Depth Dependency in Electrical Impedance Tomography with the Complete Electrode Model

Technical University of Denmark

Anders Eltved, s123875 &
Nikolaj Vestbjerg Christensen, s123631

Supervisors:

Kim Knudsen & Henrik Garde

June 19, 2015

Abstract

This project investigates depth dependencies in Electrical Impedance Tomography (EIT) with the Complete Electrode Model (CEM). It explores the relationship between the placement of an object in a domain and the quality of the reconstruction of it.

In this project the necessary theory to understand the reconstruction process of EIT is presented. This theory includes proof of existence and uniqueness of a solution with the CEM. Furthermore, additional theory on Singular Value Decomposition and Fréchet derivatives is presented, before using simulations on a variety of setups to obtain results.

The result for the depth dependency in EIT with the CEM is that the object's distance to the electrodes has a significant influence on the quality of the reconstruction. As expected the quality of the reconstruction is improved the closer the object is to the electrodes. Thus, the setup, i.e. the placement of the electrodes, is essential when searching for an object.

Abstract (Danish)

I dette projekt undersøges dybdeafhængighed i elektrisk impedanstomografi (EIT) med den fuldstændige elektrodemodel (CEM). Vi udforsker forholdet mellem placeringen af et objekt i et domæne og kvaliteten af en rekonstruktion af objektet.

Vi præsenterer den nødvendige teori for at forstå processen, der bruges, til rekonstruktion i EIT. Dette omfatter bevis for eksistens og entydighed af en løsning med CEM. Derudover præsenterer vi teori for singulær-værdi dekomposition og Fréchet afledte, før vi bruger simuleringer på forskellige setups til at opnå resultater.

Resultatet for dybdeafhængigheden af EIT med CEM er, at et objekts afstand til elektroderne har stor indflydelse på kvaliteten af en rekonstruktion. Som forventet øges kvaliteten i rekonstruktionen, når et objekt flyttes tættere på elektroderne. Dvs. opsætningen, altså placeringen af elektroder, er essentiel, når man søger efter et objekt.

Acknowledgements

We want to thank our supervisors Kim Knudsen and Henrik Garde for guidance and help throughout this project and for taking their time to meet every week. We want to thank Henrik for quick responses, for providing some needed theory and for help with the FEniCS library.

Contents

Abstract **ii**

Abstract (Danish) **iii**

Acknowledgements **iv**

1 Introduction **1**

2 The Complete Electrode Model **3**
2.1 Setting Up the Model 3
2.2 Existence and Uniqueness 7
2.3 Solving the CEM for a Simple Geometry 15

3 Linearization **18**
3.1 The Current-to-Voltage Map 18
3.2 The Fréchet Derivative 19
3.3 Calculating the Fréchet Derivative 24

4 Singular Value Decomposition and Reconstruction **26**
4.1 Singular Value Decomposition 26
4.2 Reconstruction of a Perturbation Function h 27
4.2.1 Adding Noise 27

5 Numerical Analysis of the CEM **29**
5.1 Setup for Numerical Analysis 29
5.2 Test of the Fréchet Derivative 29
5.3 Singular Vectors and Depth Dependency 31
5.4 Depth Dependency in Reconstruction 34
5.5 Testing Reconstructions with Noise 37

6 Conclusion **40**
6.1 Future Work 40

A Definitions, Theorems and Lemmas **41**
A.1 Sobolev Imbeddings for Bounded Domains 41
A.2 The Fréchet Derivative 42

B Analytical Solution to CEM **44**

C Plots **47**
C.1 Singular Vectors 48
C.2 Reconstructions 50

D Code **52**
D.1 CEMLibrary_full . . . 52
D.1.1 solver(sigma,L,I,Z,mesh) . . . 52
D.1.2 cont_plot(x,y,z) and cont_plot2(x,y,z) . . . 52
D.1.3 load_frechet() . . . 52
D.1.4 grid(x, y, z, resX=100, resY=100) . . . 52
D.1.5 gramSchmidt(L) . . . 52
D.1.6 create_R_sigma(sigma,L,Z,mesh) . . . 53
D.1.7 Frechet(sigma,L,Z,mesh) . . . 53
D.1.8 Frechet_point(sigma,L,Z,mesh) . . . 53
D.1.9 OperatorNorm(A) . . . 53
D.1.10 CEMLibrary_full.py . . . 53
D.2 h_functions . . . 57
D.3 reconstruct_moved_object . . . 59
D.4 R_sigma . . . 62
D.5 squaredomainCEM . . . 65
D.6 SVD . . . 66
D.7 SVD_reconstruct_h . . . 67
D.8 SVD_reconstruct_h_noise . . . 69

Bibliography **72**

Chapter 1

Introduction

Electrical Impedance Tomography (EIT) is the name of the technique that recovers the interior conductivity of an object by sending current trough the surface and measuring the responsive voltages. More specifically, the current is sent trough electrodes attached to the surface of the object. EIT has many practical applications (and different names in different fields). Some of these are medical EIT [2] and search for irregularities in materials [6]. From an economical point of view this technology is very interesting, since electrodes are cheap and easy to transport compared to other medical imaging techniques.

The Complete Electrode Model (CEM) assumes that current is sent trough a finite number of electrodes that does not cover the full surface in contrast to The Continuum Model that assumes the whole surface is covered completely by electrodes. In real-world problems the the CEM will be more relevant than the Continuum Model, because it is troublesome and expensive to put electrodes around the whole domain. The practical aspects of the CEM are the reasons that we have chosen to investigate this model.

The PDE that governs EIT is

$$\nabla \cdot \sigma \nabla u = 0,$$

where σ is the conductivity and u is the electric potential of the domain, which we denote the potential. The difference between the Continuum Model and the CEM is in the boundary conditions. Two important problems are discussed concerning the CEM - the forward and the backward problem. A correspondence between currents and voltages measured on the electrode are given by a map R_σ that takes in currents and maps the corresponding voltages. The forward problem is therefore to find the mapping R_σ given σ. The backward problem or the inverse problem is to find the conductivity, σ, given knowledge of the currents and voltages on the electrodes, i.e. R_σ.

The purpose of this project is to study how EIT with the CEM can be used to reconstruct small perturbations in the conductivity. Furthermore, the purpose is to investigate if there is any depth dependency in EIT with the CEM. By depth dependency we mean if the model reconstructs perturbations better near the electrodes compared to deeper in the domain, i.e. if the distance from the electrodes to the object has an impact on the reconstruction.

In order to accomplish this, we give a proof of existence and uniqueness of solutions in EIT with the CEM in Chapter 2 followed by a simple example. Since the inverse problem is non-linear, we linearize it with the help of the Fréchet derivative in Chapter 3. We will denote the linearized inverse problem around σ as the inverse problem for simplification. We show how to solve the inverse problem, for small perturbations h

in Chapter 4. This requires basic knowledge of Singular Value Decomposition (SVD), which is also introduced in this chapter. In Chapter 5 we do numerical analysis in two dimensions where the focus is primarily to investigate the depth dependency on the unit disc with three different electrode configurations.

Chapter 2

The Complete Electrode Model

In this chapter we introduce the Complete Electrode Model and prove existence and uniqueness of a solution in the model. The proves in this chapter are inspired by [8] and reworked in details. We will start off by going through the assumptions of the model.

2.1 Setting Up the Model

We let Ω be an open, bounded domain in two or three dimensions, $\mathbb{R}^n, n = 2, 3$, where the boundary $\partial\Omega$ is C^1. We let $L \in \mathbb{N}$ be the number of electrodes on the boundary. We assume that the electrodes, $e_l,\ l = 1, 2, \cdots, L$, are open connected subsets of $\partial\Omega$ whose closures are disjoint, i.e. $\bar{e}_k \cap \bar{e}_l = \emptyset, \quad k \neq l$. For the conductivity, σ, we assume that it is bounded, i.e. $\sigma \in L^\infty(\Omega)$. To account for the resistance between the electrodes and the surface, we introduce the contact impedance coefficients and denote them $z_l, l = 1, 2, \cdots, L$. We denote the current vectors $\boldsymbol{I}$ and the voltage vectors $\boldsymbol{U}$. We note that we will only consider current patterns that satisfy

$$\sum_{l=1}^{L} I_l = 0. \tag{2.1}$$

We introduce the space

$$H = H^1(\Omega) \times \mathbb{C}^L,$$

where

$$H^1(\Omega) = \left\{ f : \mathbb{R}^n \to \mathbb{C} \quad | \quad \int_\Omega \left(|f(x)|^2 + |\nabla f(x)|^2 \right) dx < \infty \right\},$$

is a Sobolov space, see Appendix A.1.

Proposition 1. *Let Ω, σ and the electrodes e_l satisfy the assumptions above, and let $z_l \in \mathbb{C},\ l = 1, 2, \cdots, L$, satisfy Re $z_l > 0$.*
Assume that $u \in H^1(\Omega)$, $U \in \mathbb{C}^L$ and that u is a weak solution to

$$\nabla \cdot \sigma \nabla u = 0 \quad \text{in } \Omega \tag{2.2}$$

subject to the boundary conditions on $\partial\Omega$

$$u + z_l \sigma \frac{\partial u}{\partial \nu} = U_l \quad \text{on } e_l,\ l = 1, 2, \cdots, L\ , \tag{2.3}$$

$$\sigma \frac{\partial u}{\partial \nu} = 0 \quad on\ \partial\Omega \backslash \overline{\bigcup_{l=1}^{L} e_l}. \tag{2.4}$$

Assume furthermore that for a prescribed current pattern $\boldsymbol{I} = (I_l)_{l=1}^{L} \in \mathbb{R}^L$,

$$\int_{e_l} \sigma \frac{\partial u}{\partial \nu}\ dS = I_l, \tag{2.5}$$

Then $(u, \boldsymbol{U}) \in H$ *satisfy*

$$B\left((u, \boldsymbol{U}), (v, \boldsymbol{V})\right) = \sum_{l=1}^{L} I_l \bar{V}_l, \tag{2.6}$$

where $B : H \times H \to \mathbb{C}$ *is the sesquilinear form defined as*

$$B\left((u, \boldsymbol{U}), (v, \boldsymbol{V})\right) = \int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v}\ dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l)\ dS,$$

for any $(v, \boldsymbol{V}) \in H$.

Conversely, if $(u, \boldsymbol{U}) \in H$ *satisfies* (2.6) *for all* $(v, \boldsymbol{V}) \in H$, *then* $(u, \boldsymbol{U})$ *is a weak solution to* (2.2)-(2.5).

Proof. Let us first assume that $(u, \boldsymbol{U}) \in H$ satisfies (2.2)-(2.5). We want to show that $(u, \boldsymbol{U})$ then satisfies (2.6) for any $(v, \boldsymbol{V}) \in H$.
We choose an arbitrary $(v, \boldsymbol{V}) \in H$. Taking equation (2.2) and multiplying with $\bar{v}$ and integrating over Ω we get

$$\int_{\Omega} \bar{v} \nabla \cdot \sigma \nabla u\ dx = 0.$$

By Green's first identity, this becomes

$$\int_{\Omega} \bar{v} \nabla \cdot \sigma \nabla u\ dx = -\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} dx + \int_{\partial\Omega} \sigma \frac{\partial u}{\partial \nu} \bar{v}\ dS = 0 \tag{2.7}$$

for all $v \in H^1(\Omega)$, where $\nabla \cdot \sigma \nabla u \in L^2(\Omega)$.
We can combine the assumptions (2.3) and (2.4) to

$$\sigma \frac{\partial u}{\partial \nu} = \sum_{l=1}^{L} \frac{1}{z_l} (U_l - u) \chi_l, \tag{2.8}$$

where χ_l is the characteristic function on electrode e_l. Inserting (2.8) into (2.7) yields

$$\begin{aligned} -\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} dx + \int_{\partial\Omega} \bar{v} \sum_{l=1}^{L} \frac{1}{z_l} (U_l - u) \chi_l\ dS = 0 \qquad &\Leftrightarrow \\ -\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (U_l - u) \bar{v}\ dS = 0 \qquad &(2.9) \end{aligned}$$

Using (2.3) and inserting the result from (2.5) we get

$$u = U_l - z_l \sigma \frac{\partial u}{\partial \nu} \qquad \Leftrightarrow$$

$$\int_{e_l} u \, dS = \int_{e_l} \left(U_l - z_l \sigma \frac{\partial u}{\partial \nu} \right) \, dS = U|e_l| - z_l I_l \qquad \Leftrightarrow$$

$$\int_{e_l} u \, dS - U|e_l| + z_l I_l = 0.$$

Using the last equality, we can multiply coefficients on the left hand side for all L electrodes and take the sum of them, and the sum will still be zero.

$$\sum_{l=1}^{L} \frac{1}{z_l} \bar{V}_l \left(\int_{e_l} u - U|e_l| + z_l I_l \right) = 0. \tag{2.10}$$

Adding (2.9) and (2.10) we get

$$-\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (U_l - u)\bar{v} \, dS + \sum_{l=1}^{L} \frac{1}{z_l} \bar{V}_l \left(\int_{e_l} u \, dS - U|e_l| + z_l I_l \right) = 0 \quad \Leftrightarrow$$

$$\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \left(\int_{e_l} (u - U_l)\bar{v} \, dS - \int_{e_l} u \bar{V}_l \, dS \; + \bar{V}_l U_l |e_l| \right) = \sum_{l=1}^{L} \bar{V}_l I_l \quad \Leftrightarrow$$

$$\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \left(\int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l) \, dS - U_l \bar{V}_l |e_l| + U_l \bar{V}_l |e_l| \right) = \sum_{l=1}^{L} \bar{V}_l I_l \quad \Leftrightarrow$$

$$\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l) \, dS = \sum_{l=1}^{L} \bar{V}_l I_l,$$

which is exactly (2.6).

For the converse part we assume that $(u, \boldsymbol{U}) \in H$ satisfies (2.6) for any $(v, \boldsymbol{V}) \in H$ and want to prove that $(u, \boldsymbol{U})$ also satisfies (2.2)-(2.5) with u as a weak solution. This is done by considering specific choices of $(v, \boldsymbol{V})$.
First choice is $v \in C_c^\infty(\Omega)$ and $\boldsymbol{V} = \boldsymbol{0}$. From (2.6) we see that

$$\int_{\Omega} \sigma \nabla u \cdot \nabla \bar{v} \, dx = 0.$$

Since $v \in H^1(\Omega)$ u satisfies (2.2) in the weak sense, i.e. $\nabla \cdot \sigma \nabla u = 0 \in L^2(\Omega)$. Therefore by (2.7) we have that

$$\int_\Omega \sigma \nabla u \cdot \nabla \bar{v}\ dx = \int_{\partial\Omega} \sigma \frac{\partial u}{\partial \nu} \bar{v}\ dS, \tag{2.11}$$

$v \in H^1(\Omega)$ arbitrary. Combining (2.11) with (2.6), choosing $\boldsymbol{V} = \boldsymbol{0}$ again, we get

$$\int_{\partial\Omega} \sigma \frac{\partial u}{\partial \nu} \bar{v}\ dS + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)\bar{v}\ dS = 0.$$

Using that

$$\int_{e_l} (u - U_l)\bar{v}\ dS = \int_{\partial\Omega} \chi_l (u - U_l)\bar{v}\ dS,$$

we have that

$$\int_{\partial\Omega} \left(\sigma \frac{\partial u}{\partial \nu} + \sum_{l=1}^{L} \frac{1}{z_l} \chi_l (u - U_l) \right) \bar{v}\ dS = 0. \tag{2.12}$$

Since v is arbitrary we see that

$$\sigma \frac{\partial u}{\partial \nu} + \sum_{l=1}^{L} \frac{1}{z_l} \chi_l (u - U_l) = 0 \quad \text{on } \partial\Omega.$$

That is (2.8), which means that (2.3) and (2.4) are satisfied by $(u, \boldsymbol{U})$.
Now let $\boldsymbol{V} \in \mathbb{C}^L$ be arbitrary. Inserting (2.11) in (2.6) we get

$$\begin{aligned}
&\sum_{l=1}^{L} I_l \bar{V}_l = \int_{\partial\Omega} \sigma \frac{\partial u}{\partial \nu} \bar{v}\ dS + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l) && \Leftrightarrow \\
&\sum_{l=1}^{L} I_l \bar{V}_l = \int_{\partial\Omega} \sigma \frac{\partial u}{\partial \nu} \bar{v}\ dS + \int_{\partial\Omega} \sum_{l=1}^{L} \frac{1}{z_l} \chi_l (u - U_l)(\bar{v} - \bar{V}_l) && \Leftrightarrow \\
&\sum_{l=1}^{L} I_l \bar{V}_l = \int_{\partial\Omega} \left(\sigma \frac{\partial u}{\partial \nu} + \sum_{l=1}^{L} \frac{1}{z_l} \chi_l (u - U_l) \right) \bar{v}\ dS - \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)\bar{V}_l\ dS.
\end{aligned}$$

We find that the first term on the right side is zero because of (2.12) so that we have

$$\begin{aligned}
\sum_{l=1}^{L} I_l \bar{V}_l &= -\sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)\bar{V}_l\ dS && \Leftrightarrow \\
0 &= \sum_{l=1}^{L} \bar{V}_l \left(I_l + \frac{1}{z_l} \int_{e_l} (u - U_l)\ dS \right).
\end{aligned}$$

Since $\boldsymbol{V} \in \mathbb{C}^L$ is arbitrary, we must have that

$$I_l + \frac{1}{z_l} \int_{e_l} (u - U_l)\ dS = 0, \quad l = 1, 2, \cdots, L\ ,$$

which is equivalent to

$$\frac{1}{z_l}\int_{e_l}(U_l-u)\,dS=I_l,\quad l=1,2,\cdots,L.$$

Using this with (2.3) we obtain (2.5) as

$$\int_{e_l}\sigma\frac{\partial u}{\partial\nu}\,dS=\frac{1}{z_l}\int_{e_l}(U_l-u)\,dS=I_l.$$

□

We have now shown that a solution to (2.6) also is a weak solution to the PDE problem (2.2)-(2.5).

Now we would like to show that the PDE problem (2.2)-(2.5) has a unique solution by showing that the weak formulation (2.6) has a unique solution.

2.2 Existence and Uniqueness

To obtain existence and uniqueness of a solution to our model we would like to use the Lax-Milgram theorem in Appendix A.1.

