$$\sigma_{4a}(x, y) = \begin{cases} 1.5 & \text{for } (x, y) \in [-0.1, 0.1] \times [0.6, 0.9] \\ 1 & \text{elsewhere} \end{cases}.$$

This results in the following mesh and conductivity, see Figure 4.12

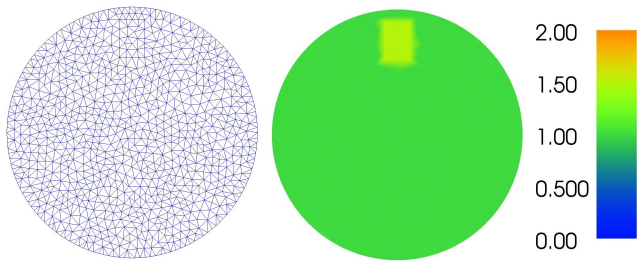
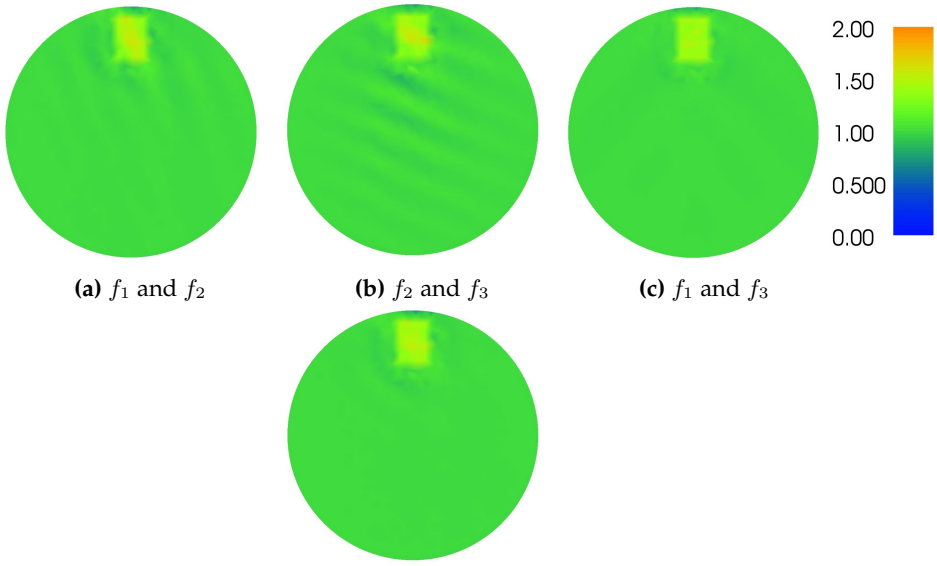


Figure 4.12: Mesh and another conductivity with discontinuity near the boundary

The reconstructions can be seen in Figure 4.13 and the norm difference can be seen below



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

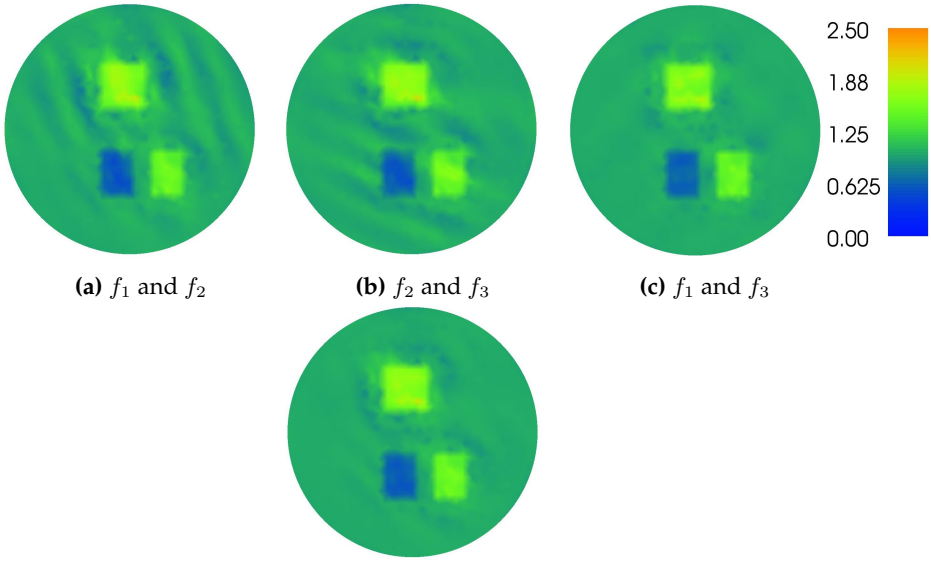
Figure 4.13: Solutions for $J = 2, 3$

Norm difference		
$J = 2$	f_1, f_2	0.0290
	f_2, f_3	0.0369
	f_1, f_3	0.0240
$J = 3$	f_1, f_2, f_3	0.0252

Here we clearly see what was expected. Now f_1 is a better choice than f_3, so the placement of the discontinuity clearly has something to say.
Another thing that is evident is that the anomalies are more dominant close to the discontinuity.

4.2.5 Three discontinuities

Again we compute the reconstructions for the same boundary conditions and get the following results, see Figure 4.14 and the table below



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.14: Solutions for $J = 2, 3$

Norm difference		
$J = 2$	f_1, f_2	0.0809
	f_2, f_3	0.0778
	f_1, f_3	0.0613
$J = 3$	f_1, f_2, f_3	0.0637

Again it is the same combination of boundary conditions that are the best however if we look at the difference in percent the difference is actually small.

The best in this example is 21 % better than the second best however in the first example the difference between the best and second best is 31 %. Also in general this conductivity is harder to reconstruct, seeing as the normalised norm difference is bigger here than in the previous examples. Visually however it is rather easy to see where the discontinuities are placed, so if used for example in medical diagnostics it still seems viable.

4.2.6 Conclusion of the initial solutions

In general the solution we get are rather good. It is clear to distinguish the areas different from the background and the contrast and the placement are in general quite good.

It is clear that all the different conductivities results in the same type of anomalies for the reconstruction. The same combination of boundary conditions results in the exact same errors in the background. There are these clear lines, artefacts, that is repeating across the domain.

Also it is very notable that the best reconstruction in all examples is the one for $J = 2$ with f_1 and f_3, which is very peculiar seeing as these will result in parallel solutions, which by Section 2.1 should be problematic.

The question is now will it be the same when we turn our attention to measurements that have noise.

4.3 Measurements with noise

As we don't live in a perfect world there will always be some sort of noise that is affecting our measurements. We already have a linearisation error that we get from linearising the model.

If we look at σ_1 we can calculate the linearisation error, $\|\boldsymbol{\mathcal{A}\delta\sigma}_{1,true} - \boldsymbol{b}\|_2$, for a couple of the boundary condition combinations and get that

Linearization error for σ_1

$J = 2$	f_1, f_2	0.005
	f_2, f_3	0.002
	f_1, f_3	0.004
$J = 3$	f_1, f_2, f_3	0.005

for the last conductivity, σ_5, which is the most complex we get

Linearization error for σ_5		
$J = 2$	f_1, f_2	0.012
	f_2, f_3	0.007
	f_1, f_3	0.012
$J = 3$	f_1, f_2, f_3	0.013

So in general the linearisation error will be at most 1.3%.

In practise the noise in measurements will naturally affect the boundary measurements, which then affects the interior data and then the reconstruction, see Figure 4.15.

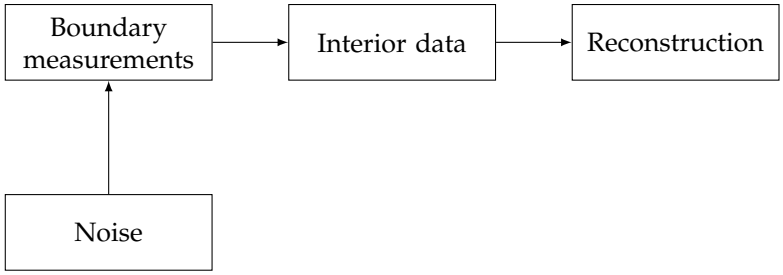


Figure 4.15: Noise in problem

The numerical implementation does not include the first step, where the boundary measurements are done, and analysing how the noise in boundary measurements will appear as noise in the interior data is not simple. Thus noise will instead be added directly to the interior data, see Figure 4.16.

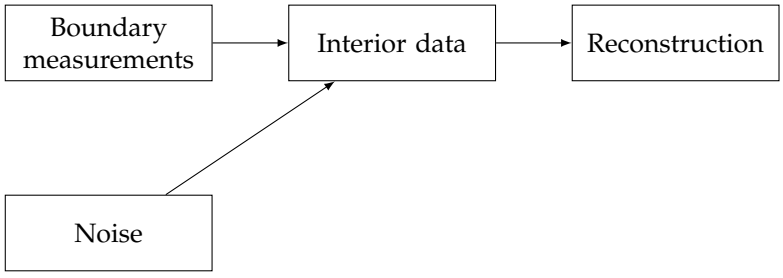


Figure 4.16: Noise in implementation

The noise is added to the interior data as Gaussian additive noise such that we go from

$H_j \to H_j^\epsilon$, where

$$H_j^\epsilon = H_j + \epsilon$$

Here ϵ is normal distributed with mean 0 and standard deviation $I \cdot \tau$, where I is the intensity of the image and τ is the desired noise level.

The code for making the noisy data can be seen in Appendix B, where we then in the original code have to change H to Hnoise.

4.3.1 Simple discontinuous conductivity

We start by looking at what happens to the solution, when we add noise to the interior data for the first simple example. This is done for noise levels $1, 5, 10\%$ and for the boundary conditions f_1, f_2 and f_3.

For 1% noise we get the reconstructions that are not so different from the original ones, which can be seen on the norm difference.

Norm difference for 1% noise

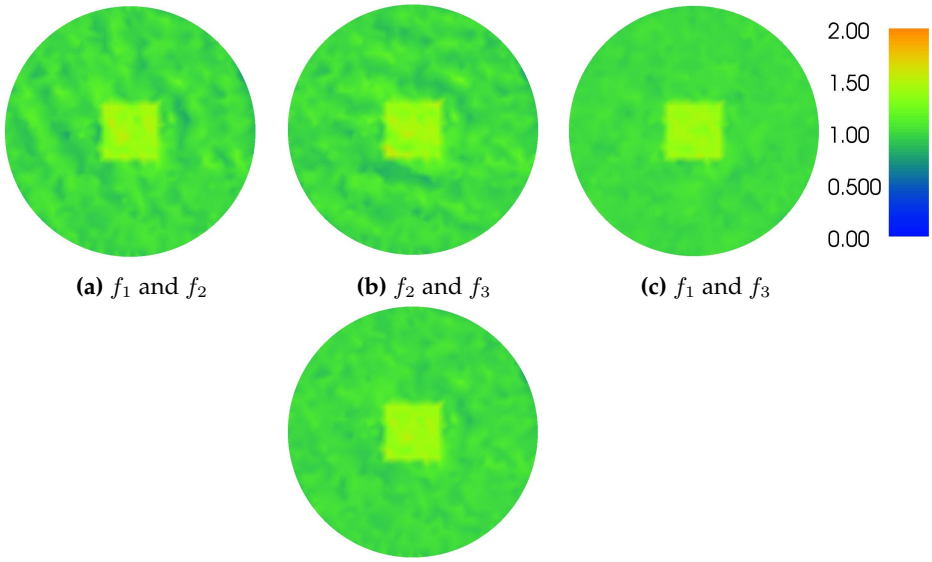
$J = 2$	f_1, f_2	0.0350
	f_2, f_3	0.0367
	f_1, f_3	0.0251
$J = 3$	f_1, f_2, f_3	0.0279

Visually this level of noise doesn't seem to affect the solution much either, which is consistent with it not being that much higher than the linearisation error for this case.