By setting

$$B\left((u,\boldsymbol{U}),(u,\boldsymbol{U})\right)=\int_\Omega\sigma\left|\nabla u\right|^2dx+\sum_{l=1}^{L}\frac{1}{z_l}\int_{e_l}\left|u-U_l\right|^2dS=0,$$

we see that our operator B does not satisfy the coercivity, since the above does not imply that $(u,\boldsymbol{U})=0$,

$$u=U_1=\cdots=U_L=\text{const.}$$

To avoid this problem, and still be able to use the Lax-Milgram theorem, we introduce the quotient space

$$\dot{H}=H/\mathbb{C},$$

where the elements $(u,\boldsymbol{U})\in H$ and $(v,\boldsymbol{V})\in H$ are equivalent if

$$u-v=U_1-V_1=\cdots=U_L-V_L=\text{const.}$$

The space $\dot{H}$ can be equipped with the quotient norm

$$\|(u,\boldsymbol{U})\|=\inf_{c\in\mathbb{C}}\left(\|u-c\|^2_{H^1(\Omega)}+\|U-C\|^2_{\mathbb{C}^L}\right)^{1/2},\tag{2.13}$$

but showing that our operator B satisfies the conditions of the Lax-Milgram theorem is much easier with a different norm in $\dot{H}$, namely

$$\|(u,\boldsymbol{U})\|_*=\left(\|\nabla u\|^2_{L^2(\Omega)}+\sum_{l=1}^{L}\int_{e_l}\left|u(x)-U_l\right|^2dS\right)^{1/2}.\tag{2.14}$$

In order to use this norm instead, we show that the two norms are equivalent.

Lemma 2. *The norms* (2.13) *and* (2.14) *are equivalent; i.e., for some constants* $0 < \lambda < \Lambda < \infty$,

$$\lambda \left\|(u,\boldsymbol{U})\right\|_* \leq \left\|(u,\boldsymbol{U})\right\| \leq \Lambda \left\|(u,\boldsymbol{U})\right\|_* \tag{2.15}$$

for all $(u,\boldsymbol{U}) \in \dot{H}$.

Proof. We start with the first inequality. For $(u,\boldsymbol{U}) \in \dot{H}$, we choose a constant $c \in \mathbb{C}$ such that

$$\|u-c\|^2_{H^1(\Omega)} + \|\boldsymbol{U}-c\|^2_{\mathbb{C}^L} < \left\|(u,\boldsymbol{U})\right\|^2 + \epsilon, \tag{2.16}$$

for $\epsilon > 0$ arbitrary. Since

$$0 \leq (a-b)^2 = a^2 + b^2 - 2ab \quad \Leftrightarrow \quad 2ab \leq a^2 + b^2,$$

we have that

$$(a+b)^2 = a^2 + b^2 + 2ab \leq 2(a^2+b^2). \tag{2.17}$$

If we modify the norm (2.14) by using the triangle inequality and (2.17) we get

$$\begin{aligned}
\left\|(u,\boldsymbol{U})\right\|^2_* &= \|\nabla(u-c)\|^2_{L^2(\Omega)} + \sum_{l=1}^{L} \int_{e_l} |u(x) - c - (U_l - c)|^2 \, dS \\
&\leq \|\nabla(u-c)\|^2_{L^2(\Omega)} + \sum_{l=1}^{L} \int_{e_l} \left(|u(x)-c| + |U_l - c|\right)^2 dS \\
&\leq \|\nabla(u-c)\|^2_{L^2(\Omega)} + \sum_{l=1}^{L} \int_{e_l} 2\left(|u(x)-c|^2 + |U_l - c|^2\right) dS \\
&= \|\nabla(u-c)\|^2_{L^2(\Omega)} + 2\sum_{l=1}^{L} \int_{e_l} |u(x)-c|^2 \, dS + 2\sum_{l=1}^{L} |e_l| \, |U_l - c|^2 .
\end{aligned} \tag{2.18}$$

Since Ω is bounded and $\partial\Omega$ is C^1 we can apply the Trace Theorem in Appendix A.2 to get a bound on the second term of (2.18) as

$$2\sum_{l=1}^{L} \int_{e_l} |u(x)-c|^2 \, dS \leq 2\|u-c\|^2_{L^2(\partial\Omega)} \leq C_1 \|u-c\|^2_{H^1(\Omega)} .$$

This gives us a bound on the norm (2.14)

$$\begin{aligned}
\left\|(u,\boldsymbol{U})\right\|^2_* &\leq \|\nabla(u-c)\|^2_{L^2(\Omega)} + 2\sum_{l=1}^{L} \int_{e_l} |u(x)-c|^2 \, dS + 2\sum_{l=1}^{L} |e_l| \, |U_l - c|^2 \\
&\leq \|\nabla(u-c)\|^2_{L^2(\Omega)} + C_1 \|u-c\|^2_{H^1(\Omega)} + 2\|U_l - c\|^2_{\mathbb{C}^L} \max_{l=1,\cdots,L} |e_l| \\
&\leq \|u-c\|^2_{H^1(\Omega)} + C_1 \|u-c\|^2_{H^1(\Omega)} + 2\|U_l - c\|^2_{\mathbb{C}^L} \max_{l=1,\cdots,L} |e_l| \\
&\leq C_3 \left(\|u-c\|^2_{H^1(\Omega)} + \|U_l - c\|^2_{\mathbb{C}^L}\right).
\end{aligned}$$

By (2.16) this is bounded by

$$\left\|(u,\boldsymbol{U})\right\|^2_* \leq C_3 \left(\left\|(u,\boldsymbol{U})\right\|^2 + \epsilon\right).$$

Since this holds for arbitrary ϵ, the first inequality of (2.15) is proved.

Now we prove the second part of the inequality $\left\|(u,\boldsymbol{U})\right\| \leq \Lambda \left\|(u,\boldsymbol{U})\right\|_*$. This is done with a contradiction proof. If we assume that the claim is not true, then there

must exist a sequence $(z_n, \boldsymbol{Z}_n) \in \dot{H}$ for which there exists a constant $M > 0$ such that $\|(z_n, \boldsymbol{Z}_n)\|_* \leq M \quad \forall n \in \mathbb{N}$ and $\|(z_n, \boldsymbol{Z}_n)\| \to \infty$.

This means that given any $n \in \mathbb{N}$ we can pick out a subsequence, which we again denote $(z_n, \boldsymbol{Z}_n)$, such that

$$\|(z_n, \boldsymbol{Z}_n)\| > nM \quad \Leftrightarrow \quad \frac{1}{n} > \frac{M}{\|(z_n, \boldsymbol{Z}_n)\|}.$$

Now let us pick the normalized sequence $(u_n, \boldsymbol{U}_n) = \frac{(z_n, \boldsymbol{Z}_n)}{\|(z_n, \boldsymbol{Z}_n)\|}$. Then we have that

$$\|(u_n, \boldsymbol{U}_n)\| = 1, \qquad \|(u_n, \boldsymbol{U}_n)\|_* = \frac{\|(z_n, \boldsymbol{Z}_n)\|_*}{\|(z_n, \boldsymbol{Z}_n)\|} \leq \frac{M}{\|(z_n, \boldsymbol{Z}_n)\|} < \frac{1}{n}. \tag{2.19}$$

Now let (c_n) be a sequence in $\mathbb{C}$ and let

$$(w_n, \boldsymbol{W}_n) = (u_n - c_n, \boldsymbol{U}_n - c_n).$$

We have that

$$1 = \|(u_n, \boldsymbol{U}_n)\|^2 = \inf_{c \in \mathbb{C}} \left(\|u_n - c\|^2_{H^1(\Omega)} + \|\boldsymbol{U}_n - c\|^2_{\mathbb{C}^L} \right) \leq \|w_n\|^2_{H^1(\Omega)} + \|\boldsymbol{W}_n\|^2_{\mathbb{C}^L},$$

since the norm (2.14) takes the infimum over all constants in $\mathbb{C}$. Now, by choosing the constants (c_n) in the right way, we can get infinitely close to the norm $\|(u_n, \boldsymbol{U}_n)\|$. For instance we can choose $\epsilon_n < \frac{1}{n}$ yielding

$$\|w_n\|^2_{H^1(\Omega)} + \|\boldsymbol{W}_n\|^2_{\mathbb{C}^L} = \|u_n - c_n\|^2_{H^1(\Omega)} + \|\boldsymbol{U}_n - c_n\|^2_{\mathbb{C}^L} \leq \|(u_n, \boldsymbol{U_n})\| + \epsilon_n < 1 + \frac{1}{n}.$$

This gives us the inequality

$$1 \leq \|w_n\|^2_{H^1(\Omega)} + \|\boldsymbol{W}_n\|^2_{\mathbb{C}^L} < 1 + \frac{1}{n}. \tag{2.20}$$

Since Ω is open and bounded we can use the compact imbedding theorem in Appendx A.7 that gives us

$$H^1(\Omega) \subset\subset L^2(\Omega).$$

Since $\|w_n\|^2_{H^1(\Omega)} \leq 1 + \frac{1}{n} \leq 2$, (w_n) is bounded. It follows from the definition of continuous and compact imbeddings in Appendix A.4 that (w_n) has a subsequence, which is a Cauchy-sequence in $L^2(\Omega)$. We denote the subsequence (w_n). $L^2(\Omega)$ is complete, so (w_n) converges. That is, for some $w \in L^2(\Omega)$

$$w_n \to w \qquad \text{for } n \to \infty. \tag{2.21}$$

However, we notice that

$$\|\nabla w_n\|_{L^2(\Omega)} = \|\nabla u_n\|_{L^2(\Omega)} \leq \|(u_n, \boldsymbol{U}_n)\|_* < \frac{1}{n}, \tag{2.22}$$

which means that, (w_n) is a Cauchy sequence in $H^1(\Omega)$ that converges because $H^1(\Omega)$ is complete. Furthermore, the limit satisfies $\nabla w = 0$ so that $w = \text{const} = c_0$.

From (2.22) we have for $n \in \mathbb{N}$

$$\begin{aligned} \frac{1}{n} > \|(u_n, \boldsymbol{U}_n)\| \geq \|(u_n, \boldsymbol{U}_n)\|^2 &= \|\nabla u_n\|_{L^2(\Omega)^2} + \sum_{l=1}^{L} \int_{e_l} |u_n(x) - U_{n,l}|^2 \, dS \\ &\geq \int_{e_l} |u_n(x) - U_{n,l}|^2 \, dS = \int_{e_l} |(w_n - c_0) - (W_{n,l} - c_0)|^2 \, dS. \end{aligned} \tag{2.23}$$

Using that $\langle \boldsymbol{f}-\boldsymbol{g}, \boldsymbol{f}-\boldsymbol{g}\rangle = \|\boldsymbol{f}\|^2 + \|\boldsymbol{g}\|^2 - 2\,\mathrm{Re}\,\langle \boldsymbol{f}, \boldsymbol{g}\rangle$ we get

$$\begin{aligned}\frac{1}{n} &> \int_{e_l} |(w(x)-c_0)|^2 \, dS \\ &\quad + \int_{e_l} |(W_{n,l}-c_0)|^2 \, dS - 2\,\mathrm{Re}\left(\int_{e_l} (w_n - c_o)(W_{n,l}-c_o)\, dS\right) \\ &\geq -2|W_{n,l}-c_0| \int_{e_l} |w_n(x)-c_0| \, dS + |W_{n,l}-c_0|^2 |e_l|, \end{aligned} \tag{2.24}$$

for each $l = 1, 2..., L$. Rearranging (2.24) we get

$$|W_{n,l}-c_0|^2|e_l| < \frac{1}{n} + 2|W_{n,l}-c_0| \int_{e_l} |w_n(x)-c_0| \, dS. \tag{2.25}$$

Now considering the relation between one element and the norm, we notice that $|W_{n,l}| \leq \|\boldsymbol{W}_n\|_{\mathbb{C}^L} \leq \|w_n\|_{H^1(\Omega)} + \|\boldsymbol{W}_n\|_{\mathbb{C}^L} \leq 1 + \frac{1}{n}$, where the last part is seen from (2.20). From (2.25) using the triangle inequality we now see that

$$\begin{aligned}|W_{n,l}-c_0|^2|e_l| &< \frac{1}{n} + 2(|W_{n,l}| + |c_0|) \int_{e_l} |w_n(x)-c_0| \, dS \\ &\leq \frac{1}{n} + 2(1 + \frac{1}{n} + |c_0|) \int_{e_l} |w_n - c_0| \, dS. \end{aligned} \tag{2.26}$$

Rewriting the last integral and using Hölder's inequality we get

$$\begin{aligned}\int_{e_l} |w_n - c_0| \, dS &= \int_{\partial\Omega} |w_n - c_0|\chi_l \, dS \leq \|\chi_l\|_{L^2(\partial\Omega)} \|w_n - c_0\|_{L^2(\partial\Omega)} \\ &= |e_l|^{1/2} \|w_n - c_0\|_{L^2(\partial\Omega)}. \end{aligned}$$

Using the above in (2.26) followed by the trace theorem yields

$$\begin{aligned}|W_{n,l}-c_0|^2|e_l| &\leq \frac{1}{n} + 2(1 + \frac{1}{n} + |c_0|) \int_{e_l} |w_n - c_0| \, dS \\ &\leq \frac{1}{n} + 2(1 + \frac{1}{n} + |c_0|)|e_l|^{1/2} \|w_n - c_0\|_{L^2(\partial\Omega)} \\ &\leq \frac{1}{n} + C(1 + \frac{1}{n} + |c_0|)|e_l|^{1/2} \|w_n - c_0\|_{H^1(\Omega)}. \end{aligned}$$

Since $w_n \to c_0$ in $H^1(\Omega)$, wee see that $W_{n,l}$ converges to c_0. However, we also must have

$$1 = \|(u_n, \boldsymbol{U}_n)\|^2 \leq \|w_n - c_0\|_{H^1(\Omega)} + \|\boldsymbol{W}_n - c_0\|_{\mathbb{C}^L} \to 0, \tag{2.27}$$

which gives us a contradiction. Hereby the second norm inequality must be true, so we conclude that $\|\cdot\|$ and $\|.\|_*$ are equivalent norms in $\dot{H}$. □

In the following we will use $\|\cdot\|_*$ when proving existence and uniqueness with the Lax-Milgrams theorem.

Theorem 3. *Suppose that there are strictly positive constants σ_0, σ_1 and Z such that*

$$|\sigma| \le \sigma_1, \tag{2.28}$$

$$Re\ \sigma \ge \sigma_0, \tag{2.29}$$

and

$$Re\ z_l > Z \quad for\ l = 1, 2, \cdots, L. \tag{2.30}$$

Then for a given current pattern $(I_l)_{l=1}^L$ satisfying (2.1), there is a unique $(u, \boldsymbol{U})$ in $\dot{H}$ satisfying

$$B\left((u, \boldsymbol{U}), (v, \boldsymbol{V})\right) = \sum_{l=1}^{L} I_l \bar{V}_l, \tag{2.31}$$

for all $(v, \boldsymbol{V}) \in \dot{H}$.

Proof. We apply the Lax-Milgram lemma. To do this we check the three conditions for B.
We have sesquilinearity since for $\alpha_1, \alpha_2 \in \mathbb{C}$ and $(u_1, \boldsymbol{U}_1), (u_2, \boldsymbol{U}_2), (v, \boldsymbol{V}) \in \dot{H}$ we see

$$\begin{aligned}
&B\left(\alpha_1(u_1, \boldsymbol{U}_1) + \alpha_2(u_2, \boldsymbol{U}_2), (v, \boldsymbol{V})\right) \\
&= \int_\Omega \sigma \nabla(\alpha_1 u_1 + \alpha_2 u_2) \cdot \nabla \bar{v}\ dx \\
&\qquad\qquad + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (\alpha_1 u_1 + \alpha_2 u_2 - \alpha_1 U_{1,l} - \alpha_2 U_{2,l})(\bar{v} - \bar{V}_l)\ dS \\
&= \alpha_1 \int_\Omega \sigma \nabla u_1 \cdot \nabla \bar{v}\ dx + \alpha_2 \int_\Omega \sigma \nabla u_2 \cdot \nabla \bar{v}\ dx \\
&\qquad\qquad + \alpha_1 \sum_{l=1}^{L} \frac{1}{\hat{u}_l} \int_{e_l} (u_1 - U_{1,l})(\bar{v} - \bar{V}_l)\ dS \\
&\qquad\qquad\qquad + \alpha_2 \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u_2 - U_{2,l})(\bar{v} - \bar{V}_l)\ dS \\
&= \alpha_1 B\left((u_1, \boldsymbol{U}_1), (v, \boldsymbol{V})\right) + \alpha_2 B\left((u_2, \boldsymbol{U}_2), (v, \boldsymbol{V})\right),
\end{aligned}$$

and for $\beta_1, \beta_2 \in \mathbb{C}$ and $(u, \boldsymbol{U}), (v_1, \boldsymbol{V}_1), (v_2, \boldsymbol{V}_2) \in \dot{H}$ we get

$$
\begin{aligned}
&B\left((u,\boldsymbol{U}), \beta_1(v_1,\boldsymbol{V}_1) + \beta_2(v_2,\boldsymbol{V}_2)\right) \\
&= \int_\Omega \sigma \nabla u \cdot \nabla(\bar{\beta}_1 \bar{v}_1 + \bar{\beta}_2 \bar{v}_2)\, dx \\
&\qquad\qquad + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{\beta}_1 \bar{v}_1 + \bar{\beta}_2 \bar{v}_2 - \bar{\beta}_1 \bar{V}_{1,l} - \bar{\beta}_2 \bar{V}_{2,l})\, dS \\
&= \bar{\beta}_1 \int_\Omega \sigma \nabla u \cdot \nabla \bar{v}_1\, dx + \bar{\beta}_2 \int_\Omega \sigma \nabla u \cdot \nabla \bar{v}_2\, dx \\
&\qquad\qquad + \bar{\beta}_1 \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v}_1 - \bar{V}_{1,l})\, dS \\
&\qquad\qquad\qquad + \bar{\beta}_2 \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v}_2 - \bar{V}_{2,l})\, dS \\
&= \bar{\beta}_1 B\left((u,\boldsymbol{U}), (v_1,\boldsymbol{V}_1)\right) + \bar{\beta}_2 B\left((u,\boldsymbol{U}), (v_2,\boldsymbol{V}_2)\right)
\end{aligned}
$$

Secondly, we want to show boundedness. That is there exist a constant $\gamma > 0$ such that $|B(a,b)| \leq \gamma \|a\|_* \|b\|_*$.