For 5% noise we get the reconstructions in Figure 4.17

Norm difference for 5% noise

$J = 2$	f_1, f_2	0.0657
	f_2, f_3	0.0664
	f_1, f_3	0.0371
$J = 3$	f_1, f_2, f_3	0.0489



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

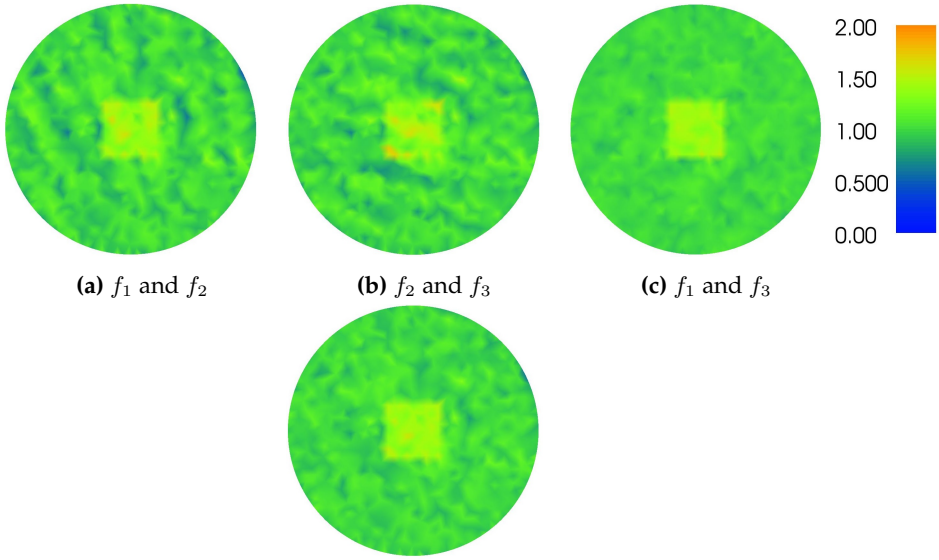
Figure 4.17: Solutions for $J = 2, 3$ with 5% noise

Here the noise is clearly visible both visually and in the norm difference. f_1, f_3 still gives the best reconstruction and it is very clear in the plot of the solution that noise isn't as prominent.

Finally for 10% the results are, see Figure 4.18

Norm difference for 10% **noise**		
$J = 2$	f_1, f_2	0.116
	f_2, f_3	0.117
	f_1, f_3	0.0600
$J = 3$	f_1, f_2, f_3	0.085

Now the noise is quite dominant. For f_1, f_2 and f_2, f_3 the discontinuity nearly mingles in with the rest and is thus far less clear. It is hard to see where the discontinuity actually is if we had no prior knowledge.
For the other two cases we are still able to see a clear distinction between the background and the discontinuity. This might have to do with the fact that the difference between the background and the discontinuity are quite prominent. However the noise



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.18: Solutions for $J = 2, 3$ with 10% noise

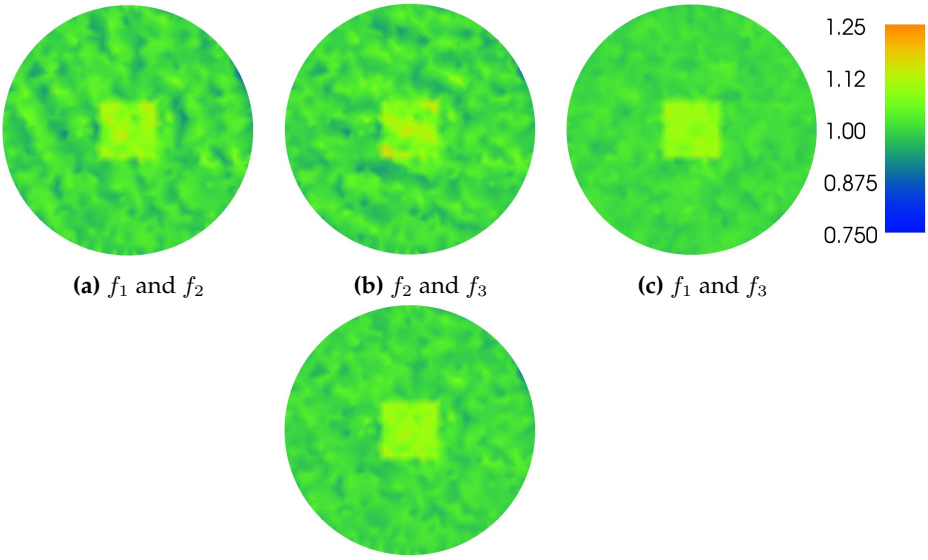
level is created depending on the intensity of H_j and one would imagine that the intensity of this would also vary depending on the intensity of σ.

4.3.2 Low contrast

To check that we get similar results depending on the intensity we compute reconstructions with noisy data for the example with lower contrast, see Figure 4.19.

Norm difference for 10% **noise**		
$J = 2$	f_1, f_2	0.1041
	f_2, f_3	0.1046
	f_1, f_3	0.1015
$J = 3$	f_1, f_2, f_3	0.1025

Here the we get quite similar solutions to the ones with higher contrast, but other than that the difference between the solutions numerically are not that far apart it just looks



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.19: Solutions for $J = 2, 3$ with 10% noise

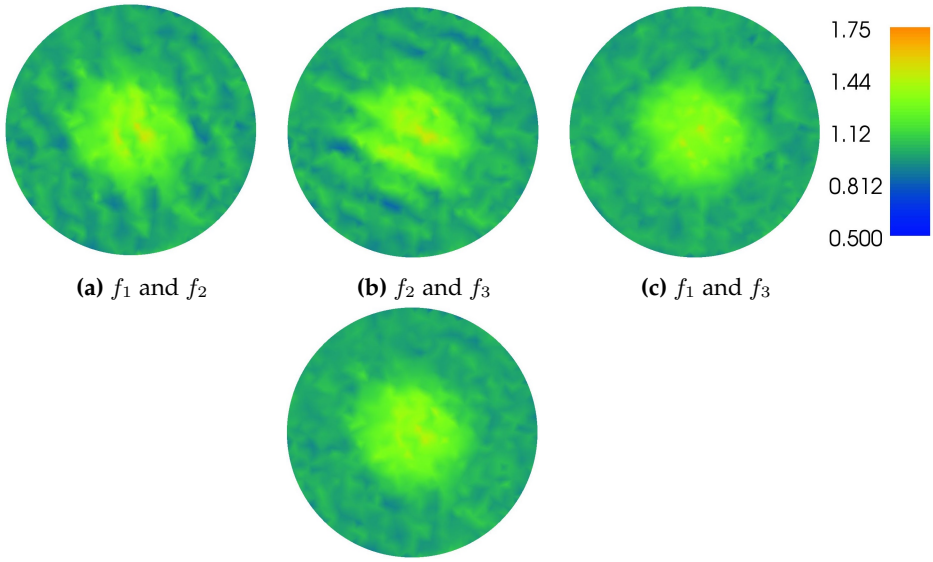
worse for f_1, f_2 and f_2, f_3. The reason for that is that the background have some sort of pattern,which disturbs the eye.

4.3.3 Mollifier

Next we add noise to the example of the mollifier to look at another type of example, where there is no sharp edges. Here 10% noise is added, which gives the following reconstructions, see Figure 4.20

Norm difference for 10% **noise**		
$J = 2$	f_1, f_2	0.0481
	f_2, f_3	0.0524
	f_1, f_3	0.0337
$J = 3$	f_1, f_2, f_3	0.0358

Here it is clear, that the true conductivity is quite unclear, and that the area gets really smeared. If we had no prior knowledge it would be hard to tell how the true conduc-



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

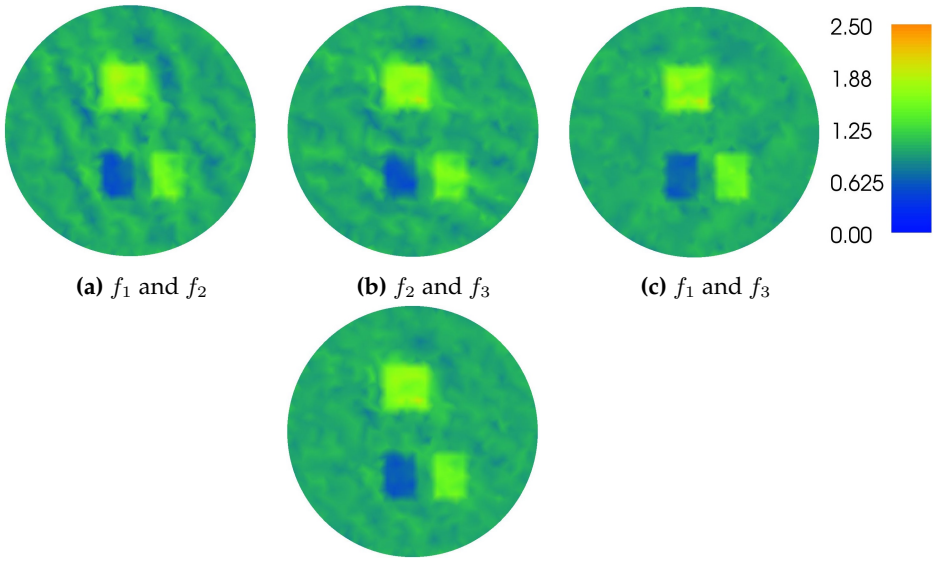
Figure 4.20: Solutions for $J = 2, 3$ with 10% noise

tivity looks exactly.

4.3.4 Three discontinuities

As a last noise example we add 5 and 10% noise to the example with the three discontinuities. This gives the following reconstructions, see Figure 4.21 and 4.22

Norm difference for 5% noise		
$J = 2$	f_1, f_2	0.1042
	f_2, f_3	0.0977
	f_1, f_3	0.0766
$J = 3$	f_1, f_2, f_3	0.0801



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

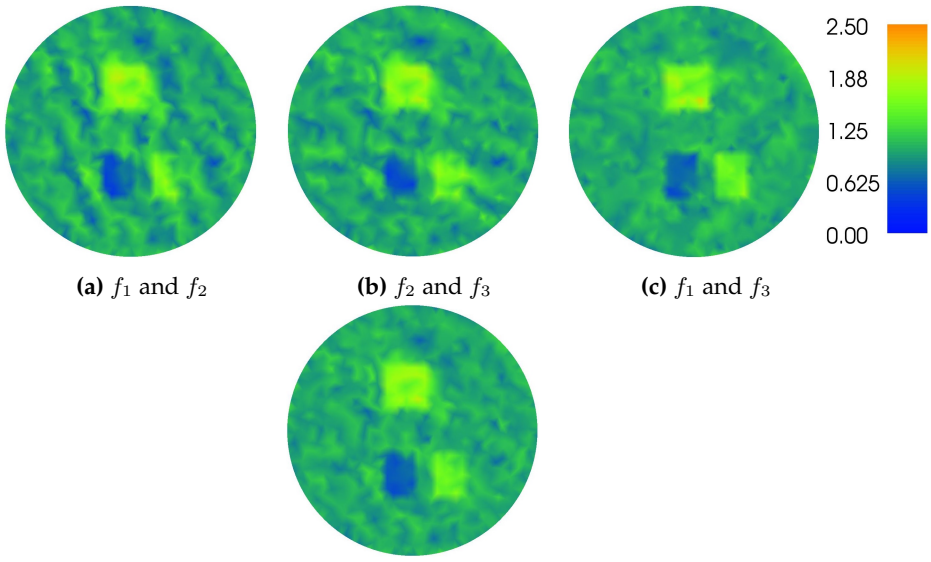
Figure 4.21: Solutions for $J = 2, 3$ with 5% noise

Norm difference for 10% **noise**		
$J = 2$	f_1, f_2	0.1559
	f_2, f_3	0.1426
	f_1, f_3	0.1086
$J = 3$	f_1, f_2, f_3	0.1149

It is clear, that the higher the noise the more the anomalies are intensified, and again the best reconstructions are for f_1, f_3 and f_1, f_2, f_3, where the background is more evenly noisy compared to the other two, where there are certain lines/directions.

4.3.5 Conclusion of noisy solutions

Again it is the same boundary conditions, that result in the best reconstructions but it is clear that with quite noisy data it is needed to do something to be able to actually get a view of the true conductivity. But in general we are actually still able in some manor to distinguish which areas, that are different from the background, but if clear edges is needed then it can be hard to tell exactly, where they are in some cases.



(a) f_1 and f_2 **(b)** f_2 and f_3 **(c)** f_1 and f_3

(d) f_1, f_2 and f_3

Figure 4.22: Solutions for $J = 2, 3$ with 10% noise

The question is now: "Is there something that can be done to get a more even reconstruction with clear discontinuities?"

4.4 Regularization

To stabilize the solution an idea is to introduce regularization. Regularization is normally added, when we have an ill-posed problem, but we start out with using regularization to see if something can be gained before looking in to the matter of ill-posedness. This means that we add another term, when we compute the solution, in the following manor

$$\|\boldsymbol{\mathcal{A}}\boldsymbol{\delta\sigma} - \boldsymbol{b}\|_2^2 + \alpha^2\|\boldsymbol{\delta\sigma}\|_2^2,$$

which corresponds to Tikhonov regularization with a Tikhonov matrix αI. α controls the trade off between fitting to the data and ensuring a small solution. This is used especially in noisy data, where fitting to the exact data would result in a wrong solution, but hopefully it could also be used to dampen some of the anomalies seen in the previous examples.