Let $a = (u, \boldsymbol{U})$ and $b = (v, \boldsymbol{V})$. By using the assumptions that $Re(z_l) > Z$ for all $l = 1, 2, \cdots, L$ and $|\sigma| \leq \sigma_1$ along with the fact that $\frac{1}{|z_l|} \leq \frac{1}{\mathrm{Re}(z_l)} < \frac{1}{Z}$ we get

$$
\begin{aligned}
|B(a,b)| &= \left| \int_\Omega \sigma \nabla u \cdot \nabla \bar{v}\, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l)\, dS \right| \\
&\leq \left| \int_\Omega \sigma \nabla u \cdot \nabla \bar{v}\, dx \right| + \left| \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l)\, dS \right| \\
&\leq \int_\Omega |\sigma \nabla u \cdot \nabla \bar{v}|\, dx + \sum_{l=1}^{L} \frac{1}{|z_l|} \int_{e_l} |u - U_l| \left|\bar{v} - \bar{V}_l\right|\, dS \\
&\leq \sigma_1 \int_\Omega |\nabla u \cdot \nabla \bar{v}|\, dx + \frac{1}{Z} \sum_{l=1}^{L} \int_{e_l} |u - U_l| \left|\bar{v} - \bar{V}_l\right|\, dS \\
&\leq \max\left(\sigma_1, \frac{1}{Z}\right) \left(\int_\Omega |\nabla u \cdot \nabla \bar{v}|\, dx + \sum_{l=1}^{L} \int_\Omega \chi_{e_l} |u - U_l| \left|\bar{v} - \bar{V}_l\right|\, dS \right).
\end{aligned}
$$

If we now use Hölder's inequality, the fact that $\|\nabla u\|_{L^2(\Omega)} \leq \|(u,\boldsymbol{U})\|_*$, $\left\|\chi_{e_l}(u - U_l)\right\|_{L^2(\Omega)} \leq$

$\|(u,\boldsymbol{U})\|_*$, and $\left\|\chi_{e_l}(v-V_l)\right\|_{L^2(\Omega)} \leq \|(v,\boldsymbol{V})\|_*$. Then we have

$$\begin{aligned}
|B(a,b)| &\leq \max\left(\sigma_1, \frac{1}{Z}\right)\Big(\|\nabla u\|_{L^2(\Omega)}\|\nabla \bar{v}\|_{L^2(\Omega)} \\
&\qquad + \sum_{l=1}^{L}\left\|\chi_{e_l}(u-U_l)\right\|_{L^2(\Omega)}\left\|\chi_{e_l}\left(\bar{v}-\bar{V}_l\right)\right\|_{L^2(\Omega)}\Big) \\
&\leq \max\left(\sigma_1, \frac{1}{Z}\right)\Big(\|(u,\boldsymbol{U})\|_*\|\nabla \bar{v}\|_{L^2(\Omega)} \\
&\qquad + \sum_{l=1}^{L}\|(u,\boldsymbol{U})\|_*\left\|\chi_{e_l}\left(\bar{v}-\bar{V}_l\right)\right\|_{L^2(\Omega)}\Big) \\
&= \max\left(\sigma_1, \frac{1}{Z}\right)\|(u,\boldsymbol{U})\|_*\left(\|\nabla \bar{v}\|_{L^2(\Omega)} + \sum_{l=1}^{L}\left\|\chi_{e_l}\left(\bar{v}-\bar{V}_l\right)\right\|_{L^2(\Omega)}\right) \\
&\leq \max\left(\sigma_1, \frac{1}{Z}\right)\|(u,\boldsymbol{U})\|_*\left(\|(v,\boldsymbol{V})\|_* + \sum_{l=1}^{L}\|(v,\boldsymbol{V})\|_*\right) \\
&= \max\left(\sigma_1, \frac{1}{Z}\right)(L+1)\|(u,\boldsymbol{U})\|_*\|(v,\boldsymbol{V})\|_* \\
&= \gamma\|a\|_*\|b\|_*,
\end{aligned}$$

as desired.

The last thing we need to show is coercivity. That is there exist a constant $\delta > 0$ such that $|B(a,a)| \geq \delta\|a\|_*^2$.
Let

$$\|a\|_*^2 = \|(u,\boldsymbol{U})\|_*^2 = \|\nabla u\|_{L^2(\Omega)}^2 + \sum_{l=i}^{L}\int_{e_l}|u-U_l|^2\,dS$$

and

$$|B(a,a)| = \left|\int_\Omega \sigma|\nabla u|^2\,dS + \sum_{l=i}^{L}\frac{1}{z_l}\int_{e_l}|u-U_l|^2\,dS\right|.$$

Since σ is complex and $z_l \in \mathbb{C}$ for all $l = 1, 2, \cdots, L$, and the fact that $|a+ib| = \sqrt{a^2+b^2} \geq \sqrt{a^2} = |a|$ we have that

$$\begin{aligned}
|B(a,a)| &= \Bigg|\int_\Omega \left(\mathrm{Re}(\sigma) + i\,\mathrm{Im}(\sigma)\right)|\nabla u|^2\,dS \\
&\qquad + \sum_{l=i}^{L}\left(\mathrm{Re}\left(\frac{1}{z_l}\right) + i\,\mathrm{Im}\left(\frac{1}{z_l}\right)\right)\int_{e_l}|u-U_l|^2\,dS\Bigg| \\
&\geq \left|\int_\Omega \mathrm{Re}(\sigma)|\nabla u|^2\,dS + \sum_{l=i}^{L}\mathrm{Re}\left(\frac{1}{z_l}\right)\int_{e_l}|u-U_l|^2\,dS\right|.
\end{aligned}$$

Since we know that $\mathrm{Re}\left(\frac{1}{z_l}\right) = \frac{\mathrm{Re}(z_l)}{|z_l|}$ and using $\mathrm{Re}(\sigma) \geq \sigma_0 > 0$ and $\mathrm{Re}(z_l) > Z > 0$ for all $l = 1, 2, \cdots, L$ we can remove the outer absolute value and reduce the equation further

$$
\begin{aligned}
|B(a,a)| &\geq \left| \int_\Omega \mathrm{Re}(\sigma)|\nabla u|^2 \, dS + \sum_{l=i}^{L} \mathrm{Re}\left(\frac{1}{z_l}\right) \int_{e_l} |u - U_l|^2 \, dS \right| \\
&= \int_\Omega \mathrm{Re}(\sigma)|\nabla u|^2 \, dS + \sum_{l=i}^{L} \frac{\mathrm{Re}(z_l)}{|z_l|} \int_{e_l} |u - U_l|^2 \, dS \\
&\geq \sigma_0 \int_\Omega |\nabla u|^2 \, dS + \frac{Z}{\max_l(|z_l|)} \sum_{l=i}^{L} \int_{e_l} |u - U_l|^2 \, dS \\
&\geq \min\left(\sigma_0, \frac{Z}{\max_l |z_l|}\right) \left(\int_\Omega |\nabla u|^2 \, dS + \sum_{l=i}^{L} \int_{e_l} |u - U_l|^2 \, dS \right) \\
&= \delta \left\| (u, \boldsymbol{U}) \right\|_*^2 = \delta \left\| a \right\|_*^2 .
\end{aligned}
$$

By choosing the constant $\delta = \min\left(\sigma_0, \frac{Z}{\max_l |z_l|}\right)$ we get the desired result.

We have now checked (a)-(c) for our B, so what is left is to check that the linear functional, $f : (v, \boldsymbol{V}) \to \sum_{l=1}^{L} I_l \bar{V}_l$, on the right-hand side of (2.31) is well defined and continuous.

To show that f is well defined we check that two equivalent elements in $\dot{H}$ are mapped into the same element. So assume that $(v, \boldsymbol{V}) \sim (\tilde{v}, \tilde{\boldsymbol{V}})$, so that $\tilde{V}_l = V_l - \text{const}$, $l = 1, \cdots, L$. Then we have by (2.1)

$$
f(v, \boldsymbol{V}) = \sum_{l=1}^{L} I_l \bar{V}_l = \sum_{l=1}^{L} I_l (\bar{V}_l - \text{const}) = \sum_{l=1}^{L} I_l \bar{\tilde{V}}_l = f(\tilde{v}, \tilde{\boldsymbol{V}}).
$$

This shows that f is well defined.

Since the domain of f is a normed space we show that it is bounded and conclude that it is also continuous. Let $c \in \mathbb{C}$ be a constant such that

$$
\left(\|v - c\|_{H^1(\Omega)}^2 + \|V - c\|_{\mathbb{C}^L}^2 \right)^{1/2} \leq \|(v, \boldsymbol{V})\| + \epsilon,
$$

which tells us especially that

$$
\|V - c\|_{\mathbb{C}^L} \leq \|(v, \boldsymbol{V})\| + \epsilon. \tag{2.32}
$$

Using first Cauchy-Schwarz and then (2.32) we get

$$
\begin{aligned}
|f(v, \boldsymbol{V})| &= \left| \sum_{l=1}^{L} I_l (\bar{V}_l - c) \right| \\
&\leq \|I\|_{\mathbb{C}^L} \|V - c\|_{\mathbb{C}^L} \\
&\leq \|I\|_{\mathbb{C}^L} \left(\|(v, \boldsymbol{V})\| + \epsilon \right),
\end{aligned}
$$

which shows that f is bounded and hence continuous as $\epsilon > 0$ can be chosen . □

Remark 1. *We have now shown that there is a unique solution to the problem in $\dot{H}$. But this gives us infinitely many solutions in H, since $\dot{H}$ is an equivalence class of solutions. In order to get a unique solution $(u, \boldsymbol{U})$ in H we add the final constraint*

$$\sum_{l=1}^{L} U_l = 0. \tag{2.33}$$

To prove that this yields a unique solution in H we pick two arbitrary solutions $(u, \boldsymbol{U}), (\hat{u}, \hat{\boldsymbol{U}}) \in H$ to (2.6). We must have the relationship $(\hat{u}, \hat{\boldsymbol{U}}) = (u+k, U+k)$ for some $k \in \mathbb{C}$. Because of the constraint (2.33), we must have

$$\sum_{l=1}^{L} U_l = 0, \qquad \sum_{l=1}^{L} \hat{U}_l = 0.$$

Now summing the $\hat{U}_l$'s yields

$$0 = \sum_{l=1}^{L} \hat{U}_l = \sum_{l=1}^{L} (U_l + k) = \sum_{l=1}^{L} U_l + \sum_{l=1}^{L} k = Lk,$$

requiring $k = 0$, since we have $L > 0$.
This proves that a unique solution to (2.6) in $\dot{H}$ also becomes a unique solution to the same problem in the space H given the constraint (2.33).

2.3 Solving the CEM for a Simple Geometry

In this section, we will find a solution to a problem in a simple geometry using the CEM.

The domain chosen is the square $\Omega = [0, 1] \times [0, 1]$ as seen from figure 2.1. Note that this domain is not C^1 in the corner points, but since the solution in those points are not interesting, we have no problem. Two electrodes are attached to the domain: One on the line $x = 0$, and one on the line $x = 1$. The conductivity is assumed constant and is set to $\sigma = 1$ and the impedance coefficients are assumed to be the same $z_1 = z_2 = z$. We choose these conditions because it is possible to find an analytical solution for comparison, which we have do in B.

The PDE formulation of this problem can be generated using (2.2)-(2.4)

$$\Delta u = 0 \quad \text{in } \Omega \tag{2.34}$$

subject to the boundary conditions on $\partial\Omega$

$$\begin{aligned} u - z u_x(0, y) &= U \\ u + z u_x(1, y) &= -U, \end{aligned} \tag{2.35}$$

and

$$\begin{aligned} -\int_0^1 u_x(0, y)\, dy &= I \\ \int_0^1 u_x(1, y)\, dy &= -I. \end{aligned} \tag{2.36}$$

The weak formulation of this problem is generated using (2.6) and is

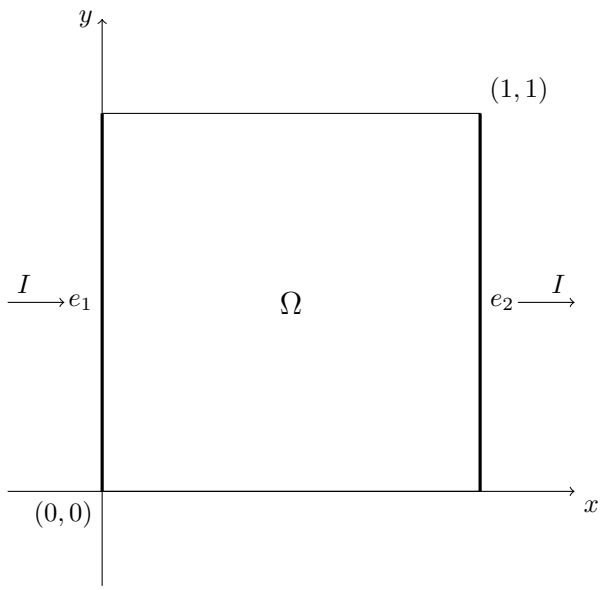


Figure 2.1: *Sketch of the square domain with electrodes attached.*

$$\int_\Omega \nabla u \cdot \nabla \bar{v}\ dx + \frac{1}{z}\int_0^1 (u(0,y) - U)(\bar{v} - \bar{V}_l)\ dy + \frac{1}{z}\int_0^1 (u(1,y) + U)(\bar{v} - \bar{V}_l)\ dy = \bar{V}_1 I - \bar{V}_2 I \quad (2.37)$$

In Python this problem is easy to solve with the Finite Elmenent Method (FEM) using the Fenics library [7]. In general when using FEniCS to find a weak solutions with FEM, it is done by the following steps:

1. Create and initialize parameters (U, I, z_l, σ)
2. Create or load mesh
3. Define and initialize subdomains (for each electrode)
4. Create functionspaces
5. Create trial- and test functions (R and $CG1$, which is piecewise affine elements)
6. Write up the weak formulation, which in FEniCS is called the variational form
7. Use solver
8. Split solution into relevant variables (u and $\boldsymbol{U}$)

The code for this can be seen in D.5 and a plotted solution can be seen in Figure 2.2. If we compare this to the analytical solution in Figure B.1, we see that the solutions are identical.

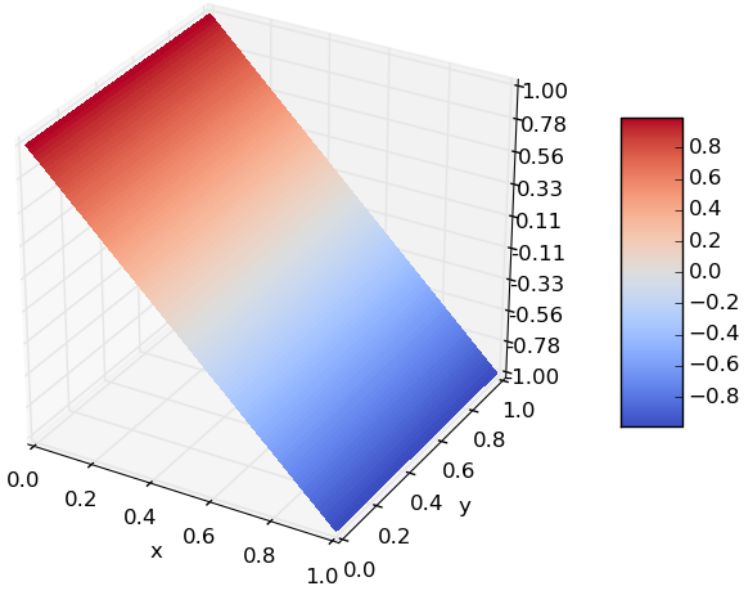


Figure 2.2: *Solution in square domain with I=2*

Chapter 3

Linearization

3.1 The Current-to-Voltage Map

In this section we investigate the current-to-voltage operator

$$R_\sigma \ : \ \mathbb{C}_\diamond^L \to \mathbb{C}_\diamond^L,$$

where

$$\mathbb{C}_\diamond^L = \left\{ \boldsymbol{X} \in \mathbb{C}^L \mid \sum_{l=1}^{L} X_l = 0 \right\},$$

that maps current vectors $\boldsymbol{I} \in \mathbb{C}_\diamond^L$ to voltage vectors $\boldsymbol{U} \in \mathbb{C}_\diamond^L$. The reason for the investigation of R_σ is that it both gives us useful information regarding the forward problem and later will be important in order to solve the inverse problem. It has been shown experimentally that the relation between $\boldsymbol{I}$ and $\boldsymbol{U}$ is linear [8]. This means that R_σ can be represented as a matrix, which we denote $\mathbf{R}_\sigma$. Since $\mathbb{C}_\diamond^L$ has the condition that vectors sum to zero, it has dimension $L-1$, so we can find a basis of $L-1$ vectors. Assume that $\{\boldsymbol{I}^j\}_{j=1}^{L-1}$ is an orthonormal basis. Then we can write

$$R_\sigma \boldsymbol{I} = R_\sigma \sum_{j=1}^{L-1} \left\langle \boldsymbol{I}, \boldsymbol{I}^j \right\rangle \boldsymbol{I}^j.$$

Using this relation, we can find the i'th basis coefficient of $\mathbf{R}_\sigma \boldsymbol{I}$ by

$$\left\langle R_\sigma \boldsymbol{I}, \boldsymbol{I}^i \right\rangle = \left\langle R_\sigma \sum_{j=1}^{L-1} \left\langle \boldsymbol{I}, \boldsymbol{I}^j \right\rangle \boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle = \sum_{j=1}^{L-1} \left\langle \boldsymbol{I}, \boldsymbol{I}^j \right\rangle \left\langle R_\sigma \boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle.$$

We notice that this is a vector product so that one representation of R_σ is the matrix $\mathbf{R}_\sigma$ with elements

$$\left[\mathbf{R}_\sigma\right]_{i,j} = \left\langle R_\sigma \boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle. \tag{3.1}$$

It is noted that $R_\sigma \boldsymbol{I}^j$ corresponds to the voltage vector that results from the solution to (2.2)-(2.5) with the current vector $\boldsymbol{I}^j$ and the conductivity σ. Therefore there are two options for finding R_σ. Either one can simulate voltage vectors as just explained or one can use EIT to measure the voltages on the electrodes. In this way using the simulation to generate R_σ corresponds to solving the forward problem.

Remark 2. *We remark that* $\boldsymbol{U}$ *and* $\boldsymbol{I}$ *have* L *elements, but the matrix* $\mathbf{R}_\sigma$ *has dimensions* $(L-1)\times(L-1)$*. This means that if one wanted to calculate the output voltages given a current vector, one would have to use a basis change matrix* $\mathbf{B} = [\boldsymbol{I}^1, \boldsymbol{I}^2, ..., \boldsymbol{I}^{L-1}]$ *with the dimension* $L \times (L-1)$ *yielding*

$$\boldsymbol{U} = \mathbf{B}\mathbf{R}_\sigma \overline{\mathbf{B}}^T \boldsymbol{I}$$

3.2 The Fréchet Derivative

In the inverse problem, we are interested in reconstructing a perturbation, h, from a background conductivity, σ, based on knowledge about $R_{\sigma+h}$. To do so, we have to investigate the mapping $h \mapsto R_{\sigma+h}$. First, we need to define a new domain. We need this domain since we will only consider perturbation that are zero close to the boundary.

Definition 4. *For* $\epsilon > 0$ *we define the domain*

$$\Omega_\epsilon = \left\{ x \in \overline{\Omega} \mid d(x, \partial\Omega) < \epsilon \right\}.$$

Using this domain we define

$$\Omega' = \Omega \backslash \Omega_\epsilon.$$

This is a compact domain and we define

$$L^\infty(\Omega') = \{ f \in L^\infty(\Omega) \mid \operatorname{supp} f \subseteq \Omega' \}.$$

We can now define the mapping

$$F : \ L^\infty(\Omega') \to B(\mathbb{C}^L_\diamond, \mathbb{C}^L_\diamond) \ : \ h \mapsto R_{\sigma+h}, \tag{3.2}$$

where $B(\mathbb{C}^L_\diamond, \mathbb{C}^L_\diamond)$ is the Banach space of linear and continuous operators that maps from $\mathbb{C}^L_\diamond$ into $\mathbb{C}^L_\diamond$.

If we linearize in a neighbourhood of σ corresponding to linearizing around $F(0)$, we can predict what happens for a small perturbation h. To do this we will use the Fréchet derivative defined in Appendix A.9. We let $R'_\sigma = F(0)'$ denote the Fréchet derivative with respect to h and $R'_\sigma[h]$ denote the Fréchet derivative in the "direction" $h \in L^\infty(\Omega')$. The Fréchet derivative is the mapping

$$R'_\sigma : \ L^\infty(\Omega') \to B(\mathbb{C}^L_\diamond, \mathbb{C}^L_\diamond).$$

To be the Fréchet derivative R'_σ must be a bounded linear operator that satisfies

$$R_{\sigma+h} = R_\sigma + R'_\sigma[h] + o(h),$$

which is equivalent to

$$\lim_{\|h\|_{L^\infty(\Omega')} \to 0} \frac{1}{\|h\|_{L^\infty(\Omega')}} \left\| R_{\sigma+h} - R_\sigma - R'_\sigma[h] \right\|_{op} = 0, \tag{3.3}$$

where the operator norm is

$$\left\| R_{\sigma+h} - R_\sigma - R'_\sigma[h] \right\|_{op} = \sup_{\substack{\boldsymbol{I}, \boldsymbol{J} \in \mathbb{C}^L_\diamond \\ \|\boldsymbol{I}\|_{\mathbb{C}^L} = \|\boldsymbol{J}\|_{\mathbb{C}^L} = 1}} \left| \langle (R_{\sigma+h} - R_\sigma - R'_\sigma[h])\boldsymbol{J}, \boldsymbol{I} \rangle \right|. \tag{3.4}$$

Before we introduce the Fréchet derivative we state a lemma, which we will need in the later proof.

Lemma 5. *let $\{z_l\}_{l=1}^L$ be real-valued and let $\sigma \in L^\infty(\Omega)$ be a real positive function, then the matrix R_σ is real-valued and self-adjoint.*

Proof. We start out by using that the adjoint operator of R_σ is $(R_\sigma)^* = \overline{R_\sigma}$ in the sense of complex conjugation of the matrix. This is shown in [8]. To show that $(R_\sigma)^* = R_\sigma$ we need to show that $\overline{R_\sigma} = R_\sigma$, i.e. that R_σ is real-valued.

Let $(w^{\boldsymbol{J}}, \boldsymbol{W^J})$ be the soulution to (2.2)-(2.5) with current vector $\boldsymbol{J}$. We now split the PDE problem into a real and imaginary PDE problem. Since the operator $\mathrm{Im} : \mathbb{C} \to \mathbb{R}$ is linear, we get the following PDE for the imaginary part

$$\nabla \cdot \sigma \nabla \operatorname{Im}(w^{\boldsymbol{J}}) = 0 \qquad \text{in } \Omega, \tag{3.5}$$

$$\operatorname{Im}(w^{\boldsymbol{J}}) + z_l \sigma \frac{\partial \operatorname{Im}(w^{\boldsymbol{J}})}{\partial \nu} = \operatorname{Im}(W_l^J) \qquad \text{on } e_l,\ l = 1, 2, \cdots, L\ , \tag{3.6}$$

$$\sigma \frac{\partial \operatorname{Im}(w^{\boldsymbol{J}})}{\partial \nu} = 0 \qquad \text{on } \partial\Omega \backslash \overline{\bigcup_{l=1}^{L} e_l}, \tag{3.7}$$

$$\int_{e_l} \sigma \frac{\partial \operatorname{Im}(w^{\boldsymbol{J}})}{\partial \nu}\, dS = \operatorname{Im}(J_l) \qquad \text{for } l = 1, 2, \cdots, L. \tag{3.8}$$

Assuming the current is real-valued, then the unique solution to (3.5)-(3.8) is $(\operatorname{Im}(w), \operatorname{Im}(W)) = (0, 0)$, i.e. the voltage is also real-valued. This means that we can find a real-valued orthonormal basis $\{\boldsymbol{I}^k\}_{k=1}^{L-1}$ for $\mathbb{C}^L_\diamond$ such that the matrix elements $\left\langle R_\sigma \boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle_{i,j=1}^{L-1}$ are real-valued since R_σ maps real-valued currents into real-valued voltages, $R_\sigma : \mathbb{R}^L_\diamond \to \mathbb{R}^L_\diamond$. As such we conclude that $(R_\sigma)^* = R_\sigma$, i.e. R_σ is self-adjoint. □

This leads to the Fréchet derivative for our problem.

Theorem 6. *Let the contact impedances be real-valued, i.e. $z_l \in \mathbb{R}, \quad l = 1, 2, .., L$, the conductivity, σ, be real, and let $h \in L^\infty(\Omega')$. Furthermore, let $(w^{\boldsymbol{I}}, \boldsymbol{W^I})$ and $(w^{\boldsymbol{J}}, \boldsymbol{W^J})$ be unique solutions to* (2.2)-(2.5) *with currents $\boldsymbol{I}$ and $\boldsymbol{J}$ respectively and conductivity σ. Then the Fréchet derivative of the mapping F in* (3.2) *is*

$$\langle R'_\sigma[h]\boldsymbol{J}, \boldsymbol{I}\rangle = -\int_\Omega h \nabla w^{\boldsymbol{J}} \cdot \overline{\nabla w^{\boldsymbol{I}}}\, \mathrm{d}x. \tag{3.9}$$

The proof of 3.9 is done in three parts. First an expression for $R_{\sigma+h} - R_\sigma$ is found by use of the weak formulations. Next the PDE problem (2.2)-(2.5) is extended to a more general case. Finally, the extension will be applied to obtain a bound on $R_{\sigma+h} - R_\sigma - R'_\sigma[h]$, which is $o(h)$.

Proof. **Part 1 - Expression for $\langle (R_{\sigma+h} - R_\sigma)\boldsymbol{J}, \boldsymbol{I}\rangle$**

To prove that (3.9) is the Fréchet derivative we need to investigate

$$\langle (R_{\sigma+h} - R_\sigma - R'_\sigma[h])\boldsymbol{J}, \boldsymbol{I}\rangle = \langle (R_{\sigma+h} - R_\sigma)\boldsymbol{J}, \boldsymbol{I}\rangle - \langle R'_\sigma[h]\boldsymbol{J}, \boldsymbol{I}\rangle\,.$$

We start with the first inner product, $\langle (R_{\sigma+h} - R_\sigma)\boldsymbol{J}, \boldsymbol{I}\rangle$.

Let $(u^{\boldsymbol{I}}, \boldsymbol{U^I})$ be the unique solution to (2.2)-(2.5) with current vector $\boldsymbol{I}$ and conductivity $\sigma + h$. Let $(w^{\boldsymbol{J}}, \boldsymbol{W^J})$ be the unique solution to (2.2)-(2.5) with current vector $\boldsymbol{J}$ and conductivity σ. Then we know according to (2.6) that the corresponding weak

formulations for the two PDE problems are

$$\langle \boldsymbol{I}, \boldsymbol{V} \rangle = \int_\Omega (\sigma + h) \nabla u^{\boldsymbol{I}} \cdot \overline{\nabla v} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u^{\boldsymbol{I}} - U_l^{\boldsymbol{I}})(\overline{v} - \overline{V}_l) \, dS, \quad \forall (v, \boldsymbol{V}) \in H, \tag{3.10}$$

$$\langle \boldsymbol{J}, \boldsymbol{V} \rangle = \int_\Omega \sigma \nabla w^{\boldsymbol{J}} \cdot \overline{\nabla v} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (w^{\boldsymbol{J}} - W_l^{\boldsymbol{J}})(\overline{v} - \overline{V}_l) \, dS, \quad \forall (v, \boldsymbol{V}) \in H. \tag{3.11}$$

Using that (3.10) and (3.11) hold for all $(v, \boldsymbol{V}) \in H$, we can insert $(v, \boldsymbol{V}) = (w^{\boldsymbol{J}}, \boldsymbol{W}^{\boldsymbol{J}})$ in (3.10) and $(v, \boldsymbol{V}) = (u^{\boldsymbol{I}}, \boldsymbol{U}^{\boldsymbol{I}})$ in (3.11). Furthermore, instead of inserting the voltages $\boldsymbol{W}^{\boldsymbol{J}}$ and $\boldsymbol{U}^{\boldsymbol{I}}$ directly, we use that $\boldsymbol{W}^{\boldsymbol{J}} = R_\sigma \boldsymbol{J}$ and $\boldsymbol{U}^{\boldsymbol{I}} = R_{\sigma+h} \boldsymbol{I}$. Doing so we get

$$\langle \boldsymbol{I}, R_\sigma \boldsymbol{J} \rangle = \int_\Omega (\sigma + h) \nabla u^{\boldsymbol{I}} \cdot \nabla \overline{w^{\boldsymbol{J}}} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u^{\boldsymbol{I}} - [R_{\sigma+h} \boldsymbol{I}]_l)(\overline{w^{\boldsymbol{J}}} - \overline{[R_\sigma \boldsymbol{J}]_l}) \, dS, \tag{3.12}$$

$$\langle \boldsymbol{J}, R_{\sigma+h} \boldsymbol{I} \rangle = \int_\Omega \sigma \nabla w^{\boldsymbol{J}} \cdot \nabla \overline{u^{\boldsymbol{I}}} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (w^{\boldsymbol{J}} - [R_\sigma \boldsymbol{J}]_l)(\overline{u^{\boldsymbol{I}}} - \overline{[R_{\sigma+h} \boldsymbol{I}]_l}) \, dS. \tag{3.13}$$

We know from Lemma 5 that R_σ is self-adjoint. Using this and inserting (3.10) and (3.11), we get

$$\begin{aligned} \langle (R_{\sigma+h} - R_\sigma) \boldsymbol{J}, \boldsymbol{I} \rangle &= \langle R_{\sigma+h} \boldsymbol{J}, \boldsymbol{I} \rangle - \langle R_\sigma \boldsymbol{J}, \boldsymbol{I} \rangle = \langle \boldsymbol{J}, R_{\sigma+h} \boldsymbol{I} \rangle - \overline{\langle \boldsymbol{I}, R_\sigma \boldsymbol{J} \rangle} \\ &= - \int_\Omega h \nabla w^{\boldsymbol{J}} \cdot \overline{\nabla u^{\boldsymbol{I}}} \, \mathrm{d}x. \end{aligned} \tag{3.14}$$

Part 2 - Extension of (2.2)**-**(2.5)

We now look at a more general PDE problem than (2.2)-(2.5). So let everything be as in Proposition 1. Now we consider the PDE problem

$$\nabla \cdot \sigma \nabla u = -\nabla \cdot h \nabla g \qquad \text{in } \Omega \,, \tag{3.15}$$

$$u + z_l \sigma \frac{\partial u}{\partial \nu} = U_l \qquad \text{on } e_l, \ l = 1, 2, \cdots, L \,, \tag{3.16}$$

$$\int_{e_l} \sigma \frac{\partial u}{\partial \nu} \, dS = I_l \qquad \text{for } l = 1, 2, \cdots, L \,, \tag{3.17}$$

$$\sigma \frac{\partial u}{\partial \nu} = 0 \qquad \text{on } \partial\Omega \backslash \overline{\bigcup_{l=1}^{L} e_l}, \tag{3.18}$$

where $g \in H^1(\Omega)$. Following a similar proof as in Proposition 1 we arrive at the weak formulation

$$\begin{aligned} &\int_\Omega \sigma \nabla u \cdot \nabla \bar{v} \, dx + \sum_{l=1}^{L} \frac{1}{z_l} \int_{e_l} (u - U_l)(\bar{v} - \bar{V}_l) \, dS \\ &\qquad = \langle \boldsymbol{I}, \boldsymbol{V} \rangle_{\mathbb{C}^L} - \langle h \nabla g, \nabla v \rangle \,, \ \forall (v, \boldsymbol{V}) \in \dot{H}. \end{aligned} \tag{3.19}$$

Writing (3.19) as an operator equation on the form

$$B((u,\boldsymbol{U}),(v,\boldsymbol{V})) = f((v,\boldsymbol{V})),$$

we recognize B as the sesquilinear operator from Proposition 1, which we have shown is bounded and coercive with the lower bound

$$|B((u,\boldsymbol{U}),(u,\boldsymbol{U}))| \geq \min\left(\sigma_0, \frac{Z}{\max\limits_l |z_l|}\right) \|(u,\boldsymbol{U})\|_*^2, \tag{3.20}$$

if σ satisfies the asumptions from Theorem 3. Looking at the right-hand side $f((v,\boldsymbol{V}))$ we see that it is anti-linear - since ∇ is linear and the inner product is anti-linear in the second entrance - and well-defined for $(v,\boldsymbol{V}) \in \dot{H}$. We know that the first part of f is bounded from the proof of Theorem 3 and by using the triangle inequality and the Cauchy-Schwarz inequality we see that

$$\begin{aligned}
|f((v,\boldsymbol{V}))| &= \left|\langle \boldsymbol{I},\boldsymbol{V}\rangle_{\mathbb{C}^L} - \langle h\nabla g, \nabla v\rangle_{L^2(\Omega)}\right| && (3.21)\\
&\leq |\langle \boldsymbol{I},\boldsymbol{V}\rangle_{\mathbb{C}^L}| + \left|\langle h\nabla g, \nabla v\rangle_{L^2(\Omega)}\right| && (3.22)\\
&\leq \|\boldsymbol{I}\|_{\mathbb{C}^L} \|(v,\boldsymbol{V})\| + \|h\nabla g\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)}. && (3.23)
\end{aligned}$$

Now using the equivalence of the norms (2.13) and (2.14), that $\|\nabla g\|_{L^2(\Omega)} \leq \|g\|_{H^1(\Omega)}$ and that $\|h\nabla g\|_{L^2(\Omega)} \leq \|h\|_{L^\infty(\Omega')} \|\nabla g\|_{L^2(\Omega)}$ we arrive at

$$\begin{aligned}
|f((v,\boldsymbol{V}))| &\leq \|\boldsymbol{I}\|_{\mathbb{C}^L} \|(v,\boldsymbol{V})\| + \|h\nabla g\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)} && (3.24)\\
&\leq C\|\boldsymbol{I}\|_{\mathbb{C}^L} \|(v,\boldsymbol{V})\|_* + \|h\|_{L^\infty(\Omega')} \|g\|_{H^1(\Omega)} \|\nabla v\|_{L^2(\Omega)} && (3.25)\\
&\leq \left(C\|\boldsymbol{I}\|_{\mathbb{C}^L} + \|h\|_{L^\infty(\Omega')} \|g\|_{H^1(\Omega)}\right) \|(v,\boldsymbol{V})\|_*. && (3.26)
\end{aligned}$$

Hence f is bounded and therefore continous, so by applying the Lax-Milgram theorem we have that there exists a solution to the general PDE problem (3.15)-(3.18) and that it is unique.

Part 3 - Using the extension on a special case

Now consider the inhomogenous PDE problem as defined in (3.15)-(3.18) with $v = u^{\boldsymbol{I}} - w^{\boldsymbol{I}}$ and $\boldsymbol{V} = \boldsymbol{U^I} - \boldsymbol{W^J}$. Since we have $\nabla\cdot((\sigma+h)\nabla u^{\boldsymbol{I}}) = 0$ and $\nabla\cdot(\sigma\nabla w^{\boldsymbol{I}}) = 0$, v satisfies

$$\nabla\cdot(\sigma+h)\nabla v = \nabla\cdot(\sigma+h)\nabla u^{\boldsymbol{I}} - \nabla\cdot\sigma\nabla w^{\boldsymbol{I}} - \nabla\cdot h\nabla w^{\boldsymbol{I}} = -\nabla\cdot h\nabla w^{\boldsymbol{I}}.$$

h is assumed to vanish on the boundary. Thus, all the products with h in the boundary conditions will vanish. Furthermore, we notice that we get a homogenous boundary condition for the input current

$$\int\limits_{e_l} \sigma\frac{\partial v}{\partial\nu}\, dS = \int\limits_{e_l} (\sigma+h)\frac{\partial v}{\partial\nu}\, dS = \int\limits_{e_l} (\sigma+h)\frac{\partial(u^{\boldsymbol{I}} - w^{\boldsymbol{I}})}{\partial\nu}\, dS = I_l - I_l = 0.$$

Hence, $(v, \boldsymbol{V})$ is the solution to the following PDE problem

$$\nabla \cdot (\sigma + h)\nabla v = -\nabla \cdot h\nabla w^{\boldsymbol{I}} \qquad \text{in } \Omega, \tag{3.27}$$

$$u + z_l \sigma \frac{\partial v}{\partial \nu} = V_l \qquad \text{on } e_l,\ l = 1, 2, \cdots, L\ , \tag{3.28}$$

$$\int_{e_l} \sigma \frac{\partial v}{\partial \nu}\, dS = 0 \qquad \text{for } l = 1, 2, \cdots, L\ , \tag{3.29}$$

$$\sigma \frac{\partial v}{\partial \nu} = 0 \qquad \text{on } \partial\Omega \backslash \overline{\bigcup_{l=1}^{L} e_l}\ . \tag{3.30}$$

Thus, it is now clear that (3.27)-(3.30) is a special case of (3.15)-(3.18) with the conversion of symbols $u := v,\ \boldsymbol{U} := \boldsymbol{V},\ \boldsymbol{I} := \boldsymbol{0},\ \sigma := \sigma + h$ and $g := w^{\boldsymbol{I}}$. This means that the bound (3.20) for the weak formulation holds given that $\sigma + h$ is appropriately bounded as in Theorem 3. Since $h \in L^\infty(\Omega')$, $\sigma + h$ is bounded from above. The lower bound can be found by assuming $\|h\|_{L^\infty(\Omega')} \le \sigma_0/2$. This yields

$$|\mathrm{Re}\,(h)| \le \frac{\sigma_0}{2} \quad \Rightarrow \quad \mathrm{Re}\,(h) \ge -\frac{\sigma_0}{2}. \tag{3.31}$$