Numerically this is handled by changing the system to

$$\boldsymbol{\mathcal{A}}_\alpha = \begin{bmatrix} \boldsymbol{\mathcal{A}} \\ \alpha \boldsymbol{I} \end{bmatrix}$$

$$\boldsymbol{b}_\alpha = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix},$$

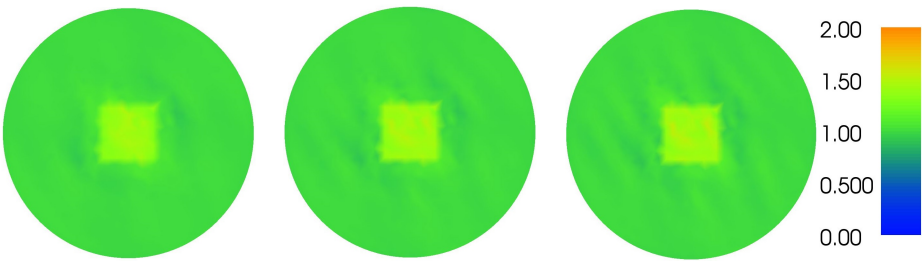
and then finding the least squares solution to

$$\|\boldsymbol{\mathcal{A}}_\alpha \boldsymbol{\delta\sigma} - \boldsymbol{b}_\alpha\|_2^2.$$

In regularization it is needed to choose α, the so-called regularization parameter, in some manor. One of the methods are called the L-curve method, see for example [7]. Here the logarithm of the residual norm is plotted against the logarithm of the solution norm and the optimal α is chosen as the corner point of the L, that the plot should hopefully look like. However the method doesn't always work and in this case the L-shape did not appear. Therefore the reconstruction will be computed for a number of parameters to see what the resulting reconstruction will be.

4.4.1 Noise free measurements

We start by trying to do regularization on noise free problems to see if it is possible to do something about the anomalies in the background.



(a) $\alpha = 0.0015$ **(b)** $\alpha = 0.0005$ **(c)** $\alpha = 0.0001$

Figure 4.23: Regularized solutions for $J = 2$ with f_1 and f_2

4.4.1.1 Simple Conductivity

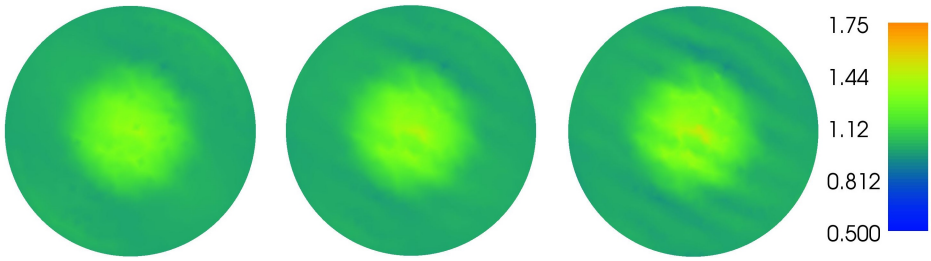
Here we start by looking at the first example for $J = 2$ with f_1, f_2. For different values of α between 0.002 and 0.0001 we get the following results, see Figure 4.23

α	**Norm difference**
0.0020	0.0468
0.0015	0.0404
0.0010	0.0346
0.00075	0.0325
0.0005	**0.0313**
0.0001	0.0321

The original reconstruction had a norm difference of 0.0323 so numerically the solution can be made a little better but it is not by much. Visually it is clear that a higher regularization results in a more even background however as this enforces a smaller norm of the reconstruction we lose some of the contrast.

4.4.1.2 Mollifier

Here we take the mollifier example for the boundary condition f_2, f_3.



(a) $\alpha = 0.0020$ **(b)** $\alpha = 0.0010$ **(c)** $\alpha = 0.0001$

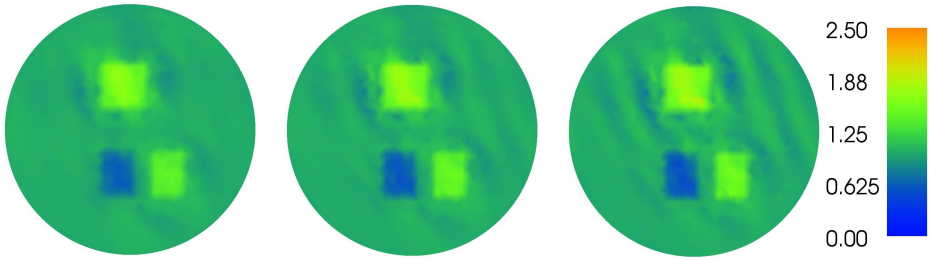
Figure 4.24: Regularized solutions for $J = 2$ with f_2 and f_3

α	**Norm difference**
0.0025	0.0278
0.0020	0.0232
0.0015	0.0197
0.0010	**0.0187**
0.0005	0.0212
0.0001	0.0238

The original norm difference was 0.0240, so all of the above solutions except the first one are numerically better than the initial reconstruction. The one that is numerically best still have some artefacts, but in general it seems that it is possible to get a reasonable solution that is still close to the original but doesn't have a lot of strange artefacts.

4.4.1.3 Three Discontinuities

As a final example of a noise free regularization we turn our attention to the last example, where we look at the one for f_1, f_2, since this is the one that gives that worst reconstruction. Here we get the following results, see Figure 4.25



(a) $\alpha = 0.0020$ **(b)** $\alpha = 0.0010$ **(c)** $\alpha = 0.0001$

Figure 4.25: Regularized solutions for $J = 2$ with f_1 and f_2

α	**Norm difference**
0.0025	0.0915
0.0020	0.0850
0.0015	0.0796
0.0010	**0.0767**
0.0005	0.0779
0.0001	0.0807

Here the original norm difference was 0.0809 thus again it is possible to create a better solution however the numerically better ones still have the artefacts in the background, where the ones that visually look more well-defined is actually worse numerically.

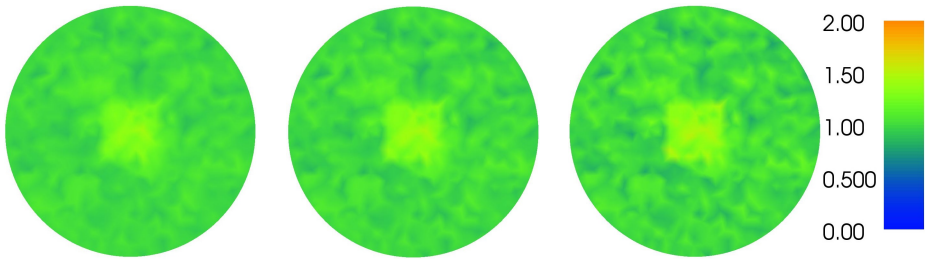
4.4.2 Noisy measurements

Now we look at what we can do with regularization, when we have noisy measurements i.e. in this case noisy interior data. We look at the case with 10% noise seeing as this is where the true conductivity gets really hard to recognize.

4.4.2.1 Simple Conductivity

We take a look at the simple conductivity for the boundary conditions f_2 and f_3 seeing as this is the one that gave the worst reconstructions with 10% noise.

Here we get the following reconstructions, see Figure 4.26



(a) $\alpha = 0.0035$ **(b)** $\alpha = 0.0025$ **(c)** $\alpha = 0.0015$

Figure 4.26: Regularized solutions for $J = 2$ with f_2 and f_3

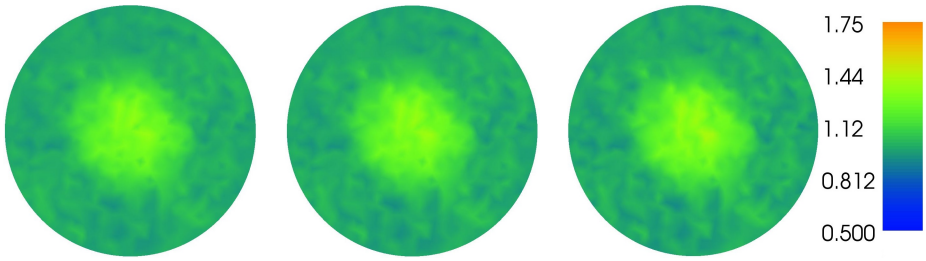
The norm differences are computed

α	**Norm difference**
0.0040	0.0801
0.0035	0.0781
0.0030	0.0768
0.0025	**0.0767**
0.0020	0.0787
0.0015	0.0837

The original noisy reconstruction had a norm difference of 0.1046, whereas the one with no noise had a norm difference of 0.0343. Some data is naturally lost, when noise is added, so we can't get as good a solution as the original one but we are still able to get a much better solution than the noisy one.

4.4.2.2 Mollifier

Now for the mollifier example we look at the one with the boundary conditions f_1 and f_2 again because that was the worst reconstruction. The 3 best regularized reconstructions can be seen in Figure 4.27



(a) $\alpha = 0.0025$ **(b)** $\alpha = 0.0020$ **(c)** $\alpha = 0.0015$

Figure 4.27: Regularized solutions for $J = 2$ with f_1 and f_2

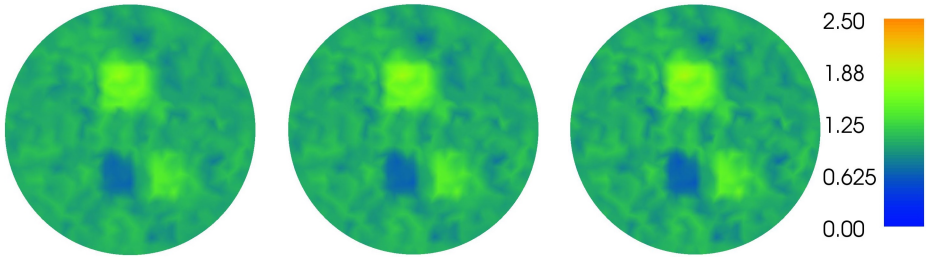
and the computed norm differences are

α	**Norm difference**
0.0035	0.0412
0.0030	0.0375
0.0025	0.0346
0.0020	**0.0329**
0.0015	0.0332
0.001	0.0362

The original noisy reconstruction has the norm difference 0.1042 and the non noisy one is 0.232. Here we are actually able to get a really good reconstruction with regularization. Here we want to reconstruct something smooth, which is easier, than the previous discontinuous example.

4.4.2.3 Three Discontinuities

Lastly we look at the one with 3 discontinuities for f_1 and f_2, which is the worst one. The reconstructions seen in Figure 4.28 is the 3 best results



(a) $\alpha = 0.0030$ **(b)** $\alpha = 0.0025$ **(c)** $\alpha = 0.0020$

Figure 4.28: Regularized solutions for $J = 2$ with f_1 and f_2

As for the norm difference we get

α	**Norm difference**
0.0035	0.1180
0.0030	0.1150
0.0025	**0.1136**
0.0020	0.1145
0.0015	0.1191

The noisy reconstruction had a norm difference of 0.1559 and the original one 0.0809 so all things considered the regularized reconstructions are rather good, but we still clearly see noise in the reconstruction.

4.4.3 Conclusion to regularization

It is clear that there is not much to gain using regularization for the noise-free examples however it is possible to gain a little better reconstruction especially for the mollifier seeing as this is smooth conductivity, which is easier to reconstruct. For the noisy cases however we see a much better reconstruction numerically, but the noise is still quite clear visually.

CHAPTER 5

Limited-view data

There might be cases, where we won't have access to the entire boundary, thus we can only measure part of the boundary. In these case we will only have limited-view data seeing as the data close to the unmeasured boundary might be hard to reconstruct anything from, as it will be rather unreliable. In this chapter we will investigate how good solutions we are able to produce in these cases. This is done by constructing two different boundary conditions, see Figure 5.1

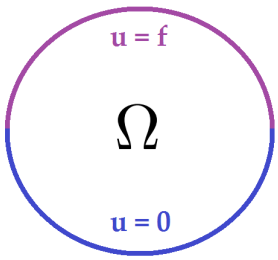


Figure 5.1: Partial boundary data

On one part (purple) of the boundary we have the Dirichlet condition like earlier on the other (blue) part we impose a homogeneous Dirichlet condition.

In this manor we will only have boundary data on the first part (purple) of the boundary. This does not need to be one half of the boundary but could be any portion.
Another way this could have been done was to use a homogeneous Neumann condition on the other (blue) boundary, $\frac{\partial u}{\partial n} = 0$.

The boundary conditions for the reference potential will also be the limited view ones.