Combining (3.31) with the assumption $\mathrm{Re}\,(\sigma) \ge \sigma_0$ from Theorem 3 yields

$$\mathrm{Re}\,(\sigma + h) = \mathrm{Re}\,(\sigma) + \mathrm{Re}\,(h) \ge \sigma_0 - \frac{\sigma_0}{2} = \frac{\sigma_0}{2},$$

Hence, we have the apropriate bounds for $\sigma + h$. Following the same procedure as in Theorem 3 for the coercivity, we get

$$|B((v, \boldsymbol{V}), (v, \boldsymbol{V}))| \ge \min\left(\frac{\sigma_0}{2}, \frac{Z}{\max\limits_l |z_l|}\right) \|(v, \boldsymbol{V})\|_*^2 = \delta \|(v, \boldsymbol{V})\|_*^2, \tag{3.32}$$

where δ is a constant independent of h. We now combine the lower bound (3.32) and the upper bound (3.26). Since $\|\boldsymbol{I}\|_{\mathbb{C}^L} = 0$ as $\boldsymbol{I} = \boldsymbol{0}$, we get the inequality

$$\delta \|(v, \boldsymbol{V})\|_*^2 \le |B((v, \boldsymbol{V}), (v, \boldsymbol{V}))| = |f((v, \boldsymbol{V}))| \le \|h\|_{L^\infty(\Omega')} \left\|w^{\boldsymbol{I}}\right\|_{H^1(\Omega)} \|(v, \boldsymbol{V})\|_*\,. \tag{3.33}$$

Reordering (3.33), it follows that

$$\|(v, \boldsymbol{V})\|_* \le \frac{1}{\delta} \|h\|_{L^\infty(\Omega')} \left\|w^{\boldsymbol{I}}\right\|_{H^1(\Omega)}. \tag{3.34}$$

Combining (3.9) and (3.14) we see that by (3.34) we have

$$\begin{aligned} |\langle (R_{\sigma+h} - R_\sigma - R'_\sigma[h])\boldsymbol{J}, \boldsymbol{I}\rangle| &= \left| \int_\Omega h\nabla w^{\boldsymbol{I}} \cdot \overline{\nabla v}\, \mathrm{d}x \right| \\ &\le \|h\|_{L^\infty(\Omega')} \left\|\nabla w^{\boldsymbol{I}}\right\|_{L^2(\Omega)} \|\nabla v\|_{L^2(\Omega)} \\ &\le \|h\|_{L^\infty(\Omega')} \left\|\nabla w^{\boldsymbol{I}}\right\|_{L^2(\Omega)} \|(v, \boldsymbol{V})\|_* \\ &\le \left\|\nabla w^{\boldsymbol{I}}\right\|_{L^2(\Omega)} \frac{1}{\delta} \|h\|_{L^\infty(\Omega')}^2 \left\|w^{\boldsymbol{I}}\right\|_{H^1(\Omega)}. \end{aligned}$$

If we now go back to (3.3), with the operator norm (3.4), we have that

$$\lim_{h \to 0} \frac{1}{\|h\|_{L^\infty(\Omega')}} \|R_{\sigma+h} - R_\sigma - R'_\sigma[h]\|_{op} = 0.$$

This proves that (3.9) is the Fréchet derivative. □

Remark 3. *Note that Theorem 6 only applies to small perturbation, which are* 0 *near the boundary. This means that this need to be checked before applying the theorem.*

3.3 Calculating the Fréchet Derivative

In this section we wish to investigate the Fréchet derivative. We know that given $h \in L^\infty(\Omega')$, $R'_\sigma[h]$ maps a current vector into a voltage vector. As we saw for R_σ in Section 3.1 we can represent $R'_\sigma[h]$ as a matrix $\mathbf{R}'_\sigma\mathbf{h}$ with elements

$$[\mathbf{R}'_\sigma\mathbf{h}]_{i,j} = \left\langle R'_\sigma[h]\boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle = -\int_\Omega h\nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x,$$

where $\{\boldsymbol{I}^j\}_{j=1}^{L-1}$ is an orthonormal basis for $\mathbb{C}^L_\diamond$ and w^i is the first entry of the solution to (2.2)-(2.5) with current vector $\boldsymbol{I}^i$.

To calculate the elements of the matrix representation of $R'_\sigma[h]$, we make a pixel-discretization of our domain, Ω, by triangulation, where each triangle is a pixel. We make N pixels and denote them $p_k,\ k = 1, 2, \cdots, N$. Now using the basis functions

$$\phi_k(x,y) = \begin{cases} 1 & ,(x,y) \in p_k, \\ 0 & ,\text{otherwise}, \end{cases}$$

we can write

$$\left\langle R'_\sigma[h]\boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle = -\sum_{k=1}^N \int_\Omega \phi_k h\nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x = -\sum_{k=1}^N \int_{p_k} h\nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x.$$

We now make the approximation that h is constant on each pixel (the more pixels the better the approximation), with value h_k on p_k, so that

$$\left\langle R'_\sigma[h]\boldsymbol{I}^j, \boldsymbol{I}^i \right\rangle \approx -\sum_{k=1}^N \int_{p_k} h_k\nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x = -\sum_{k=1}^N h_k \int_{p_k} \nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x.$$

We see that this can be written as the vector product

$$\mathbf{p}^\top\mathbf{h},$$

where $\mathbf{p} = (-\int_{p_1} \nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x, -\int_{p_2} \nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x, \cdots, -\int_{p_N} \nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x)$ and $\mathbf{h} = (h_1, h_2, \cdots, h_N)$. This means that we can calculate $\mathbf{R}'_\sigma\mathbf{h}$ by calculating the matrix-vector product

$$(L-1)^2 \left\{ \begin{bmatrix} -\int_{p_1} \nabla w^1 \cdot \overline{\nabla w^1}\,\mathrm{d}x & \cdots & -\int_{p_N} \nabla w^1 \cdot \overline{\nabla w^1}\,\mathrm{d}x \\ -\int_{p_1} \nabla w^2 \cdot \overline{\nabla w^1}\,\mathrm{d}x & \ddots & \vdots \\ \vdots & & \\ -\int_{p_1} \nabla w^{(L-1)} \cdot \overline{\nabla w^1}\,\mathrm{d}x & & \\ -\int_{p_1} \nabla w^1 \cdot \overline{\nabla w^2}\,\mathrm{d}x & & \\ \vdots & & \\ -\int_{p_1} \nabla w^{(L-1)} \cdot \overline{\nabla w^{(L-1)}}\,\mathrm{d}x & \cdots & -\int_{p_N} \nabla w^{(L-1)} \cdot \overline{\nabla w^{(L-1)}}\,\mathrm{d}x \end{bmatrix} \right. \left. \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ \\ \\ \\ \\ \vdots \\ h_N \end{bmatrix} \right., \tag{3.35}$$

and then unstacking the resulting vector of length $(L-1)^2$ into a $(L-1)\times(L-1)$ matrix. The matrix in (3.35) is denoted $\mathbf{R}'_\sigma$ and is of great interest to us as we can analyze it to determine some of the depth dependencies in CEM. We will come back to this in the next chapters.

Since we often use many pixels and integrating takes time, we make yet another approximation, namely

$$-\int_{p_k} \nabla w^j \cdot \overline{\nabla w^i}\,\mathrm{d}x \approx -\,|p_k|\,\nabla w^j(m_k)\cdot\overline{\nabla w^i(m_k)}, \tag{3.36}$$

where $|p_k|$ is the area of pixel k and m_k is the midpoint. Here we make the approximation that also $w^i,\ i=1,2,\cdots,L-1$, is constant on each pixel. When we use the approximation (3.36) in calculation of the Fréchet derivative we will use the notation $\mathbf{R}'_\sigma[\mathbf{h}]_{\text{point}}$. When we do not use the approximation we will use the notation $\mathbf{R}'_\sigma[\mathbf{h}]_{\text{int}}$.

Chapter 4

Singular Value Decomposition and Reconstruction

This chapter is meant as a short introduction to Singular Value Decomposition (SVD) and the use of it to reconstruct small perturbations from the conductivity. As inspiration we have used [9].

4.1 Singular Value Decomposition

SVD primarily concerns the equation

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{4.1}$$

where $\mathbf{A} \in \mathbb{R}^{n\times m}$, $\mathbf{x} \in \mathbb{R}^m$, and $\mathbf{b} \in \mathbb{R}^n$. If $\mathbf{A}$ is square and has full rank, it is invertible, and (4.1) is easy to solve for $\mathbf{x}$. On the other hand, if $\mathbf{A}$ is not invertible, tools like SVD are used.

SVD is a method of factorization of a matrix $\mathbf{A}$ into a product of matrices $\mathbf{U}\mathbf{S}\mathbf{V}^\top$ with the goal of finding an approximate solution $\mathbf{x}^\dagger$ to 4.1. The dimensions of the matrices are

$$\mathbf{A}_{m\times n} = \mathbf{U}_{m\times m}\mathbf{S}_{m\times n}\mathbf{V}^\top_{n\times n}.$$

The eigenvectors of $\mathbf{A}\mathbf{A}^\top$ are called the left singular vectors and makes up the columns $\mathbf{u}_i$ of $\mathbf{U}$. The eigenvectors of $\mathbf{A}^\top\mathbf{A}$ are called the right singular vectors and makes up the columns $\mathbf{v}_i$ of $\mathbf{V}$. $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, which means that the respective inverse matrices are $\mathbf{U}^\top$ and $\mathbf{V}^\top$. As such the columns of $\mathbf{U}$ and $\mathbf{V}$ form an orthonormal basis for $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively. $\mathbf{S}$ is a diagonal matrix that consists of the square root of the eigenvalues of both $\mathbf{A}\mathbf{A}^\top$ and $\mathbf{A}^\top\mathbf{A}$, which are called the singular values s_i. The values are sorted such that $s_1 \geq s_2 \geq \cdots \geq s_{\min(m,n)}$. In order to solve (4.1) for $\mathbf{x}$, we introduce the pseudoinverse, $\mathbf{A}^\dagger$, of $\mathbf{A}$. If $\mathbf{A}$ is invertible the pseudoinverse is just the inverse of $\mathbf{A}$. The pseudoinverse is defined mathematically as

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{S}^\dagger\mathbf{U}^\top,$$

where $\mathbf{S}^\dagger_{n\times m}$ is a diagonal matrix with the values $1/s_i$ for all non-zero s_i and zeros for $s_i = 0$. Let n be the index of the smallest non-zero singular value s_n, then the reconstruction $\mathbf{x}^\dagger$ of $\mathbf{x}$ will be

$$\mathbf{x}^\dagger = \mathbf{A}^\dagger\mathbf{b} = \sum_{i=1}^{n} \frac{\langle \mathbf{b}, \mathbf{u}_i \rangle}{s_i}\mathbf{v}_i. \tag{4.2}$$

It is seen from equation (4.3) that the singular values close to zero give a large contribution to the reconstruction. Therefore, when reconstructing perturbations with noise, it can some times be advantageous not to use the smallest singular values in the reconstruction and instead only use the largest k singular values, since the noise coefficients $\langle \mathbf{b}, \mathbf{u}_i \rangle$ are amplified for small singular values s_i. We indicate the use of this k-truncation by using $\mathbf{A}_k^\dagger$ instead of $\mathbf{A}^\dagger$, so that

$$\mathbf{x}_k^\dagger = \mathbf{A}_k^\dagger \mathbf{b} = \sum_{i=1}^{k} \frac{\langle \mathbf{b}, \mathbf{u}_i \rangle}{s_i} \mathbf{v}_i. \tag{4.3}$$

We will not go into details of how to choose this k in an effective way, but only mention that there exist heuristic methods for this. For more information see [5].

4.2 Reconstruction of a Perturbation Function h

Assume that we want to search for a small perturbation h in the conductivity σ. We could for instance be interested in looking for cracks in a block of concrete to make sure that the concrete is stable. We would have knowledge about the "normal" conductivity for concrete, and from that we could find $\mathbf{R}_\sigma$ and $\mathbf{R}'_\sigma$ by simulation. $\mathbf{R}_{\sigma+\mathbf{h}}$ would be calculated from the measured data from EIT. In the end with the help of truncated SVD, we could reconstruct the perturbation and find the cracks. This section is about how to do that in practice.

The equation we would like to solve for $\mathbf{h}$ is

$$\mathbf{R}'_\sigma \mathbf{h} = (\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma), \tag{4.4}$$

where $\mathbf{h}$ is a stacked vector of length N equal to the number of pixels in the mesh with the value of the unknown perturbation on each pixel. Furthermore, $(\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma)$ is stacked as a $(L-1)^2$ vector instead of a matrix. The solution to (4.4) is found using the pseudoinverse of $\mathbf{R}'_\sigma$, that is ${\mathbf{R}'_\sigma}^\dagger = \mathbf{V}\mathbf{S}^\dagger\mathbf{U}^\top$. This means that $\mathbf{h}^\dagger$ is found using (4.3) and is on the form

$$\mathbf{h}^\dagger = \sum_{i=1}^{n} \frac{\langle (\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma), \mathbf{u}_i \rangle}{s_i} \mathbf{v}_i, \tag{4.5}$$

with n being the number of the smallest singular value different from zero. As mentioned in the Section 4.1 a k-truncation could be necessary if there is noise on the measurements. Instead of using an advanced method for finding the k for the k-truncation in our SVD, we calculate the L^2-norm on the difference between the reconstructed and the analytical solution, $\left\|\mathbf{h}^\dagger - \mathbf{h}\right\|_2$, every time we have added one singular value and choose the k that gives the smallest L^2-norm. In this way we choose the reconstruction that are closest to the analytical solution.

4.2.1 Adding Noise

When we have data from actual measurements it will contain some noise, i.e. the right side in (4.1) becomes

$$\mathbf{b}^{\text{noise}} = (\mathbf{R}^{\text{noise}}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma),$$

This means that in the reconstruction formula (4.5) the noise will also be reconstructed and especially for smaller singular values the noise will interfere a lot.

In practice the noise will appear on the measured voltages. So before we calculate $\mathbf{R}_{\sigma+\mathbf{h}}$ by (3.1) we add noise to $R_{\sigma+h}\boldsymbol{I}^j,\ j = 1, 2, \cdots, L-1$. So we have

$$R_{\sigma+h}\boldsymbol{I}^j + \mathbf{n}.$$

In this project we have chosen to work with normal distributed noise with mean 0 and a standard deviation depending on the data. We calculate $\mathbf{n}$ by

$$\mathbf{n} \in \mathcal{N}_L(\mathbf{0}, sd^2), \quad sd = \epsilon \max_{j=1,2,\cdots,L-1} \max_{i=1,2,\cdots,L} \left[R_{\sigma+h}\boldsymbol{I}^j\right]_i,$$

where $\left[R_{\sigma+h}\boldsymbol{I}^j\right]_i$ is the i'th element of the j'th voltage vector. We call ϵ the noise level. To determine the noise level we calculate $\mathbf{b}$ and $\mathbf{b}^{\text{noise}}$ for different noise levels and look at the relative error

$$e^{\text{noise}} = \frac{\left\|\mathbf{b}^{\text{noise}} - \mathbf{b}\right\|_2}{\|\mathbf{b}\|_2} = \frac{\left\|(\mathbf{R}^{\text{noise}}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma) - (\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma)\right\|_2}{\|(\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma)\|_2}. \tag{4.6}$$

We aim to use a relative error close to 1%.

Chapter 5

Numerical Analysis of the CEM

5.1 Setup for Numerical Analysis

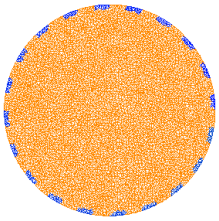
In general all numerical calculations in this chapter are made using the programming language Python [10]. We use FEniCS [7] as a library in order to use FEM to solve the forward problem. The unit disk is chosen as our domain, Ω. The boundary of the circle is approximated by a 300th polygon and the coarseness of the mesh is calculated to $N = 9920$ pixels. The contact-impedances are set to $z_i = 0.1,\ i = 1, 2, ..., L$ and the conductivity is chosen to $\sigma = 1$. In order to calculate for instance $\mathbf{R}_\sigma$, we need an orthonormal basis $\{I_k\}_{k=1}^{L-1}$ for $\mathbb{C}_\diamond^L$. This orthonormal basis is calculated by the Gram-Schmidt process with the starting vector $(1, -1, 0, 0, \cdots, 0)$. In addition to this setup, we operate with 3 different electrode configurations all with uniform spacing. The full electrode configuration is $L = 20$ electrodes attached to the whole boundary, see Figure 5.1a. The half electrode configuration is still $L = 20$ electrodes, but instead spread over only the top of the boundary, see Figure 5.1b. The quarter electrode configuration is $L = 10$ electrodes spread over the top right corner of the boundary, see Figure 5.1c. If nothing else is stated we use perturbations, h, with amplitude 0.3.

5.2 Test of the Fréchet Derivative

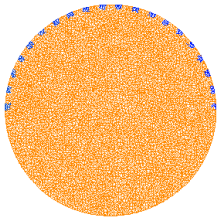
In order to test the Fréchet derivative, we do two numerical experiments. First we check if $\mathbf{R}'_\sigma$ satisfies (3.3). We let the test situation be as in 5.1 and use the full electrode configuration. We make a convergence test with a specific type of function $h_i \in L^\infty(\Omega')$

$$h_i(x, y) = \begin{cases} i & , x^2 + y^2 \leq (\frac{1}{3})^2 \\ 0 & , \text{otherwise,} \end{cases} \tag{5.1}$$

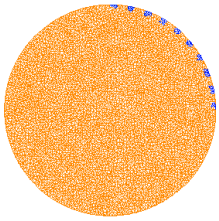
for $i < \frac{1}{2}$. This type of function takes the value i inside a circle with radius $r = \frac{1}{3}$ and 0 elsewhere. By calculating $\|\mathbf{R}_{\sigma+\mathbf{h_i}} - \mathbf{R}_\sigma - \mathbf{R}'_\sigma \mathbf{h_i}\|_{op}$ and $\|h_i\|_{L^\infty(\Omega')} = i$ for chosen i-values, a convergence plot is created.



(a) The full electrode configuration.



(b) The half electrode configuration.



(c) The quarter electrode configuration.

Figure 5.1: *The 3 different electrode configurations we use in the project. The blue color is where the electrodes are places.*

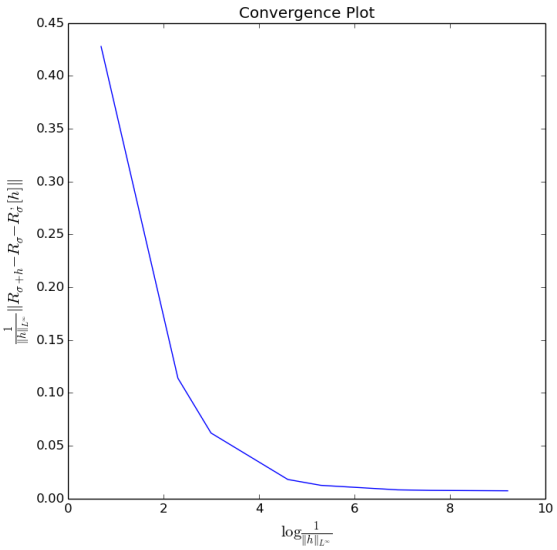


Figure 5.2: *Convergence plot of* $\frac{\|\mathbf{R}_{\sigma+\mathbf{h_i}}-\mathbf{R}_\sigma-\mathbf{R}'_\sigma\mathbf{h_i}\|_{op}}{\|h_i\|_{L^\infty(\Omega')}}$ *as a function of* $\log(\frac{1}{\|h_i\|_{L^\infty(\Omega')}})$.

From Figure 5.2 it is observed that $\|\mathbf{R}_{\sigma+\mathbf{h_i}} - \mathbf{R}_\sigma - \mathbf{R}'_\sigma\mathbf{h_i}\|_{op}$ goes to zero as $\log(\frac{1}{\|h_i\|_{L^\infty(\Omega')}})$ gets larger which corresponds to $\|h_i\|_{L^\infty(\Omega')}$ going to zero. The reason we use $\log(\frac{1}{\|h_i\|_{L^\infty(\Omega')}})$

and not $\|h_i\|_{L^\infty(\Omega')}$ is that we use i-values which spand from $i = 0.001$ to $i = 0.5$. This means that (3.3) is satisfied for this numerical example. We calulate the operator norm as the largest eigenvalue of the matrix.

The second numerical experiment is to observe the matrices $\mathbf{R}'_\sigma[\mathbf{h}]$ and $(\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma)$ and the absolute difference between these.

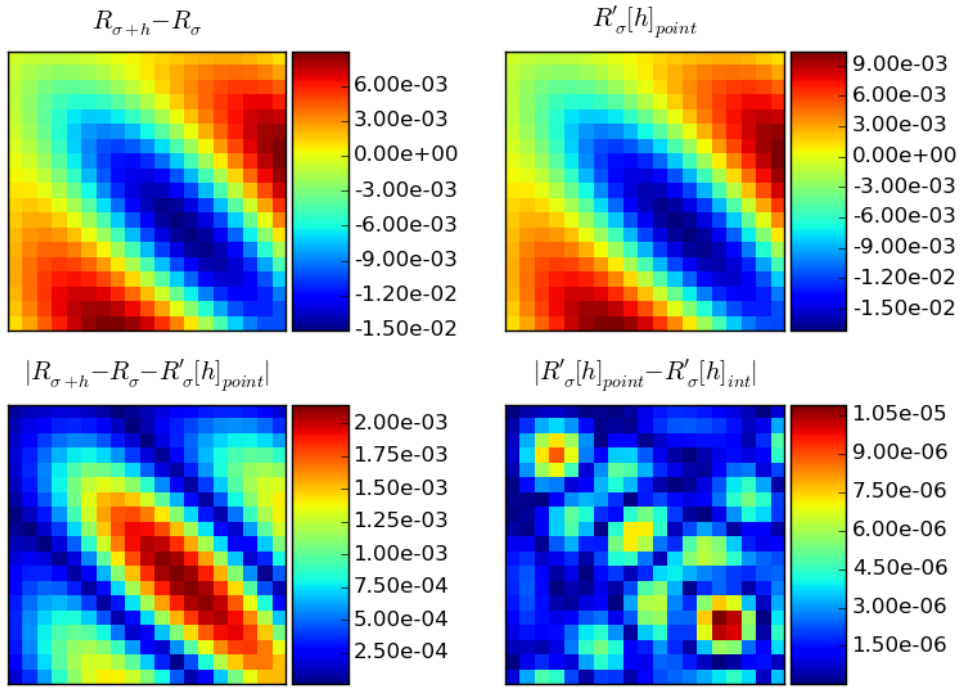


Figure 5.3: *Plots to test the linearization. In these plots we have used a mesh with $N = 3694$ pixels.* ***Top left:*** *the actual difference $\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma$.* ***Top right:*** *the linear approximation using* (3.36)*.* ***Bottom left:*** *the absolute difference to the linear approximation.* ***Bottom right:*** *the absolute difference when using the approximation* (3.36) *compared to not using it.*

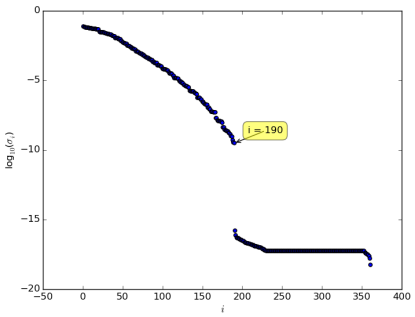
We observe in Figure 5.3 that the error we make using the linearization is roughly a tenth of the largest value in the matrices. This is close to zero but still a large error compared to the values in the matrix. We see that the tendencies in the matrices are the same but that the values are a little off. In the bottom right we see that the error we make using the approximation (3.36) is quite small, and will be smaller as we usually use $N = 9920$ pixels. Here we note that calculating the Fréchet derivative with the integrals takes hours with our code, while it takes minutes with the approximation.

5.3 Singular Vectors and Depth Dependency

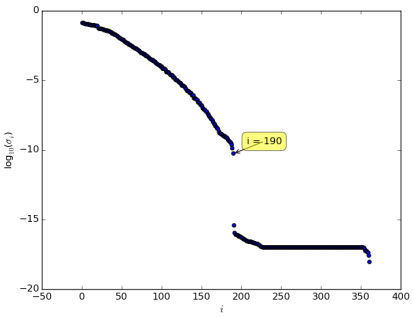
As we have seen in (4.5) the singular values and vectors play a big role in reconstruction. We know that $\mathbf{V}$ is a orthogonal $N \times N$ matrix and therefore that its columns form a basis for the "pixel-space". Therefore we wish to investigate the right singular vectors. Specifically we are interested in the size of the corresponding singular value for each vector since this will tell us in which areas of the domain a perturbation is harder to reconstruct. A perturbation in areas with contribution of vectors with larger singular values is easier

to reconstruct since noise will not have that much of an effect. Likewise a perturbation in areas with contribution of vectors with small singular values is hard to reconstruct.

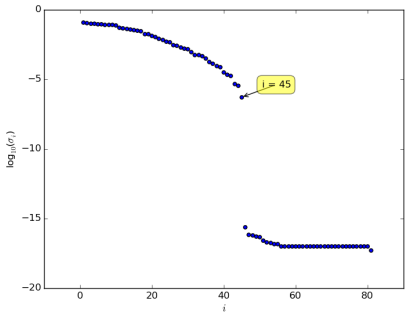
We have plotted the singular values of the setup described in Section 5.1 with the different electrode configuration on a logarithmic scale in Figure 5.4. We notice that for the full and half configurations, roughly there is an exponential decrease in the values until the 190th singular value, where the values go to machine precision which we interpret as 0. We see the same decrease for the quarter configuration but until the 45th singular value. This means that in reconstruction we use $n = 190$ values at most.



(a) Full. $L = 20$.



(b) Half. $L = 20$.

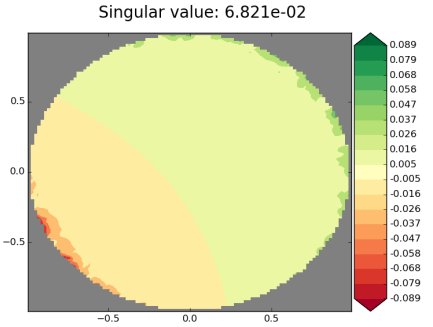


(c) Quarter. $L = 10$.

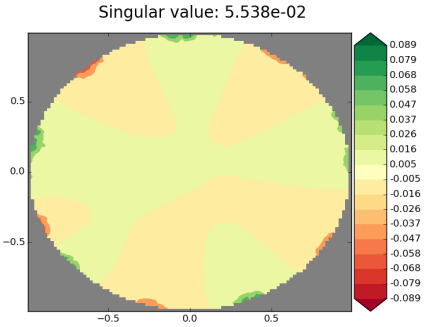
Figure 5.4: *Singular values for the three electrode configurations. We have marked the point where the singular values goes to machine precision (equivalent to 0).*

In Figure 5.5 we have plotted the singular vectors for some singular values. We see in Figures 5.5a and 5.5b that the vectors for larger singular values are located very close to the boundary. In Figures 5.5c and 5.5d we see that the vectors for the smaller singular values are well spread over the domain. In Figures 5.5e and 5.5f we see that the vectors for the even smaller values are primarily located close to the middle of the domain. For singular values equal to 0 we have spikes around the domain, which seem arbitrary as seen in Figures 5.5g and 5.5h.

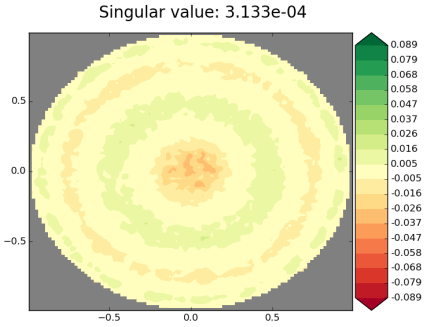
To sum up the previous paragraph we see a pattern where the vectors move towards the middle, and thereby further away from the electrodes, when the singular values decrease. This shows that there is a depth dependency in the CEM, i.e. it is easier to reconstruct perturbations close to the electrodes.



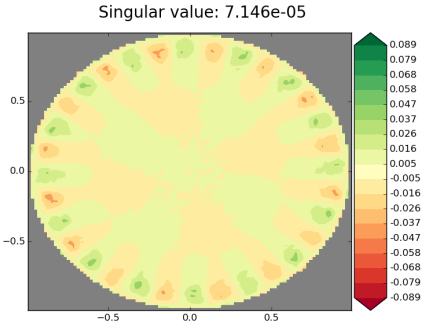
(a) 2nd singular vector.



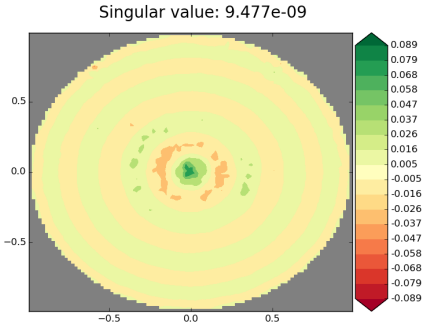
(b) 10th singular vector.



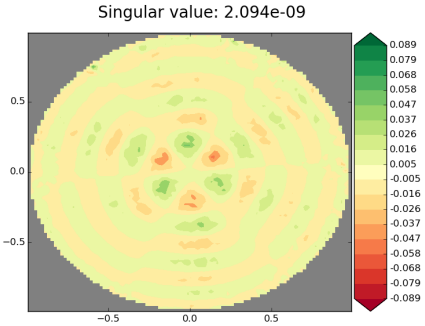
(c) 85th singular vector.



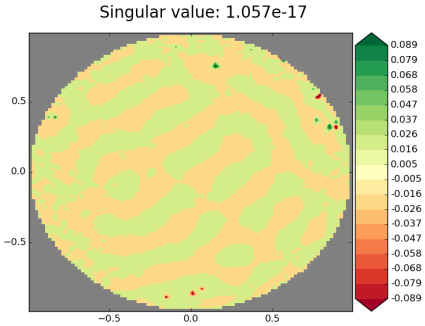
(d) 100th singular vector.



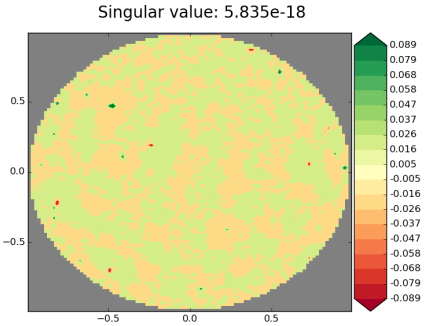
(e) 175th singular vector.



(f) 182nd singular vector



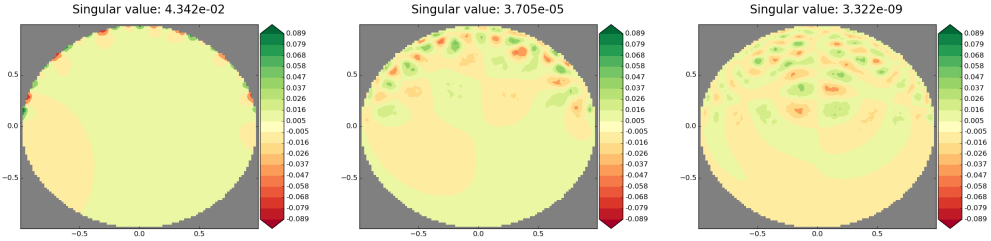
(g) 220th singular vector.



(h) 260th singular vector.

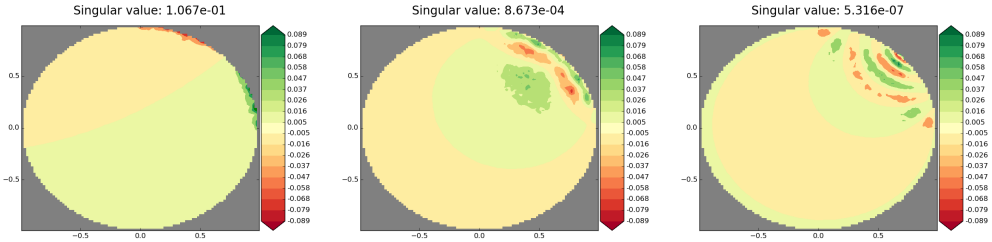
Figure 5.5: *Plots of some chosen singular vectors. In the first row we have large singular values and in the third row we have small values while the second row has values somewhere in between. In the fourth row we see the vectors for singular values that are 0.*

If we look at the singular vectors for the half and quarter configurations in Figures 5.6 and 5.7 respectively, we see the same tendencies as for the full configuration. That is, the larger the singular values the closer the contribution moves towards the electrodes. Furthermore, we notice that there are no contributions in the bottom half for the half configuration and no contributions outside of the top right quarter for the quarter configuration. This means that reconstruction of perturbation in these areas is not possible with these configurations. For more singular vectors for the half and quarter electrode configurations see Appendix C.1.



(a) *28th singular vector.* ***(b)*** *109th singular vector.* ***(c)*** *171st singular vector.*

Figure 5.6: *Plots of some chosen singular vectors for the half electrode configuration.*

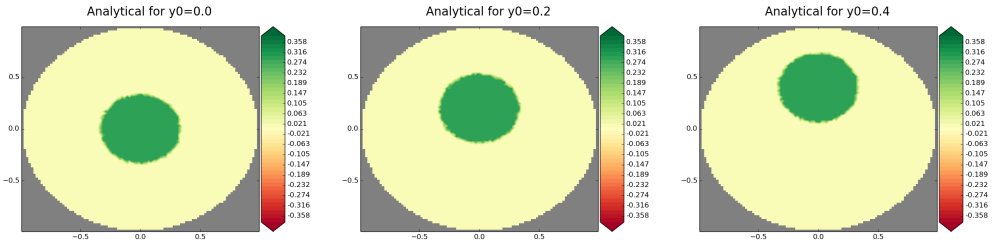


(a) *2nd singular vector.* ***(b)*** *31st singular vector.* ***(c)*** *45th singular vector.*

Figure 5.7: *Plots of some chosen singular vectors for the quarter electrode configuration.*

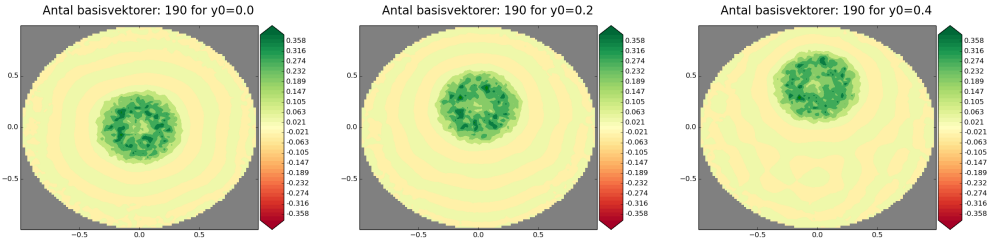
5.4 Depth Dependency in Reconstruction

We want to test the depth dependency of the reconstruction for the 3 different electrode configurations. The way we approach this is to create a symmetric object with its center coordinates (x_0, y_0) moved towards the electrodes. Appropriate step intervals are chosen, the reconstructions are plotted and the relative error $\left\|\mathbf{h} - \mathbf{h}^\dagger\right\|_2 / \|\mathbf{h}\|_2$ between the analytical solution, $\mathbf{h}$, and the reconstructed solution, $\mathbf{h}^\dagger$ is calculated in every step. The code can be found in Appendix D.3.