We will expect that the area near the boundary where $u = 0$ will be unreliable and far from the true conductivity.

5.1 Half of the boundary

We start out by looking at a boundary like the one in Figure 5.1, where we have measurements on the upper half of the boundary. This is done using the following functions as boundary conditions to ensure that the boundary condition is continuous

$$
\begin{aligned}
h_1 &= \sin\theta = y \\
h_2 &= \sin 2\theta = 2xy \\
h_3 &= \sin^2\theta = y^2 \\
h_4 &= 2\sin\theta = 2y \\
h_5 &= 2\sin 2\theta = 4xy
\end{aligned}
$$

We use the boundary conditions h_4 and h_5 in hope that we might get something different by combining for example h_1 with h_5 instead of h_1 and h_2.

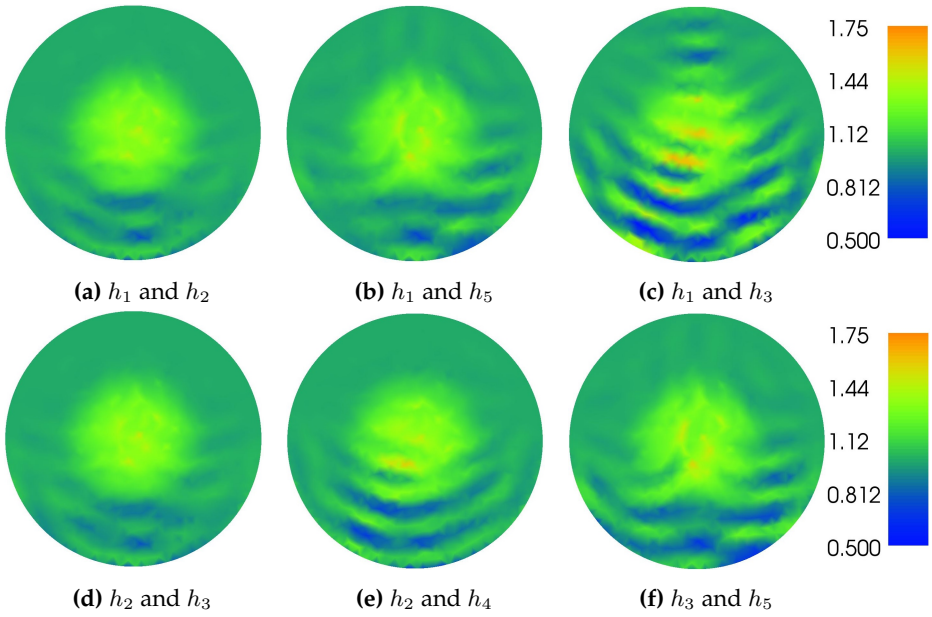
In Appendix B the code for defining the two different boundaries that should have different conditions and then there is code for how the boundary conditions should then be defined differently in the earlier code.

5.1.1 Mollifier

First we look at the mollifier case. Here we get the following results, see Figure 5.2

Here it is clear that the limited view gives us bad reconstructions for the part close to

the boundary with homogeneous Dirichlet condition as expected. Here we also see that h_1, h_2 gives results different from h_1, h_5.



(a) h_1 and h_2 **(b)** h_1 and h_5 **(c)** h_1 and h_3

(d) h_2 and h_3 **(e)** h_2 and h_4 **(f)** h_3 and h_5

Figure 5.2: Solutions for $J = 2$ with limited-view 50 %

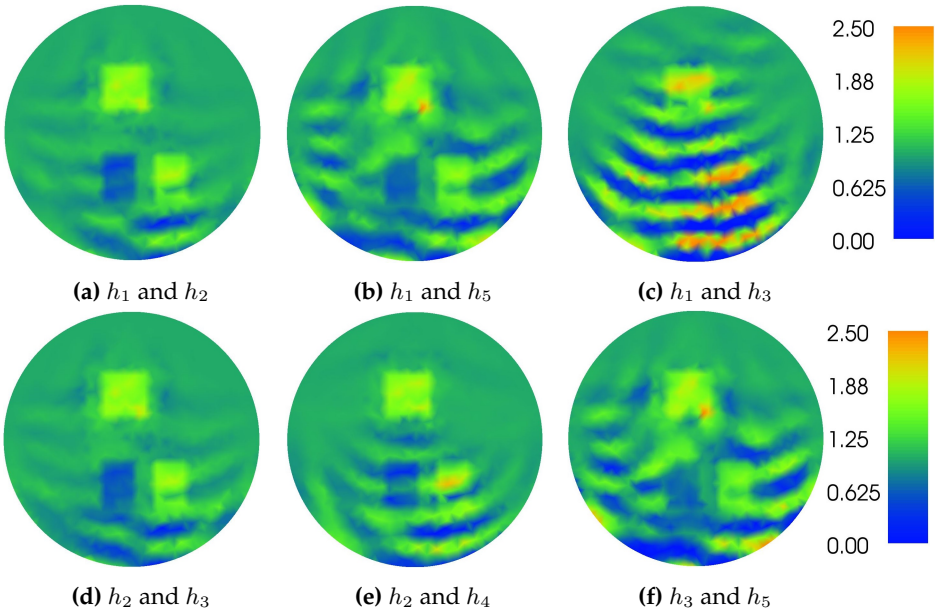
Norm difference for 50% **view**	
h_1, h_2	0.0362
h_1, h_5	0.0506
h_1, h_3	0.1183
h_2, h_3	0.0365
h_2, h_4	0.0677
h_3, h_5	0.0731

5.1.2 Three discontinuities

Next we look at the phantom with the three discontinuities. Here we get the following numerical results, see Figure 5.3

Again we see similar results as before. It is clear that the elements that are far away from the measured boundary gets harder to notice and with no prior knowledge you

would not reliable be ablt to say what the true conductivity looks like.



(a) h_1 and h_2 **(b)** h_1 and h_5 **(c)** h_1 and h_3

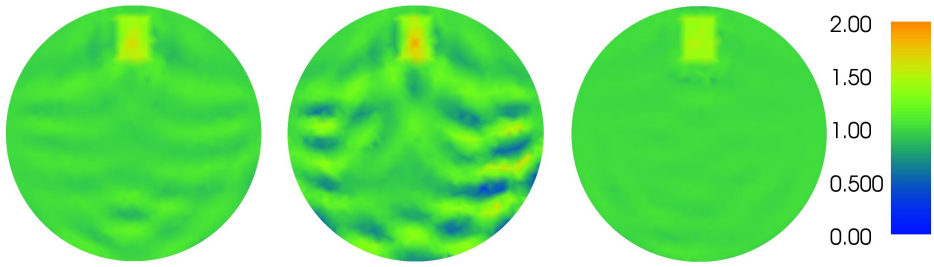
(d) h_2 and h_3 **(e)** h_2 and h_4 **(f)** h_3 and h_5

Figure 5.3: Solutions for $J = 2$ with limited-view 50 %

Norm difference for 50% **view**	
h_1, h_2	0.1360
h_1, h_5	0.2518
h_1, h_3	0.5263
h_2, h_3	0.1488
h_2, h_4	0.2096
h_3, h_5	0.3614

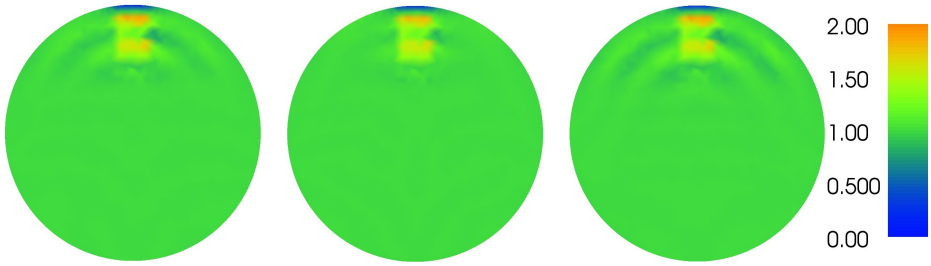
5.1.3 Near boundary

Lastly we take an element located close to the boundary. Here we do 2 different experiments - one where we measure the boundary close to the element and one where we measure the opposite boundary. This gives the following results, Figure 5.4 and 5.5



(a) h_1 and h_2 **(b)** h_1 and h_5 **(c)** h_2 and h_4

Figure 5.4: Solutions for $J = 2$ with limited-view 50 %, measure top half



(a) h_1 and h_2 **(b)** h_1 and h_5 **(c)** h_2 and h_4

Figure 5.5: Solutions for $J = 2$ with limited-view 50 %, measure bottom half

Norm difference for 50% view, top	
h_1, h_2	0.0517
h_1, h_5	0.1709
h_2, h_4	0.0252

Norm difference for 50% view, bottom	
h_1, h_2	0.0839
h_1, h_5	0.0730
h_2, h_4	0.0838

It is clear that when we measure the top half the box is more well-defined than when we measure the bottom. When we measure the bottom the box doesn't get as even and well-defined and it is hard to know, if there is something behind the box. The solution for the bottom view looks visually better, since we have an even area, where we in the other have artefacts. This makes sense due to the fact that we have more information

in the bottom, when we measure here. When we look at the norm differences it is clear that the bottom case is worse than the others except for the one with a lot of artefacts.

5.2 One quarter of the boundary

Now we limit the view to only the part of the boundary where $x, y > 0$ i.e. the part of the boundary that is located in the first quadrant.

Here we use the following functions as the boundary conditions

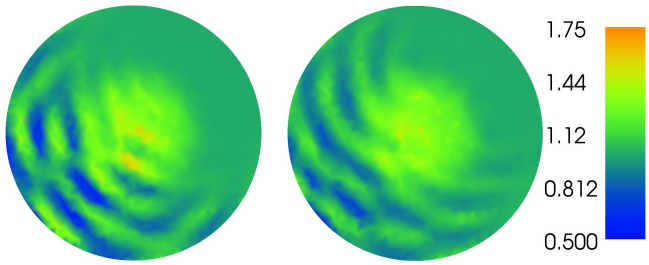
$$\begin{aligned} p_1 &= \sin 2\theta = 2xy \\ p_2 &= \sin 4\theta = 4xy\left(x^2 - y^2\right) \\ p_3 &= 2\sin 4\theta = 8xy\left(x^2 - y^2\right) \end{aligned}$$

again to ensure that we have continuous boundaries. We use p_3 to check if combining this with p_1 gives something better than with p_2.
We expect to see that the area far away from the boundary have heavy artefacts.

The code for the two different boundaries can be seen in Appendix B.

5.2.1 Mollifier

First for the case of the mollifier we get the following results, see Figure 5.6



(a) p_1 and p_2 **(b)** p_1 and p_3

Figure 5.6: Solutions for $J = 2$ with limited-view 25 %

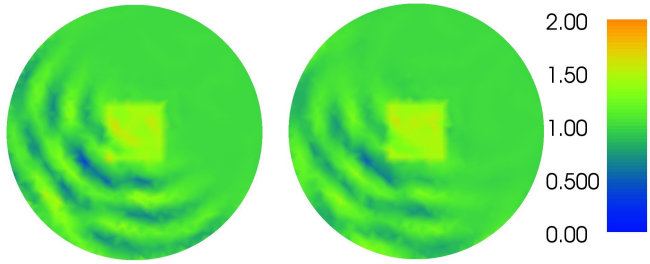
It is clear that using p_1, p_2 is worse than p_1, p_3.

Norm difference for 25% **view**	
p_1, p_2	0.1104
p_1, p_3	0.0725

As expected we see that there are heavy artefacts in the part that is not being measured.

5.2.2 Simple conductivity

Next for the case of the simple discontinuous conductivity we get the following results, see Figure 5.7



(a) p_1 and p_2 **(b)** p_1 and p_3

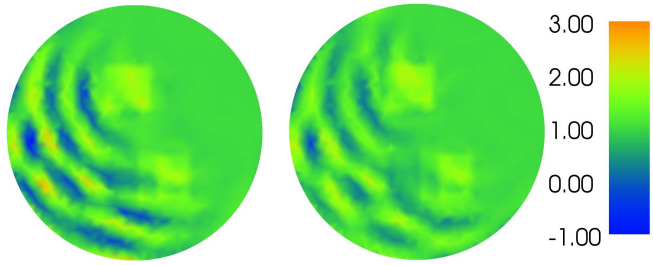
Figure 5.7: Solutions for $J = 2$ with limited-view 25 %

Norm difference for 25% **view**	
p_1, p_2	0.1170
p_1, p_3	0.0927

Again the same type of artefacts are appearing and it is the same set of boundary conditions that gives the best result.

5.2.3 Three discontinuities

For the three discontinuities we get quite oscillatory results (with a much higher and lower value than earlier), see Figure 5.8 - note the new colour scale



(a) p_1 and p_2 **(b)** p_1 and p_3

Figure 5.8: Solutions for $J = 2$ with limited-view 25 %

Norm difference for 25% **view**	
p_1, p_2	0.3948
p_1, p_3	0.2789

Here we see that the artefacts makes elements disappear or nearly obscure them. Also the higher an lower values fits with the ealier theory that the size of the artefacts depend on the intensity in th conductivity

5.3 Conclusion to limited-view

It is clear that when we use limited-view data we will have a bad reconstruction for the areas away from the measured boundary. The further away from the boundary we get, the worse it gets. Thus if we know that a certain element is close to one boundary it is best to use this boundary for the measurements.
It is clear that we will need some sort of regularization to try and get rid of the artefacts we get in the reconstructions and to see if it is actually possible to get usable results when we only have access to a portion of the boundary. One way would be to do Tikhonov regularization again as done in Chapter 4, but in the next Chapter we will see that there is also another way to handle this.

CHAPTER 6

Singular Value Decomposition (SVD)

For a given matrix, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n \times m}$, it is possible to do a singular value decomposition such that

$$\boldsymbol{\mathcal{A}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{U} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times m}$ are orthogonal matrices i.e. $\boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$.
$\boldsymbol{\Sigma} \in \mathbb{R}^{n \times m}$ is a rectangular diagonal matrix containing the singular values, s_i in the diagonal in descending order, $s_1 \geq s_2 \geq \cdots \geq s_{\min\{m,n\}} > 0$.

The solution computed earlier using least-squares $\|\boldsymbol{\mathcal{A}}\boldsymbol{\delta\sigma} - \boldsymbol{b}\|_2^2$ is in Python the same solution as the one using the pseudo-inverse of $\mathcal{A}$, symbolized by $\mathcal{A}^\dagger$ given by

$$\boldsymbol{\mathcal{A}}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T,$$

where $\boldsymbol{\Sigma}^\dagger \in \mathbb{R}^{m \times n}$ is a diagonal matrix with the reciprocal of the singular values in the diagonal.

The solution to the least-squares problem is then computed as

$$\boldsymbol{\delta\sigma} = \boldsymbol{\mathcal{A}}^{\dagger}\boldsymbol{b}.$$

One notable thing here is, that if some of the singular values are very small the reciprocal will be quite large. Thus this will be quite dominant and enhance any noise there might be in the data.

Thus one way to check whether the matrix is ill-posed is to computed the so-called conditional number of the matrix by

$$\text{cond}(\boldsymbol{\mathcal{A}}) = \frac{s_1}{s_{\min\{m,n\}}}, \tag{6.1}$$

if this is large, it means, that there is a big difference in the singular values, and thus it might be needed to do some sort of regularization to ensure, that we get a descent solution.

Another thing to look at would be the singular vectors, the columns of $\boldsymbol{V}$. In the solutions we see some very characteristic lines, artefacts, which might be explained by looking at the last singular vectors - the ones corresponding to the small singular values - seeing as these are the ones that will dominate the solution.

6.1 Truncated singular value decomposition (TSVD)

In truncated singular value decomposition we try to make the matrix better conditioned by cutting off some of the smallest singular values. Thus we construct the matrix $\boldsymbol{\mathcal{A}}_{\alpha}^{\dagger}$ in the manor

$$\boldsymbol{\mathcal{A}}_{\alpha}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}_{\alpha}^{\dagger}\boldsymbol{U}^{T},$$

where $\boldsymbol{\Sigma}_{\alpha}^{\dagger}$ is again the reciprocal of the singular value in the diagonal as long as $s_i > \alpha$, and the rest of the diagonal will be set to 0. The solution is then constructed as

$$\boldsymbol{\delta\sigma} = \boldsymbol{\mathcal{A}}_{\alpha}^{\dagger}\boldsymbol{b}.$$

The code for doing this in Python can be seen in Appendix B.

6.2 Mollifier

Seeing as the matrix will be the same for every conductivity as the matrix only depend on the reference conductivity, we should have the same condition number for all of the phantoms.

Remark *The matrices will be a bit different in the numerical implementation due to the different meshes used but the overall behaviour will be similar.*

Looking at the original system we compute the condition numbers for the different matrices used

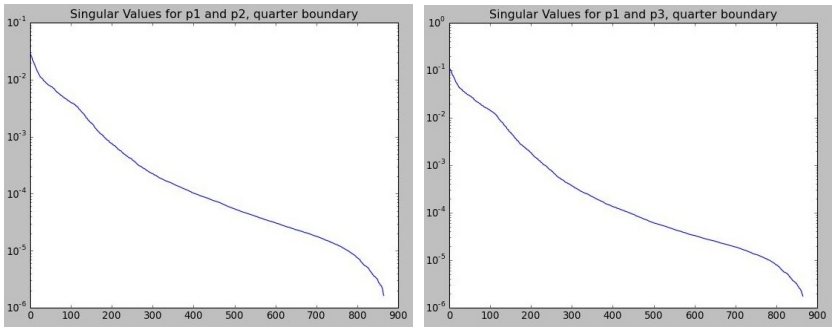
Condition number	
f_1, f_2	18.4
f_2, f_3	18.5
f_1, f_3	18.1
f_1, f_2, f_3	18.3

thus the condition number is about the same for each combination and not big, so it is quite well-posed.
If we however look at the limited-view case we get for the different boundary conditions

Condition number for 50% **view**	
h_1, h_2	1472.8
h_1, h_5	5859.2
h_1, h_3	434.6
h_2, h_3	2621.3
h_2, h_4	468.1
h_3, h_5	10360.0

Condition number for 25% **view**	
p_1, p_2	22903.3
p_1, p_3	85207.1

Another way to look at this is to plot the singular values, which also gives a good indication of the behaviour of the matrix, see Figure 6.1.

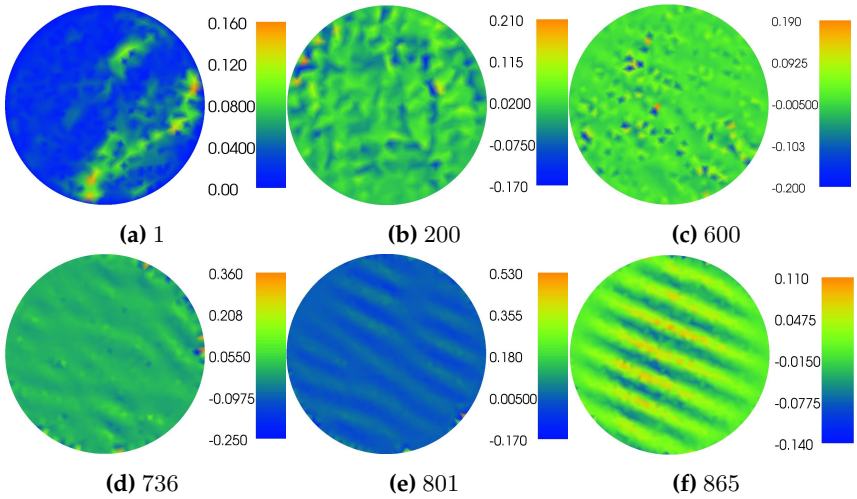


(a) p_1 and p_2 **(b)** p_1 and p_3

Figure 6.1: Singular values for mollifier with limited-view 25 %

Here we see that we have small values, which when we compute the solution by use of the pseudo-inverse, will dominate the solution, so for the limited-view we have matrices that is more or less ill-posed.

To see how the small values dominate the solution, we look at the singular vectors, since these are the building blocks, that the solutions consists of. Thus we take a look at the columns of the matrix $\boldsymbol{V}$. Seeing as there is 866 singular vectors in the numerical implementation, we can't look at all of them but in Figure 6.2 we see some for the boundary conditions f_2, f_3.



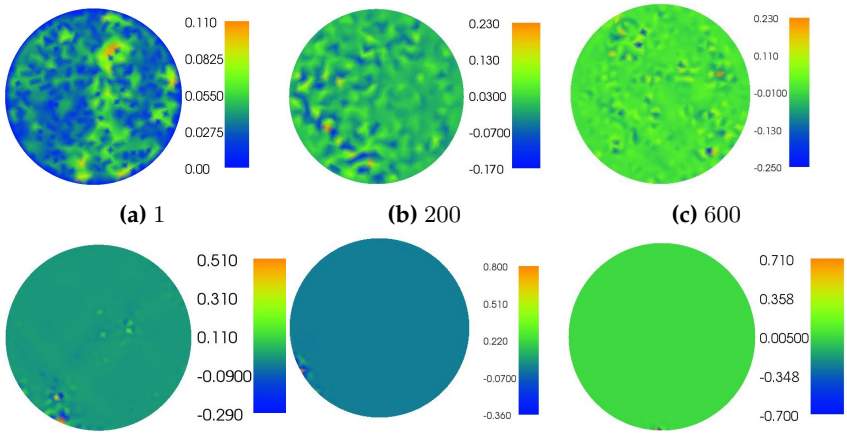
(a) 1 **(b)** 200 **(c)** 600

(d) 736 **(e)** 801 **(f)** 865

Figure 6.2: Singular vectors for mollifier f_2, f_3

In general the singular vectors, that show a pattern looking like the artefacts, are the ones corresponding to the small values, while the others don't seem to have much of a pattern. This fits with the fact that the smallest singular values should dominate the solutions. It should however be said that not all of the singular vectors corresponding to small values have that sort of pattern.

Seeing as we don't see a pattern when we use the boundary conditions f_1, f_3 we try and take a look at the same singular vectors as for f_2, f_3 to see the difference in behaviour.



(a) 1 **(b)** 200 **(c)** 600

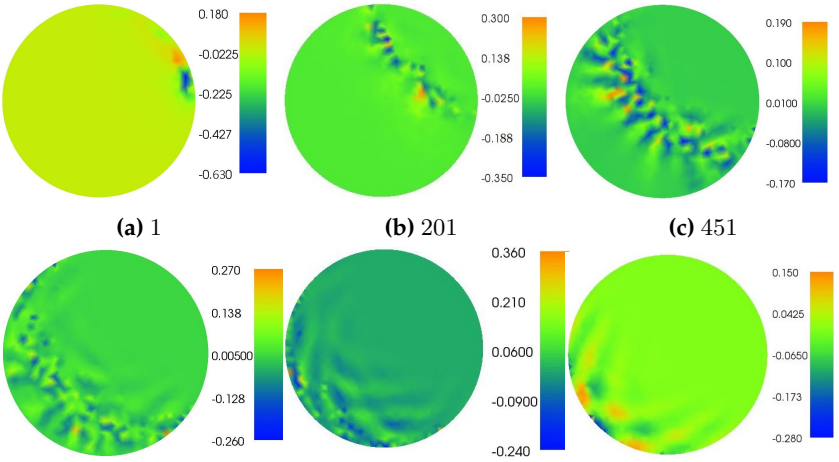
(d) 736 **(e)** 801 **(f)** 865

Figure 6.3: Singular vectors for mollifier f_1, f_3

Here it is clear, that the last singular vectors, that is the dominant part, does not have a behaviour to induce a pattern. Actually if you plot several of the last singular vector many of them look like the 865 one from Figure 6.3 i.e. they are rather smooth except for a small section near the boundary. So looking at the singular vectors it makes sense, that we don't see the same sort of behaviour for these conditions.

Looking at the ones for the case with a 25% view, Figure 6.4, we see that the information about the different areas of the domain is located in the different singular vectors. When the singular value gets lower the corresponding singular vector is dominating an area of the domain further away from the measured boundary.

The same is the case if one looks at the 50% view. (The images however are not included here.) We also here see that the singular vectors for small singular values dominate the area furthest from the measured boundary and the bigger the singular value the closer



(a) 1 **(b)** 201 **(c)** 451

(d) 650 **(e)** 801 **(f)** 865

Figure 6.4: Singular vectors for mollifier, 25% view, p_1, p_2

we move to the measured boundary.

This makes good sense. We know that the last singular values will be the ones dominating the reconstruction so it makes sense, that they represent the area away from the boundary seeing as this is where we will have bad reconstructions due to lack of good data.