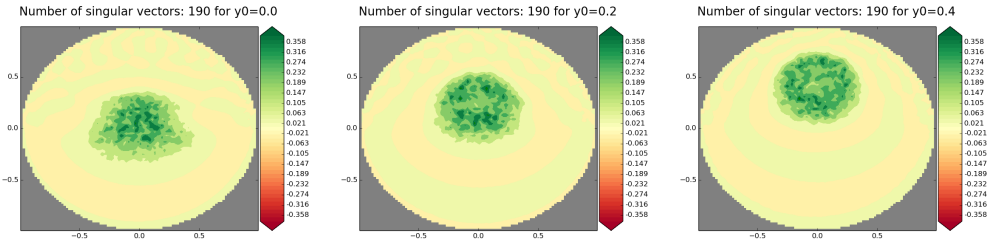
(a) y0=0. *(b) y0=0.20.* *(c) y0=0.40.*

Figure 5.8: *The analytical circle function for given y_0 and $x_0 = 0$.*



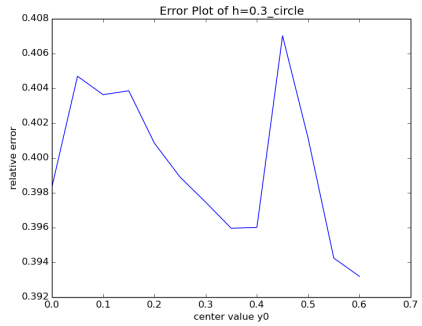
(a) y0=0. *(b) y0=0.20.* *(c) y0=0.40.*

Figure 5.9: *The reconstructed circle function with the full electrode configuration for given y_0 and $x_0 = 0$.*

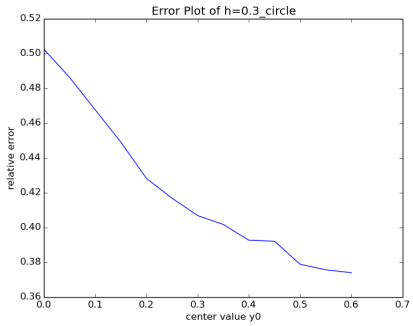


(a) y0=0. *(b) y0=0.20.* *(c) y0=0.40.*

Figure 5.10: *The reconstructed circle function with the half electrode configuration for given y_0 and $x_0 = 0$.*



(a) *relative error as a function of y_0. Tested on the circle function with the full electrode configuration.*

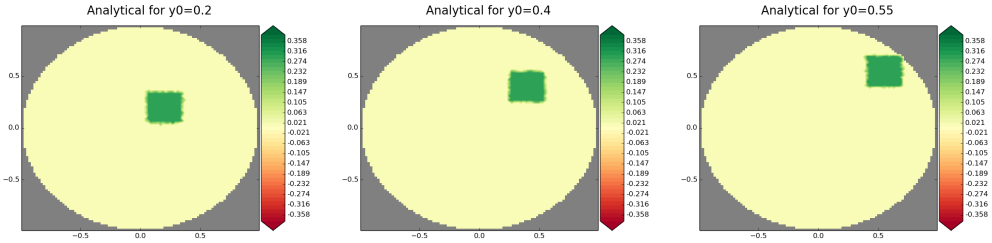


(b) *relative error as a function of y_0. Tested on the circle function with the half electrode configuration.*

Figure 5.11: *Relative errors for moved objects.*

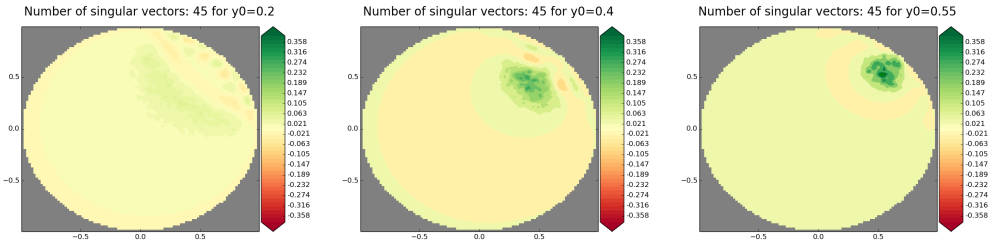
We do a test that compares the reconstruction of the circle function with the full and the half electrode configurations. It is observed that the reconstruction is quite good in all the steps in the full configuration, whereas for the half configuration the reconstruction gets better the closer the perturbation is on the electrodes. This is also confirmed quantitatively with the two plots of the relative error, since Figure C.7a shows no clear tendencies in the relative error as we get closer to the boundary and Figure C.7b shows that the relative error decreases.

For the quarter electrode configuration we investigate the depth dependency in the same way, but with a square function. These plots, see Figure 5.13, show the importance of the perturbation being really close to the boundary if one would like to find it using the quarter electrode configuration. The visual observations are supported by Figure 5.14, where the relative error decreases drastically from nearly 100% to around 40% when we get closer to the boundary. For more examples on reconstructions with the half and quarter electrode configurations, see Appendix C.2.



(a) *x0=y0=0.20.* ***(b)*** *x0=y0=0.40.* ***(c)*** *x0=y0=0.55.*

Figure 5.12: *The analytical square function for given (x_0, y_0).*



(a) *x0=y0=0.20.* *(b)* *x0=y0=0.40.* *(c)* *x0=y0=0.55.*

Figure 5.13: *The reconstructed square function with the quarter electrode configuration for given* (x_0, y_0).

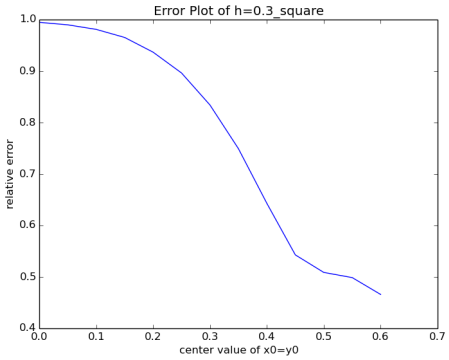


Figure 5.14: *relative error as a function of* y_0. *Tested on the square function with the quarter electrode configuration.*

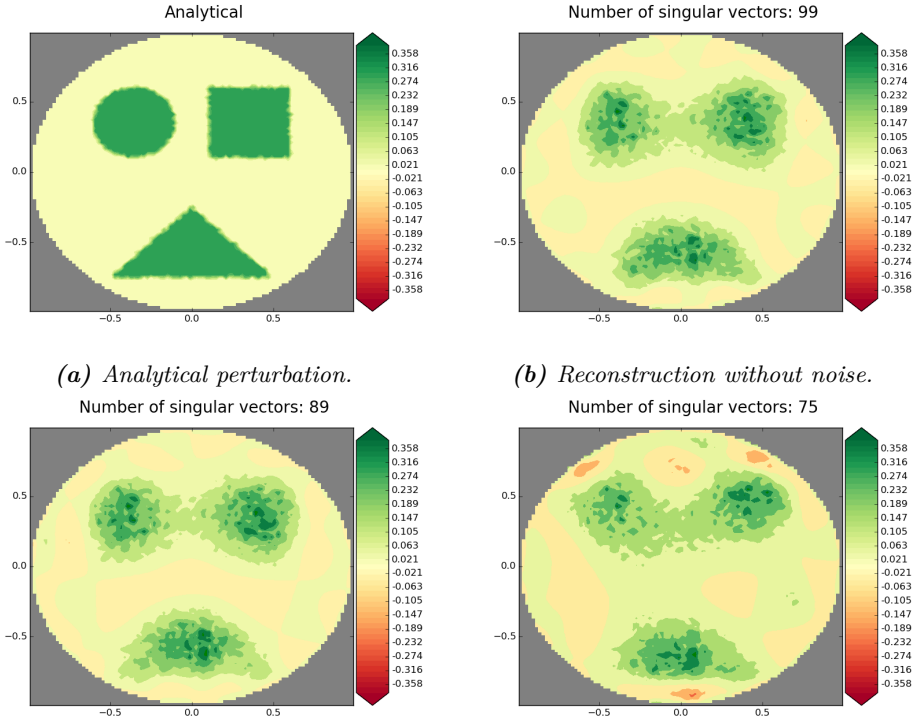
5.5 Testing Reconstructions with Noise

In this section we will test reconstructions with different levels of noise as well as looking at the linearization error. The relative error for noise will be calculated as in (4.6) and the error of linearization by

$$e^{\text{lin}} = \frac{\|\mathbf{b}_{\text{lin}} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2} = \frac{\|\mathbf{R}'_\sigma \mathbf{h} - (\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma)\|_2}{\|(\mathbf{R}_{\sigma+\mathbf{h}} - \mathbf{R}_\sigma)\|_2}$$

.

In Figure 5.15 we have plotted a reconstruction with the full electrode configuration of the analytical function seen in 5.15a for different noise levels, ϵ. We see in Figure 5.15b that the best reconstruction without noise is with 99 singular vectors, but still the shapes are not reconstructed well. When we add some noise with noise level $\epsilon = 10^{-4}$ we get the best reconstruction, seen in Figure 5.15c, with 89 singular vectors. With a noise level of $\epsilon = 10^{-3}$ we get the reconstruction in Figure 5.15d with 75 singular vectors. We see the tendency that the more noise the smaller truncation level, k. Another thing we notice is that the noise does not really affect the reconstruction much at the shown noise levels compared to the error of the linearization, which we have calculated to be 16.78% for this perturbation with the full electrode configuration.



(a) *Analytical perturbation.*

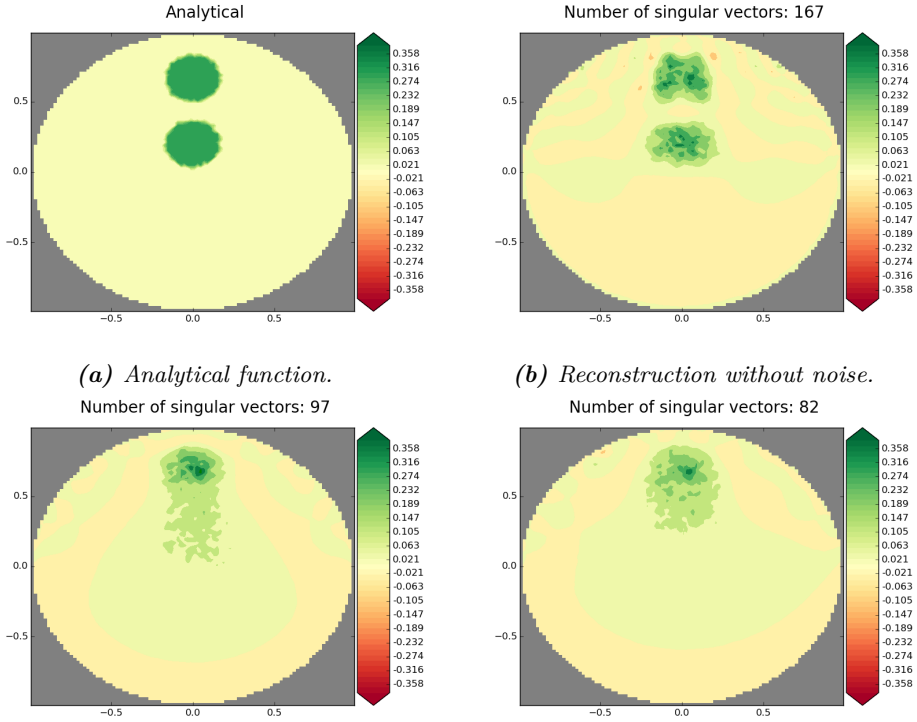
(b) *Reconstruction without noise.*

(c) *Reconstruction with noise level* $\epsilon = 10^{-4}$. $e^{noise} = 0.66\%$.

(d) *Reconstruction with noise level* $\epsilon = 10^{-3}$. $e^{noise} = 6.96\%$.

Figure 5.15: *Reconstructions with different noise levels for the full electrode configuration.*

If we instead use the half electrode configuration for reconstruction of two circles we get the plots in Figure 5.16. One circle is close to the center of the disc, and the other is close to the boundary, where the electrodes are placed. We see in Figure 5.16a that without noise both circles get reconstructed apart from each other. When we add a little noise in Figure 5.16c we see that in the reconstruction the circles are no longer separate and only the circle close to the boundary is about the right amplitude. If we add even more noise in Figure 5.16d the circle close to the center is not in the reconstruction.



(a) *Analytical function.*

(b) *Reconstruction without noise.*

(c) *Reconstruction with noise level* $\epsilon = 10^{-5}$. $e^{noise} = 0.26\%$.

(d) *Reconstruction with noise level* $\epsilon = 10^{-4}$. $e^{noise} = 2.63\%$.

Figure 5.16: *Reconstructions with different noise levels for the half electrode configuration.*

Chapter 6

Conclusion

in Chapter 2 we have shown that there exists a unique solution to the PDE problem that governs EIT using CEM. The fact that unique solutions exist, makes it possible to solve the forward problem as explained in Section 3.1. We have introduced the Fréchet derivative and proved that we can find it for an appropriate mapping in Section 3.2 and used it to linearize the inverse problem. We have tested that the linearization is reasonable in Section 5.2. We have introduced SVD in Section 4.1 and used it to analyze an appropriate matrix representation of the Fréchet derivative in Section 4.5 with the purpose of reconstructing objects.

In order to investigate depth dependency in EIT with the CEM, we have tested the unit disc in three ways. The results are described in Section 5.3 and show that noise on the data will make reconstruction of objects further away from the electrodes harder to reconstruct.

Furthermore, in Section 5.4, we have tested the depth dependency of reconstructions of small perturbations, where we moved the objects closer to the electrodes. The result was that the reconstruction improved significantly when the object moved closer to the electrodes. In this way we can conclude that the distance between the perturbations and the electrodes has great impact on the reconstruction.

Finally, In Section 5.5, we have tested how noise on data affects the reconstruction of objects in practise. We have seen that the linearization of the inverse problem affects the reconstruction more than the noise for small levels of noise and we have found that noise affects objects further away from the electrodes more.

This all builds to the conclusion that the distance to the electrodes has a great influence in reconstruction and that there is a depth depencency in EIT with the CEM.

6.1 Future Work

In our project we have chosen both to go into details with the theory behind the reconstruction of perturbations and to investigate the depth dependency for specific choices of domain and electrode configuration. To get an even more comprehensive picture of depth dependency in EIT with CEM, one could do further testing with different domains, different electrode configurations, and also change the dimension from 2D to 3D.

Appendix A

Definitions, Theorems and Lemmas

Definitions and theorems are inspired by [1] and [3].

Theorem A.1 (Lax-Milgram)**.** *Let X be a Hilbert space. Let B be a complex-valued functional defined on the product space $X \times X$ satisfying the following conditions:*

(a) sesquilinearity:

$$B(\alpha_1 a_1 + \alpha_2 a_2, b) = \alpha_1 B(a_1, b) + \alpha_2 B(a_2, b),$$
$$B(a, \beta_1 b_1 + \beta_2 b_2) = \bar{\beta}_1 B(a, b_1) + \bar{\beta}_2 B(a, b_2);$$

(b) boundedness: $\exists \gamma > 0 : |B(a,b)| \leq \gamma \|a\| \|b\|$*;*

(c) coercivity: $\exists \delta > 0 : |B(a,a)| \geq \delta \|a\|^2$*.*

Then, for any continuous linear functional $f : X \to \mathbb{C}$, there is a unique $b \in X$ such that $f(a) = B(a,b), \quad \forall a \in X$.

Theorem A.2 (Trace Theorem)**.** *Assume Ω is bounded and $\partial\Omega$ is C^1. Then there exists a bounded linear operator*

$$T : W^{1,p}(\Omega) \to L^p(\partial\Omega)$$

such that

$$Tu = u|_{\partial\Omega} \quad \text{if} \quad u \in W^{1,p}(\Omega) \cap C(\overline{\Omega}) \tag{i}$$

and

$$\|Tu\|_{L^p(\partial\Omega)} \leq C \|u\|_{W^{1,p}(\Omega)}, \tag{ii}$$

for each $u \in W^{1,p}(\Omega)$, with the constant C depending on p and Ω only.

A.1 Sobolev Imbeddings for Bounded Domains

Definition A.3 (Sobolev Spaces)**.** *Fix $p \in [1, \infty[$ and let $k \in \mathbb{N}$. Then we define*

$$W^{k,p}(\Omega),$$

as the functionspace consisting of local summable functions, i.e. functions $u \in L^1(\Omega)$ where $u : \Omega \to \mathbb{R}$ and $D^\alpha u$ exists in the weak sense and belongs to $L^p(\Omega)$ for all α that satisfy $|\alpha| \leq k$.

Definition A.4 (Continuous and Compact Imbeddings)**.** *Given normed spaces* $(X, \|\cdot\|_X)$ *and* $(Y, \|\cdot\|_Y)$ *such that* $X \subseteq Y$*, define the* inclusion operator $\iota : X \to Y$ *by* $\iota x = x,\ x \in X$*.*

(i) A continuous imbedding*, written* $X \hookrightarrow Y$*, is when* ι *is continuous or equivalently (as* ι *is linear) bounded*

$$\exists C > 0\, \forall x \in X : \|x\|_Y \leq C \|x\|_X .$$

(ii) A compact imbedding*, written* $X \subset\subset Y$*, is when* $X \hookrightarrow Y$ *and* ι *is compact. I.e. for every bounded sequence* $(x_n) \subset X$ *with* $\sup_n \|x_n\|_X < \infty$*, there exists a subsequence* $(x_{n_k}) \subseteq (x_n)$ *which is Cauchy in* $(Y, \|\cdot\|_Y)$*.*

Lemma A.5. *Let* Ω *be an open bounded domain, then* $W^{m,p}(\Omega) \hookrightarrow W^{m,q}(\Omega)$ *for* $m \in \mathbb{N}_0$ *and* $1 \leq q \leq p \leq \infty$*.*

Theorem A.6 (Sobolev Imbedding)**.** *Let* Ω *be an open bounded domain in* $\mathbb{R}^d$ *with* C^1 *boundary. Furthermore, let* $1 \leq p < \infty$*,* $k \in \mathbb{N}$ *and* $m \in \mathbb{N}_0$*, then the following continuous imbeddings apply.*

(i) If $k < \frac{d}{p}$ *then*

$$W^{m+k,p}(\Omega) \hookrightarrow W^{m,q}(\Omega),\ \ 1 \leq q \leq dp/(d-kp).$$

(ii) If $k > \frac{d}{p}$ *or if* $k = d$ *and* $p = 1$ *then*

$$W^{m+k,p}(\Omega) \hookrightarrow W^{m,q}(\Omega),\ \ 1 \leq q \leq \infty.$$

(iii) If $k = \frac{d}{p}$ *then*

$$W^{m+k,p}(\Omega) \hookrightarrow W^{m,q}(\Omega),\ \ 1 \leq q < \infty.$$

Theorem A.7 (Rellich-Kondrachov)**.** *Let* Ω *be an open bounded domain in* $\mathbb{R}^d$ *with* C^1 *boundary. Furthermore, let* $1 \leq p < \infty$*,* $k \in \mathbb{N}$ *and* $m \in \mathbb{N}_0$*, then the following compact imbeddings apply.*

(i) If $k < \frac{d}{p}$ *then*

$$W^{m+k,p}(\Omega) \subset\subset W^{m,q}(\Omega),\ \ 1 \leq q < dp/(d-kp).$$

(ii) If $k \geq \frac{d}{p}$ *then*

$$W^{m+k,p}(\Omega) \subset\subset W^{m,q}(\Omega),\ \ 1 \leq q < \infty.$$

Remark 4. *Note that* $L^p(\Omega) = W^{0,p}(\Omega)$*, and* $H^k(\Omega) = W^{k,2}(\Omega)$*.*

A.2 The Fréchet Derivative

This section is inspired by [4].

The Fréchet derivative of an operator can be compared to the derivative of a function in the sense that it describes what happens when we take a point and move a little away from it. Before defining the Fréchet derivative we need to introduce some notation.

Definition A.8. *Suppose* $f : X \to M$ *is defined on a neighbourhood of* $0 \in X \subset N$*. We say* $f(h) = o(h)$ *(read '*$f(h)$ *is little oh of* h*') if* $\|f(h)\| / \|h\| \to 0$ *as* $h \to 0$*.*

The letter o stands for order of magnitude, and $f = o(h)$ means that f is of a smaller order of magnitude than h. This leads to the main definition of this section.