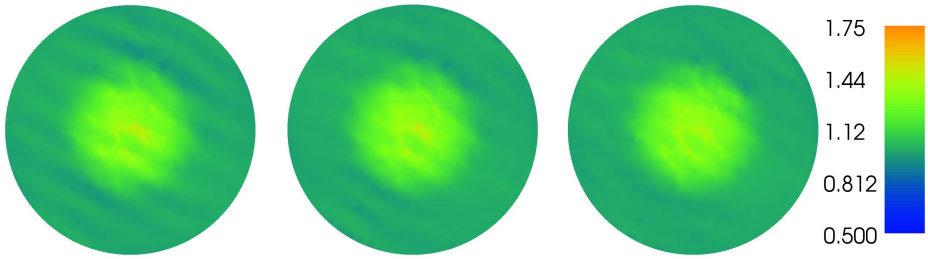
6.2.1 TSVD

We start by trying to do TSVD on the original solution even though the matrix is quite well-behaved. For the boundary conditions f_2 and f_3 we get the following results - here the number indicates the number of singular values that are set to 0.

Singular values = 0	Norm difference
25	0.0236
75	0.0209
125	0.0196
130	**0.0195**
140	0.0204

Here the best solution is when 130 singular values are excluded, which is considerable better than the original norm difference of 0.0240, but not better than the one for Tikohonov regularization that had a norm difference of 0.0187.

A couple of the solutions from TSVD can be seen in Figure 6.5



(a) -25 singular values **(b)** -75 singular values **(c)** -130 singular values

Figure 6.5: TSVD solutions for different number of excluded singular values

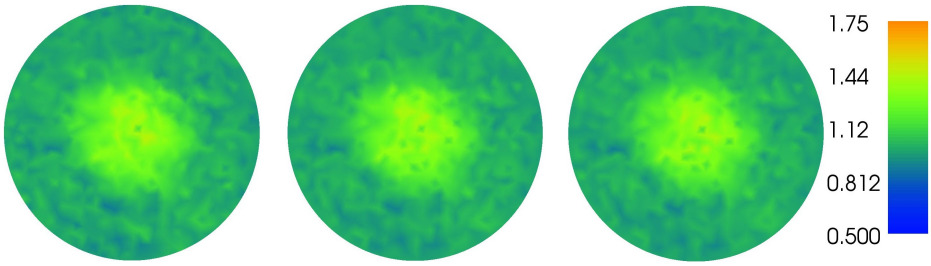
It is quite clear that the best reconstruction is close to not having the prominent artefacts like the earlier solutions.

In the case of the noisy data it should help to use TSVD seeing as small singular values will enhance the noise. The matrix is the same as before but the left-hand side, $\boldsymbol{b}$ is changed.

Here we get the following solutions from reconstructions, see Figure 6.6, and norm differences

Singular values = 0	Norm difference
50	0.0458
100	0.0407
150	0.0383
200	0.0373
250	0.0365
275	**0.0364**
300	0.0365
350	0.0367

We see that the noise is still quite dominant in the solution, which is probably due to the fact, that the last singular vectors only help to move the artefact lines and thus the noise



(a) -150 singular values **(b)** -275 singular values **(c)** -350 singular values

Figure 6.6: Noisy TSVD solutions

is still applied to the rest. Also the matrix is quite well-behaved, so the noise doesn't get enhanced much by the smallest singular values.

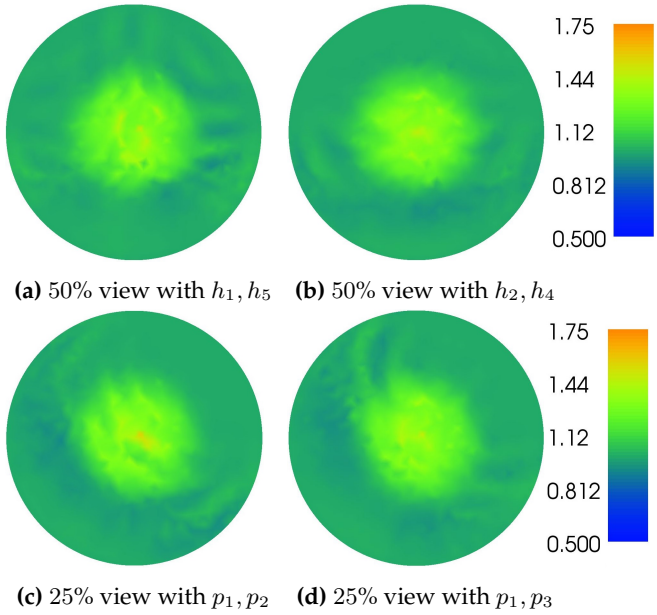
Lastly we do TSVD for the case of limited-view. Here the matrices are quite ill-conditioned so hopefully it should be possible to get a much better solution.

In Figure 6.7 the best solutions can be seen, where it is clear that we are able to get quite good reconstructions, seeing as we also showed before that it was the last singular values that dominated the lower part, which was the worst area.

We get the following norm differences from the computations.

Solution for 50% view

Singular values = 0	**Norm difference,** h_1, h_5	**Norm difference,** h_2, h_4
50	0.0363	0.0328
100	0.0335	0.0257
150	0.0312	0.0223
200	0.0290	0.0212
225	0.0279	**0.0208**
250	**0.0262**	0.0211
275	0.0264	0.0240



(a) 50% view with h_1, h_5 **(b)** 50% view with h_2, h_4

(c) 25% view with p_1, p_2 **(d)** 25% view with p_1, p_3

Figure 6.7: Best TSVD solutions for limited-view

Solution for 25% view

Singular values = 0	Norm difference, p_1, p_2	Norm difference, p_1, p_3
250	0.0318	0.0304
300	**0.0304**	0.0270
325	0.0308	**0.0269**
375	0.0312	0.0282

6.3 Three discontinuities

As noted before the condition number should be relatively the same for this case, since the matrix only depend on the reference. The mesh however is different, which is the cause of the slight variation we see in the condition numbers below.

Condition number	
f_1, f_2	17.3
f_2, f_3	15.8
f_1, f_3	18.1
f_1, f_2, f_3	15.9

Condition number for 50% view	
h_1, h_2	1590.4
h_1, h_5	6099.7
h_1, h_3	527.4
h_2, h_3	2796.7
h_2, h_4	489.4
h_3, h_5	10617.4

Condition number for 25% view	
p_1, p_2	21166.5
p_1, p_3	77165.8

6.3.1 TSVD

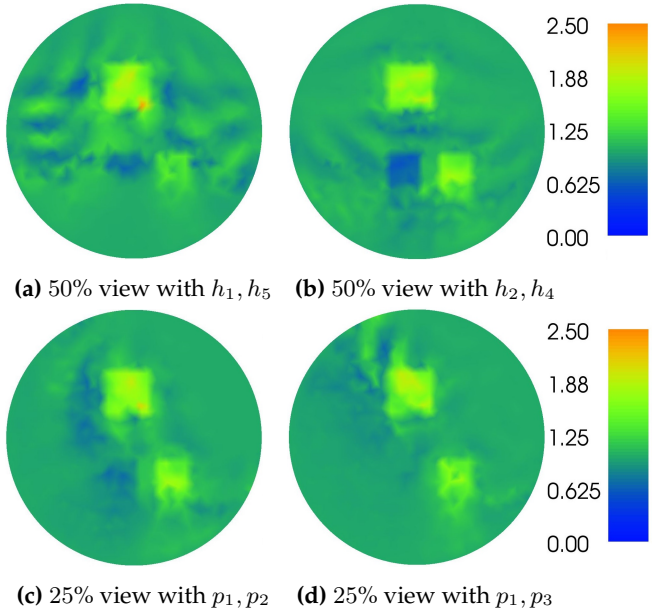
Again we try to use TSVD to see if we can get better solutions for the limited-view case. (We omit doing it for the original and noisy case seeing as it is limited what can be gained from this)

Solution for 50% view

Singular values = 0	Norm difference, h_1, h_5	Norm difference, h_2, h_4
50	0.1729	0.1028
100	0.1497	0.0871
150	0.1469	0.0849
175	0.1440	**0.0844**
200	0.1424	0.0860
250	0.1294	–
300	**0.1254**	–
350	0.1316	–

Solution for 25% view		
Singular values = 0	**Norm difference,** p_1, p_2	**Norm difference,** p_1, p_3
300	0.1064	0.1230
350	0.0952	0.1200
375	0.0931	0.1178
400	**0.0935**	0.1111
450	0.0971	**0.1089**
500	–	0.1096

The best numerical solutions can be seen in Figure 6.8

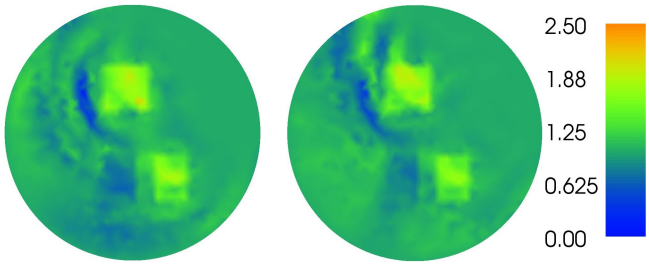


(a) 50% view with h_1, h_5 **(b)** 50% view with h_2, h_4

(c) 25% view with p_1, p_2 **(d)** 25% view with p_1, p_3

Figure 6.8: Best TSVD solutions for limited-view

It is clear that when we only have access to one quarter of the boundary, elements hidden behind other elements relative to the available boundary gets hard to recover. However if we allow more artefacts/noise to appear in the reconstruction by allowing more singular values it is possible to actually see the area, see Figure 6.9. The problem is just that there is a lot of artefacts in the image, thus if we had no idea of the real conductivity, we might mistake some areas to be part of the true conductivity, when they are actually not.



(a) 25% view with p_1, p_2, -250 singular values **(b)** 25% view with p_1, p_3, -300 singular values

Figure 6.9: Other TSVD solutions for limited-view

In accordance with the observed singular vectors earlier it also makes sense that when we exclude the last singular vectors we excluded that ones that dominate the areas far away from the measured area and thus we end up with an area that gets more and more smooth.

6.4 Simple discontinuity

For this one we restrict our attention to the 25% case. Here we have the condition numbers

Condition number for 25% view	
p_1, p_2	19961.7
p_1, p_3	76297.6

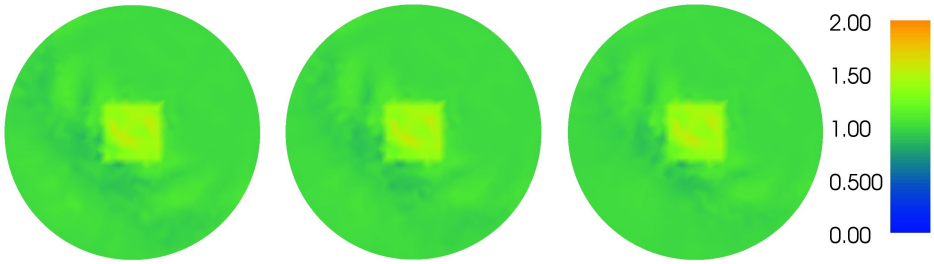
which again is of the same magnitude like the last examples so it fits with the statement that there shouldn't be much of many difference.

6.4.1 TSVD

Again we try to do TSVD though only for the boundary combination p_1, p_2

Solution for 25% view	
Singular values = 0	**Norm difference,** p_1, p_2
250	0.0443
275	**0.0431**
300	0.0432
350	0.0469

A couple of the solutions can be seen in Figure 6.10



(a) -250 singular values **(b)** -275 singular values **(c)** -300 singular values

Figure 6.10: TSVD solutions for 25% view, p_1, p_2

It is clear that like for the mollifier we can get a quite good reconstruction since we don't have elements of the conductivity that is located down in the worst part of the solution. Also it is nice to see that we are actually able to construct a nice square shape even though we only measure one quarter of the boundary.

6.5 Conclusion to SVD

It is clear, that when we cut off the small singular values, we can get rid of most of the artefacts and get solutions that are a lot better.

For the ones with full-view we are able to get a better solution, however it doesn't change much.

As we do regularization by TSVD for the limited-view case we get rid of the very little information there is of the areas far away from the measured area, which results in some solution, where elements, that should be there, disappear. However we can not use the

solutions with more singular values, since here we get a lot of elements that shouldn't be there and without prior knowledge it is hard to tell, which are right, and which are wrong areas.

It is clear that the higher the condition number is, the more singular values we need to cut-off to get the best numerical solution.

As for the singular vectors it is clear, that the last ones are the ones, that carry the information resulting in the artefacts, and for the limited-view the ones, that carry information about the problematic areas, which explains why we get more smooth solutions, when we cut these off.

CHAPTER 7

Discussion and Conclusion

Through the use of Fréchet derivatives it is clear, that is possible to linearise the model of AET to get an approximation of the true solution. Looking at the solutions for the different phantoms in Chapter 4 it is clear, that we are actually able to reconstruct the conductivity rather well using 2 or more boundary conditions. However we see these strange artefacts appearing, which is not consistent with what has already been seen in [8], as Hoffmann does not have these artefacts in his solutions. It is clear after looking at the singular value decomposition of the system, that it is the matrix $\mathcal{A}$, that induces these artefacts as the same lines are apparent in some of the last singular vectors.

It is hard to tell, what exactly induces this structure in the matrix $\mathcal{A}$. One theory could be that the fact, that we decide to decouple our system, is to blame for the artefacts, seeing as this is one thing that have been done in this thesis compared to [8]. In [8] the full linear system is implemented and not the decoupled like here. Another thing is that [8] also uses some iterative methods to reconstruct. However the method of reconstruction shouldn't be that different seeing, as it is the matrix, that seem to induce the structure.

Another peculiar thing is that we keep getting the best results for the reconstructions, when we use 2 boundary conditions, that according to [3] and [8] should not induce an elliptic system and thus is should not necessarily be stable nor have uniqueness.

However we still see that both with and without noise, these are the best boundary conditions. This again seems to be due to this strange structure, that the matrix induces, since the 2 "wrong" conditions does not have any sort of specific structure it seems. The matrices for all combinations of boundary conditions are all more or less equally behaving, when looking at the condition numbers for the full-view case, so one could say that the singular vectors in some sense are weighed the same in all cases however in f_1, f_2 and f_2, f_3, there is artefacts in the last singular vectors, whereas in the good one, f_1, f_3, the singular vectors does not have a specific pattern.

One thing that we also noted, when doing reconstructions is, it seems, that the artefacts in the reconstruction depend on the intensity of the conductivity, as they appear more heavily in the ones with high intensity and are almost gone, when it is low.

To try and get rid of some of the artefacts we tried doing regularization, but it was clear that the reconstructions doesn't get much better than before, and when we did over regularization to get rid of the artefacts, it was clear, that we then made the rest of the reconstruction worse. This makes sense seeing as we have seen that the matrix is not that ill-posed, so there is not much to be gained when doing regularization.

When we turn our attention to the limited-view case it is clear, that at the areas furthest away from where we measure we get more unreliable reconstructions. Using regularization it has been possible to get rid at some of the disturbing artefacts by removing some of the last singular vectors. However this has been at the cost of whatever little information there was about these areas. This is due to the fact, that the last singular vectors are the ones containing the information about these areas. So getting rid of these to remove the error, we end up getting rid of everything, there was to say about this area.

One thing to note though about the limited-view is, that if we look at the case of an element near the boundary it is clear that we are actually able to see it even though it is far away from the measured boundary, as long as there is not some elements obscuring the path from the measured boundary.

7.1 Future work

One very interesting thing to investigate would be why the matrix induces the artefacts, we see in our reconstructions and also why we don't see any, when we use cosine and sine as boundary conditions. In relation to this one should maybe look in to how the stability and uniqueness look for the decoupled system compared to the original system in the numerical implementation.
Another thing would be to figure out how noise in the measurements would effect the interior data to make reliable noise experiments.
Also it would be interesting to see, if it would be possible to do limited-view reconstruction that maybe with a better choice for boundary conditions or something else would yield better reconstructions for the areas far away from the measured boundaries.
One could also investigate, if there would be other numerical methods than just using least-squares, that would be able to yield better results with the decoupled system. An iterative method like in [8] could be a idea.
Lastly it would also be interesting to see if other reference conductivities than the one we choose, $\sigma = 1$, would result in better or worse reconstructions.

APPENDIX A

Lax-Milgram

Theorem A.1 *(Lax-Milgram Theorem)*
I assume that $B : H \times H \to \mathbb{R}$ is a bilinear mapping for which there exists constant $\alpha, \beta > 0$ such that

$$|B(u,v)| \leq \alpha \|u\| \|v\| \ \ (u,v \in H)$$

and

$$\beta \|u\|^2 \geq B(u,u) \ \ (u \in H).$$

Finally let $f : H \to \mathbb{R}$ be a bounded linear functional on H.
Then there exists a unique element $u \in H$ such that

$$B(u,v) = f(v), \ \ \forall v \in H.$$

Theorem A.2 *Let $\sigma_A \in L^\infty_+(\Omega)$, $\sigma_B \in L^\infty(\Omega)$ and $F \in H^1(\Omega)$ then there exists a unique weak solution $u \in H^1_0(\Omega)$ to*

$$\begin{cases} \nabla \cdot \sigma_A \nabla u &= -\nabla \cdot \sigma_B \nabla F \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega \end{cases}, \tag{A.1}$$

and

$$\|u\|_{H^1(\Omega)} \leq C \frac{\|\sigma_B\|_{L^\infty(\Omega)}}{K_{\sigma_A}} \|F\|_{H^1(\Omega)}, \tag{A.2}$$

where the constant C is independent of σ_A and σ_B and K_{σ_A} is the essential infimum of σ_A in accordance with Definition 1.1.

PROOF. In accordance with Theorem A.1 we define

$$B(u,v) = \int_\Omega \sigma_A \nabla u \cdot \nabla v dx$$

and

$$\bar{F}(v) = -\int_\Omega \sigma_B \nabla F \cdot \nabla v dx$$

from the weak formulation of (A.1).
Clearly B is bilinear and $\bar{F}$ is linear.
We need to check the existence of the constant $\alpha, \beta > 0$

$$\begin{aligned} |B(u,v)| &= \left| \int_\Omega \sigma_A \nabla u \cdot \nabla v dx \right| \\ &\leq \int_\Omega |\sigma_A \nabla u \cdot \nabla v| \, dx \\ &\leq \|\sigma_A\|_{L^\infty(\Omega)} \|\nabla u\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)} \\ &\leq \|\sigma_A\|_{L^\infty(\Omega)} \|u\|_{H_0^1(\Omega)} \|v\|_{H_0^1(\Omega)}. \end{aligned}$$

By Hölder inequality and theorem of absolutely integrable functions we get

$$\begin{aligned} B(u,u) &= \int_\Omega \sigma_A \nabla u \cdot \nabla u dx \\ &= \int_\Omega \sigma_A |\nabla u|^2 \, dx \\ &\geq K_{\sigma_A} \|\nabla u\|^2_{L^2(\Omega)} \\ &\geq K_{\sigma_A} C \|u\|^2_{H_0^1(\Omega)} \end{aligned}$$

for some positive constant C independent of σ_A.
$\|u\|_{H_0^1(\Omega)} \leq c\|\nabla u\|_{L^2(\Omega)}$ by Poincaré inequality [6, Theorem 3, Section 5.6.1] since

$$\begin{gathered} \|u\|_{L^2(\Omega)} \leq c\|\nabla u\|_{L^2(\Omega)} \\ \Downarrow \\ \|u\|_{H_0^1(\Omega)} = \|u\|_{L^2(\Omega)} + \|\nabla u\|_{L^2(\Omega)} \leq c\|\nabla u\|_{L^2(\Omega)} + \|\nabla u\|_{L^2(\Omega)} = \tilde{c}\|\nabla u\|_{L^2(\Omega)}. \end{gathered}$$

Lastly one needs to show that $\bar{F}$ is bounded

$$\begin{aligned}
|\bar{F}(v)| &= \left| \int_\Omega \sigma_B \nabla F \cdot \nabla v dx \right| \\
&\leq \int_\Omega |\sigma_B \nabla F \cdot \nabla v| \, dx \\
&\leq \|\sigma_B\|_{L^\infty(\Omega)} \int_\Omega |\nabla F \cdot \nabla v| \, dx \\
&\leq \|\sigma_B\|_{L^\infty(\Omega)} \|\nabla F\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)} \\
&\leq \|\sigma_B\|_{L^\infty(\Omega)} \|F\|_{H^1(\Omega)} \|v\|_{H_0^1(\Omega)}.
\end{aligned}$$

Thus by Theorem A.1 there exists a unique weak solution $u \in H_0^1(\Omega)$ to (A.1), since

$$B(u, v) = \bar{F}(v).$$

Now taking absolute value and choosing u as the test function we have that

$$K_{\sigma_A} C \|u\|^2_{H_0^1(\Omega)} \leq |B(u, u)| = |\bar{F}(u)| \leq \|\sigma_B\|_{L^\infty(\Omega)} \|F\|_{H^1(\Omega)} \|u\|_{H_0^1(\Omega)}$$

i.e.

$$\|u\|_{H_0^1(\Omega)} \leq \tilde{C} \frac{\|\sigma_B\|_{L^\infty(\Omega)}}{K_{\sigma_A}} \|F\|_{H^1(\Omega)},$$

where the constant $\tilde{C}$ is independent of σ_A and σ_B.

□

Remark *Note that also*

$$\|u\|_{H_0^1(\Omega)} \leq C \frac{\|\sigma_B\|_{L^\infty(\Omega)}}{K_{\sigma_A}} \|\nabla F\|_{L^2(\Omega)},$$

for some constant C independent of σ_A and σ_B.

The boundary value problem from (1.1)

$$\begin{cases} \nabla \cdot \sigma \nabla u &= 0 \quad \text{in } \Omega \\ u &= f \quad \text{on } \partial\Omega \end{cases}$$

for $\sigma \in L^\infty_+(\Omega)$ and $f \in H^{1/2}(\partial\Omega)$ can be transformed to a problem with homogeneous Dirichlet boundary conditions.
We do this by introducing the function $F \in H^1(\Omega)$, where $TF = f$. Here TF is the

trace of F to the boundary. We then define $\bar{u} = u - F$, which gives us a boundary value problem of the form

$$\begin{cases} \nabla \cdot \sigma \nabla \bar{u} &= -\nabla \cdot \sigma \nabla F \quad \text{in } \Omega \\ \bar{u} &= 0 \qquad\qquad\quad \text{on } \partial\Omega \end{cases}. \tag{A.3}$$

Looking at the solution $u = \bar{u} + F$ for the original problem (1.1) and using (A.2) on (A.3) we get

$$\begin{aligned} \|u\|_{H_0^1(\Omega)} &\leq \|\bar{u}\|_{H_0^1(\Omega)} + \|F\|_{H^1(\Omega)} \\ &\leq \left(\tilde{C} \frac{\|\sigma\|_{L^\infty(\Omega)}}{K_\sigma} + 1 \right) \|F\|_{H^1(\Omega)}. \end{aligned}$$

By [5, Proposition 3.32] we can choose $F \in H^1(\Omega)$ as an extension of $f \in H^{1/2}(\partial\Omega)$ such that $\|F\|_{H^1(\Omega)} \leq c(\Omega)\|f\|_{H^{1/2}(\partial\Omega)}$, where the constant only depends on the domain. Thus we get that

$$\|u\|_{H_0^1(\Omega)} \leq c(\Omega) \left(\tilde{C} \frac{\|\sigma\|_{L^\infty(\Omega)}}{K_\sigma} + 1 \right) \|f\|_{H^{1/2}(\partial\Omega)}. \tag{A.4}$$

By use of the bounds of σ from Definition 1.1 we get that

$$\|u\|_{H_0^1(\Omega)} \leq c(\Omega) \left(\tilde{C} \frac{M_\sigma}{K_\sigma} + 1 \right) \|f\|_{H^{1/2}(\Omega)}.$$

Appendix B

FEniCS/Python code

B.1 Mesh Generation

B.1.1 Simple conductivity and small contrast

```
from dolfin import *
from mshr import *

domain = Circle(Point(0, 0), 1, 150)

domain.set_subdomain(1, Rectangle(Point(-0.2,-0.2), Point(0.2, 0.2)))

N = 20

mesh = generate_mesh(domain, N)
```

B.1.2 Near boundary 1

```
from dolfin import *
from mshr import *
```

```
domain = Circle(Point(0, 0), 1, 150)

domain.set_subdomain(1, Rectangle(Point(0.6,-0.1), Point(0.9, 0.1)))

N = 20

mesh = generate_mesh(domain, N)
```

B.1.3 Near boundary 2

```
from dolfin import *
from mshr import *

domain = Circle(Point(0, 0), 1, 150)

domain.set_subdomain(1, Rectangle(Point(-0.1,0.6), Point(0.1, 0.9)))

N = 20

mesh = generate_mesh(domain, N)
```

B.1.4 Three discontinuities

```
from dolfin import *
from mshr import *

domain = Circle(Point(0, 0), 1, 150)

domain.set_subdomain(1, Rectangle(Point(-0.2,0.2), Point(0.1, 0.5)))
domain.set_subdomain(2, Rectangle(Point(0.2,-0.5), Point(0.4, -0.2)))
domain.set_subdomain(3, Rectangle(Point(-0.2,-0.5), Point(0, -0.2)))

N = 20

mesh = generate_mesh(domain, N)
```

B.2 Definition of Conductivity

B.2.1 Simple conductivity

```
a0 = Constant(1.0)
a1 = Constant(1.5)

class sigmafun(Expression):
       def eval(self,values,x):
                if (between(x[1], (-0.2,0.2)) and
                between(x[0], (-0.2, 0.2))):
                        values[0] = a1
                else:
                        values[0] = a0

sigma = sigmafun()
```

B.2.2 Mollifier

```
  class sigmafun(Expression):
       def eval(self,values,x):
                if (x[0]**2+x[1]**2<0.5):
                        values[0] = 1+3*math.exp(1/(x[0]**2+x[1]**2-0.5))
                else:
                        values[0] = 1

  sigma = sigmafun()
```

B.2.3 Small contrast

```
  a0 = Constant(1.0)
  a1 = Constant(1.1)

  class sigmafun(Expression):
       def eval(self,values,x):
                if (between(x[1], (-0.2,0.2)) and
                between(x[0], (-0.2, 0.2))):
                        values[0] = a1
                else:
                        values[0] = a0

  sigma = sigmafun()
```

B.2.4 Near boundary 1

```
a0 = Constant(1.0)
a1 = Constant(1.5)

class sigmafun(Expression):
        def eval(self,values,x):
                if (between(x[1], (-0.1,0.1)) and
                between(x[0], (0.6, 0.9))):
                        values[0] = a1
                else:
                        values[0] = a0

sigma = sigmafun()
```

B.2.5 Near boundary 2

```
a0 = Constant(1.0)
a1 = Constant(1.5)

class sigmafun(Expression):
        def eval(self,values,x):
                if (between(x[1], (0.6,0.9)) and
                between(x[0], (-0.1, 0.1))):
                        values[0] = a1
                else:
                        values[0] = a0

sigma = sigmafun()
```

B.2.6 Three discontinuities

```
 a0 = Constant(1.0)
 a1 = Constant(1.9)
 a2 = Constant(1.6)
 a3 = Constant(0.7)
 class sigmafun(Expression):
        def eval(self,values,x):
                if (between(x[1], (0.2,0.5)) and
                between(x[0], (-0.2, 0.1))):
                        values[0] = a1
                elif (between(x[1], (-0.5,-0.2)) and
                between(x[0], (0.2, 0.4))):
                        values[0] = a2
                elif (between(x[1], (-0.5,-0.2)) and
```

```
                between(x[0], (-0.2, 0))):
                        values[0] = a3
                else:
                        values[0] = a0

sigma = sigmafun()
```

B.3 Computation of Data

B.3.1 Gradient of Green's function, ∇G

```
cord = mesh.coordinates()
K = mesh.num_vertices()
NabGarray = np.zeros((K,K,2))

def xx0absgrad(x,x0):
    y = (x-x0)
    return y/(y[0]**2+y[1]**2)

def xx0abs(x,x0):
    y = (x-x0)
    return (y[0]**2+y[1]**2)

for i in range(K):
    x1 = cord[i]
    for j in range(K):
        if (j<>i):
            x2 = cord[j]
            if (x2[0]**2+x2[1]**2==0):
                NabGarray[i,j,:] = (1/(2*math.pi)*
                x1/(x1[0]**2+x1[1]**2))
            else:
                x2star =x2/(x2[0]**2+x2[1]**2)
                NabGarray[i,j,:] = (1/(2*math.pi)*
                (xx0absgrad(x1,x2)-xx0absgrad(x1,x2star)))
```

B.3.2 Setting up the system

Here it should be noted that the previous code for computing ∇G should be included in the code for it to run properly

```
from dolfin import *
from mshr import *
import numpy as np
import shelve
import math

# Mesh generated earlier and saved
mesh = Mesh('mitmesh.xml')

# Conductivity
a0 = Constant(1.0)
a1 = Constant(1.5)

class sigmafun(Expression):
        def eval(self,values,x):
                if (between(x[1], (-0.2,0.2)) and
                between(x[0], (-0.2, 0.2))):
                        values[0] = a1
                else:
                        values[0] = a0

sigma = sigmafun()

V = FunctionSpace(mesh,"CG",1)

u = TrialFunction(V)
v = TestFunction(V)

# lhs of the weak formulation
a = sigma*inner(grad(u),grad(v))*dx
aref = inner(grad(u),grad(v))*dx #reference sigma = 1

# rhs of weak formulation
uleft = Constant("0.00")
L = uleft*v*dx

# Boundary condition
f = Expression("x[0]") #cos(theta)
bcs = DirichletBC(V,f,"on_boundary")

u = Function(V)
uref = Function(V)

# Solve weak formulation
solve(a == L,u,bcs)
solve(aref == L,uref,bcs)
```

```
# Interior data
H = sigma*inner(grad(u),grad(u))
Href = inner(grad(uref),grad(uref))

# For handling gradient, vector space
parameters.reorder_dofs_serial = False
Vvec = VectorFunctionSpace(mesh,"CG",1)
dof0 = Vvec.sub(0).dofmap().dofs()
dof1 = Vvec.sub(1).dofmap().dofs()

# Function for computing M
def Gint(dsigma):
    I = Function(V)
    NabG = Function(Vvec)
    Iarray = np.zeros((K))
    for i in range(K):
        nabarray = NabG.vector().array()
        nabarray[dof0] = NabGarray[:,i,0]
        nabarray[dof1] = NabGarray[:,i,1]
        NabG.vector()[:] = nabarray
        Iarray[i] = assemble(inner(NabG,dsigma*grad(uref))*dx)
    I.vector()[:] = Iarray
    return I

A = np.zeros((K,K))
b = np.zeros((K))

# Compute A and b
for i in range(K):
    ibasis = Function(V)
    ibasis.vector()[i] = 1
    delu = Gint(ibasis)
    for j in range(K):
        jbasis = Function(V)
        jbasis.vector()[j] = 1
        A[j,i] = (assemble((ibasis*inner(grad(uref),grad(uref))+
        2*inner(grad(uref),grad(delu)))*jbasis*dx))
    b[i] = assemble(inner(H-Href,ibasis)*dx)


# Save data for later use
data = shelve.open("decoup.db")

data["A_system"] = A
data["b_system"] = b

data.close()
```

B.3.3 Check of Green's solution

```
from dolfin import *
from mshr import *
import numpy as np
import math

parameters.reorder_dofs_serial = False

# Mesh
domain = Circle(Point(0, 0), 1, 150)

N = 20

mesh = generate_mesh(domain, N)

V = FunctionSpace(mesh,"CG",1)

# Conductivity
class sigmafun(Expression):
        def eval(self,values,x):
                if (x[0]**2+x[1]**2<0.5):
                        values[0] = 3*math.exp(1/(x[0]**2+x[1]**2-0.5))
                else:
                        values[0] = 0

sigma = sigmafun()


u0 = Constant("0.00")

cord = mesh.coordinates()
K = mesh.num_vertices()

# Compution af gradient
NabGarray = np.zeros((K,K,2))

def xx0absgrad(x,x0):
    y = (x-x0)
    return y/(y[0]**2+y[1]**2)
def xx0abs(x,x0):
    y = (x-x0)
    return (y[0]**2+y[1]**2)

for i in range(K):
     x1 = cord[i]
     for j in range(K):
         if (j<>i):
```

```
                x2 = cord[j]
                if (x2[0]**2+x2[1]**2==0):
                    NabGarray[i,j,:] = (1/(2*math.pi)*
                    x1/(x1[0]**2+x1[1]**2))
                else:
                    x2star =x2/(x2[0]**2+x2[1]**2)
                    NabGarray[i,j,:] = (1/(2*math.pi)*
                    (xx0absgrad(x1,x2)-xx0absgrad(x1,x2star)))

Vvec = VectorFunctionSpace(mesh,"CG",1)
dof0 = Vvec.sub(0).dofmap().dofs()
dof1 = Vvec.sub(1).dofmap().dofs()

# Reference potential
f = Expression("x[0]") #cos(theta)

u = TrialFunction(V)
v = TestFunction(V)

bcs = DirichletBC(V,f,"on_boundary")
aref = inner(grad(u),grad(v))*dx #reference sigma = 1
L = u0*v*dx

uref = Function(V)
solve(aref == L,uref,bcs)

# Computation of M
def Gint(dsigma):
    I = Function(V)
    NabG = Function(Vvec)
    Iarray = np.zeros((K))
    for i in range(K):
        nabarray = NabG.vector().array()
        nabarray[dof0] = NabGarray[:,i,0]
        nabarray[dof1] = NabGarray[:,i,1]
        NabG.vector()[:] = nabarray
        Iarray[i] = assemble(inner(NabG,dsigma*grad(uref))*dx)
    I.vector()[:] = Iarray
    return I

# Green's solution
u2 = Gint(sigma)

u = TrialFunction(V)
v = TestFunction(V)

# True solution to system
```

```
acheck = inner(grad(u),grad(v))*dx
Lcheck = -inner(sigma*grad(uref),grad(v))*dx
bcscheck = DirichletBC(V,u0,"on_boundary")

ucheck = Function(V)

solve(acheck == Lcheck,ucheck,bcscheck)

# Norm difference
print norm(u2.vector()-ucheck.vector())/norm(ucheck.vector())

# Plot
h=plot(u2,scale=0.0)
h.set_min_max(-0.05, 0.05)
h.elevate(65)
h.update(u2)

k=plot(ucheck,scale=0.0)
k.set_min_max(-0.05, 0.05)
k.elevate(65)
k.update(ucheck)

interactive()
```

B.4 Noisy measurements

```
H = sigma*inner(grad(u),grad(u))

H2 = project(H,V)

maxH = H2.vector().array().max()

minH = H2.vector().array().min()

noise = level*(maxH-minH)*np.random.normal(0,1,K)

Hnoise = Function(V)

Hnoise.vector()[:] = H2.vector().array() + noise
```

B.5 Partial boundary

B.5.1 Half of the boundary

```
class Boundary1(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and x[1] < 0.0

class Boundary2(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and not (x[1] < 0.0)

boundary1 = Boundary1()
boundary2 = Boundary2()
```

B.5.2 One quarter of the boundary

```
class Boundary1(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and not (x[1] > 0.0 and x[0] > 0.0 )

class Boundary2(SubDomain):
    def inside(self, x, on_boundary):
        return on_boundary and (x[1] > 0.0 and x[0] > 0.0 )

boundary1 = Boundary1()
boundary2 = Boundary2()
```

B.5.3 Change to the boundary condition in the code

```
bcs = [DirichletBC(V, Constant(0.0), boundary1),
       DirichletBC(V, f, boundary2)]
```

B.6 TSVD

```
N = mesh.num_vertices()

# Doing SVD
Us,Ss,Vs = np.linalg.svd(Abig)

# Construction inverse of diagonal matrix
```

```
Sinv = np.zeros((N,N*2))
K = N - 50 # Number of included singular values
Sinv[:K,:K] = np.diag(1.00/Ss[:K])

# Construction pseudo-inverse
AinvSVD= np.dot(np.dot(Vs.T,Sinv),Us.T)

# Construct solution
u1 = Function(V)
u1.vector()[:] = np.inner(AinvSVD,bbig)+1
```

Bibliography

[1] Fenics project. `http://http://fenicsproject.org/`. Accessed June 11, 2015.

[2] G. Bal. Hybrid inverse problems and internal functionals. *Inverse Problems and Applications: Inside Out II*, 60:325–368, 2011.

[3] G. Bal. Hybrid inverse problems and redundant systems of partial differential equations. page 33, 2013.

[4] G. Bal, E. Bonnetier, F. Monard, and F. Triki. Inverse diffusion from knowledge of power densities. page 24, 2012.

[5] D. Cioranescu and P. Donato. *An introduction to homogenization*. Oxford University Press, 1999.

[6] L. C. Evans. *Partial differential equations*. American Mathematical Society, 2008.

[7] P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring images : Matrices, spectra, and filtering*. SIAM, 2006.

[8] K. Hoffmann. *Reconstruction Methods for Inverse Problems with Partial Data*. PhD thesis, Technical University of Denmark, 2014.

[9] I. Kocyigit. Acousto-electric tomography and CGO solutions with internal data. *Inverse Problems*, 28(12):1, 2012.

[10] P. Kuchment and L. Kunyansky. Synthetic focusing in ultrasound modulated tomography. *Inverse Problems and Imaging*, 4(4):665–673, 2010.

[11] W. Strauss. *Partial differential equations : an introduction*. John Wiley & Sons, 2008.