Definition A.9. *A continuous linear operator $L : N \to M$ is said to be the* ***Fréchet derivative*** *of $f : X \subset N \to M$ at the point $x \in X$ if*

$$f(x + h) = f(x) + Lh + o(h) \quad \text{as} \quad h \to 0,$$

where N and M are normed spaces.

Appendix B

Analytical Solution to CEM

The problem is the same as in Chapter 2.3. We will solve the problem analytically using seperation of variables. Assume therefore that

$$u(x, y) = X(x)Y(y). \tag{B.1}$$

We have then, that

$$\nabla u = 0 \quad \Leftrightarrow \quad X''Y + Y''X = 0 \quad \Leftrightarrow \quad \frac{X''}{X} + \frac{Y''}{Y} = 0. \tag{B.2}$$

Since $\frac{X''}{X}$ is independent of y and $\frac{Y''}{Y}$ is independent of x, they are both constant. Let the constant be denoted by λ, then we have

$$\frac{X''}{X} = -\frac{Y''}{Y} = \lambda, \tag{B.3}$$

For X we get the equation

$$X'' - X\lambda = 0. \tag{B.4}$$

For $\lambda = 0$ we get the linear solutions $X_0(x) = C_0 + C_1 x$. For $\lambda \neq 0$ we get the solutions

$$X_n(x) = C_{1,n}e^{\sqrt{\lambda}x} + C_{2,n}e^{-\sqrt{\lambda}x} \tag{B.5}$$

For Y we get the equation

$$Y'' + Y\lambda = 0, \tag{B.6}$$

which for $\lambda = 0$, too, gives us linear solutions $Y_0(y) = K_0 + K_1 y$. Imposing the boundary condition $Y'(0) = 0$ show that $K_1 = 0$ so that we have the constant solution $Y_0(y) = K_0$. For $\lambda \neq 0$ the solutions become

$$Y_n(y) = A_n \cos(\sqrt{\lambda}y) + B_n \sin(\sqrt{\lambda}y) \tag{B.7}$$

Now we want to impose the boundary conditions on Y. We have

$$Y_n'(y) = -A_n\sqrt{\lambda}\sin(\sqrt{\lambda}y) + B_n\sqrt{\lambda}\cos(\sqrt{\lambda}y) \tag{B.8}$$

Using $Y'(0) = 0$ yields $B_n = 0$, since $\lambda \neq 0$. Using $Y'(1) = 0$ we get

$$Y_n'(1) = -A_n\sqrt{\lambda}\sin(\sqrt{\lambda}) = 0,$$

resulting in $\lambda = n^2\pi^2, \quad n \in \mathbb{N}$. The solution $u(x,y)$ will then be

$$\begin{aligned} u(x,y) &= X_0(x)Y_0(y) + \sum_{n=1}^{\infty} X_n(x)Y_n(y) \\ &= D_0 + D_1 x \\ &+ \sum_{n=1}^{\infty} (A_n \cos(n\pi y))(C_{1,n}e^{n\pi x} + C_{2,n}e^{-n\pi x}) \end{aligned}$$

We calculate $u_x(x,y)$ as

$$u_x(x,y) = D_1 + \sum_{n=1}^{\infty} A_n \cos(n\pi y)(C_{1,n}n\pi e^{n\pi x} - C_{2,n}n\pi e^{-n\pi x}).$$

Using the boundary conditions $-\int_0^1 u_x(0,y) = I$ we get the coefficients

$$D_1 = -I, \; A_n = 0, \; n \in \mathbb{N}.$$

To get the remaining coefficients we sum the boundary conditions yielding

$$\begin{aligned} u(0,y) - zu_x(0,y) + (u(1,y) + zu_x(1,y)) = U - U = 0 \quad &\Leftrightarrow \\ D_0 - zD_1 + (D_0 + D_1 + zD_1) = 0 \quad &\Leftrightarrow \\ D_0 = \frac{-D_1}{2} = \frac{I}{2}. \end{aligned}$$

The final solution is therefore

$$u(x,y) = \frac{I}{2} - Ix. \tag{B.9}$$

Notice that the solution is independent of y and z.

In Figure B.1 we see a plot of the solution for $I = 2$.

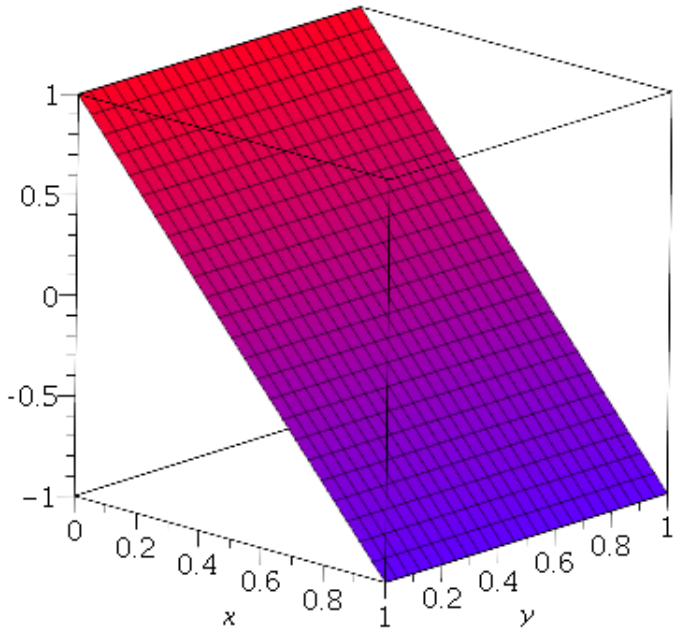
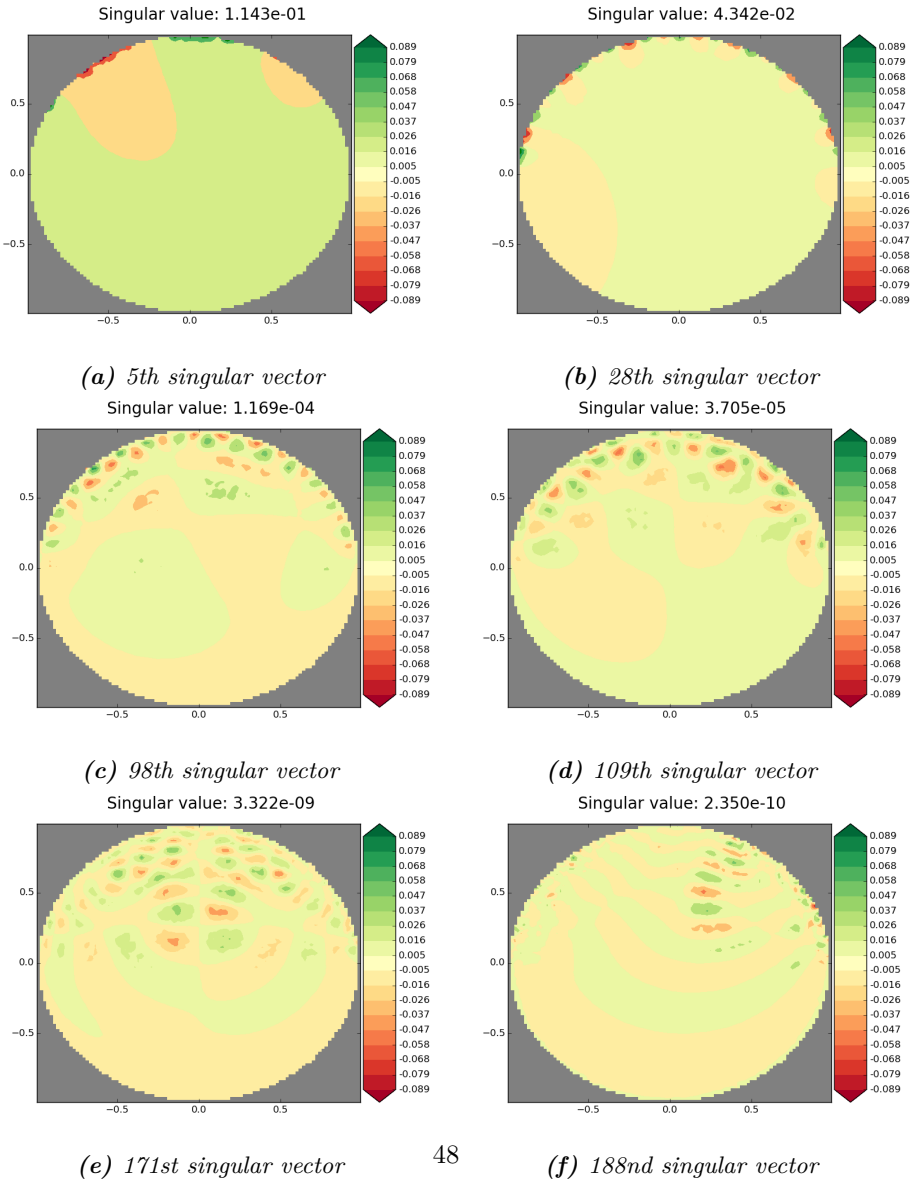


Figure B.1: *Plot of solution in squaredomain with I=2*

Appendix C

Plots

C.1 Singular Vectors



(a) 5th singular vector

(b) 28th singular vector

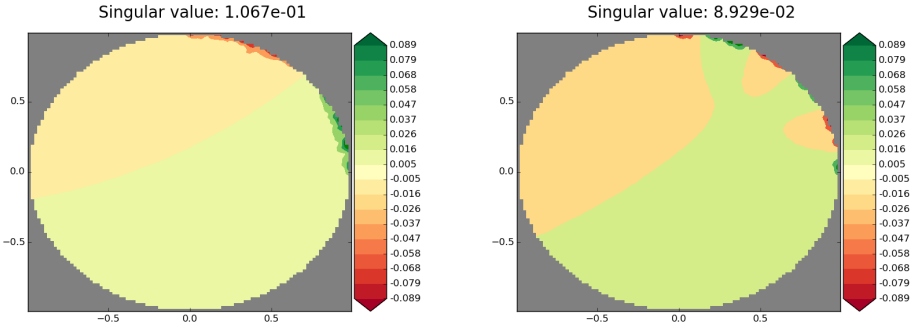
(c) 98th singular vector

(d) 109th singular vector

(e) 171st singular vector

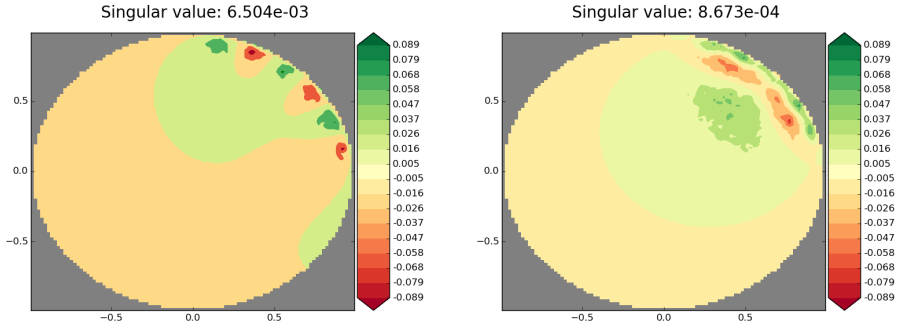
(f) 188nd singular vector

Figure C.1: *Plots of some chosen singular vectors for the half electrode configuration.*



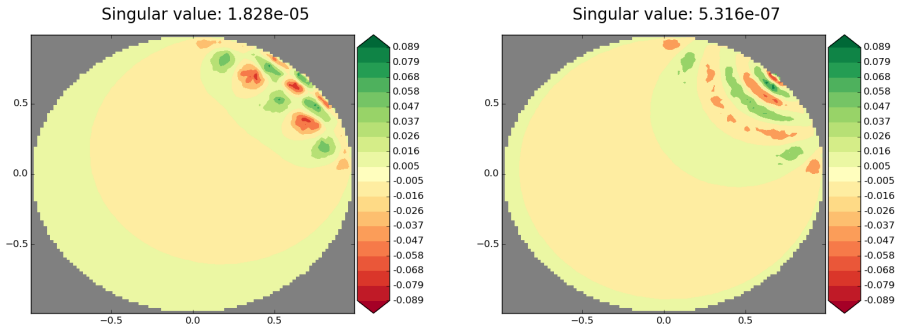
(a) *2nd singular vector*

(b) *6th singular vector*



(c) *23rd singular vector*

(d) *31st singular vector*

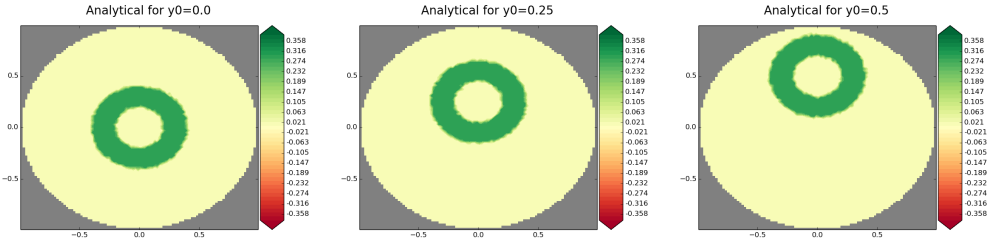


(e) *42nd singular vector*

(f) *45th singular vector*

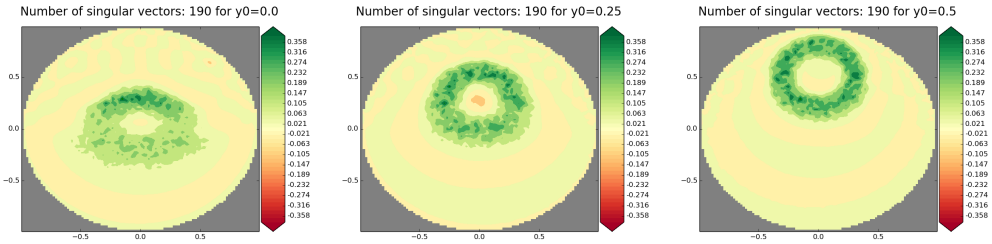
Figure C.2: *Plots of some chosen singular vectors for the quarter electrode configuration.*

C.2 Reconstructions



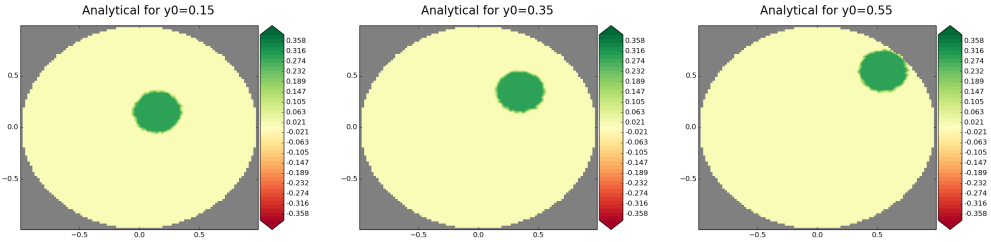
(a) y0=0 *(b) y0=0.25* *(c) y0=0.50*

Figure C.3: *The analytical hole function for given y_0 and $x_0 = 0$*



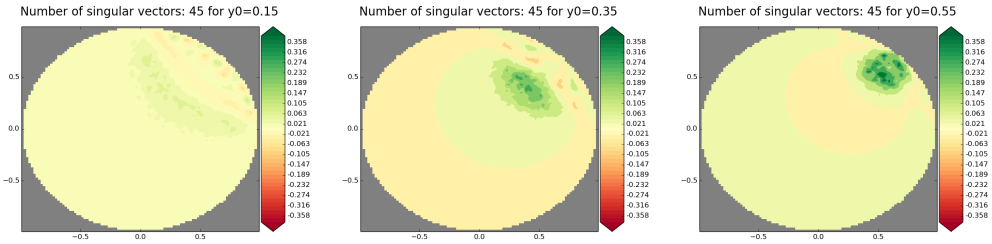
(a) y0=0 *(b) y0=0.25* *(c) y0=0.50*

Figure C.4: *The reconstructed hole function with the half electrode configuration for given y_0 and $x_0 = 0$*



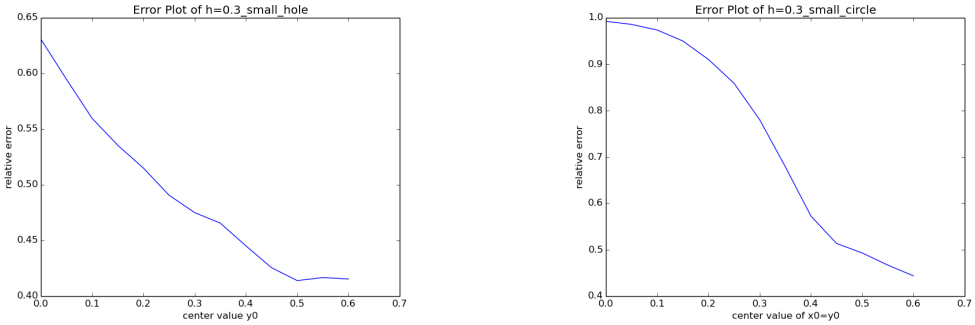
(a) x0=y0=0.15 *(b) x0=y0=0.35* *(c) x0=y0=0.55*

Figure C.5: *The analytical circle function for given (x_0, y_0)*



(a) *x0=y0=0.15* ***(b)*** *x0=y0=0.35* ***(c)*** *x0=y0=0.55*

Figure C.6: *The reconstructed circle function with the quarter electrode configuration for given* (x_0, y_0)



(a) *relative error as a function of* y_0*. Tested on the hole function with the half electrode configuration*

(b) *relative error as a function of* y_0*. Tested on the circle function with the quarter electrode configuration*

Figure C.7: *Relative errors for objects moved from the middle and towards the electrodes*

Appendix D

Code

In this appendix we list the purpose of important functions and scripts used in the project along with the code.

D.1 CEMLibrary_full

This library consists of important functions and is imported in every script file. The prenotation "full" indicates the type of the electrode configuration. We have also two libraries "half" and "quarter", but as these are the same except for lines 34-35 where the length of the electrodens and the space between them are defined, we will not mention them.

D.1.1 solver(sigma,L,I,Z,mesh)

Takes in a FEniCS expression sigma, a number of electrode L, a current vector I of length L, a contact impedance vector Z of length L, and a mesh. It returns the solution (u, U).

D.1.2 cont_plot(x,y,z) and cont_plot2(x,y,z)

Both of these function takes in 3 vectors x,y,z of length N and returns a figure with the corresponding contour plot with appropriate colorbars.

D.1.3 load_frechet()

Loads the matrix representation of the Fréchet derivative equivalent to the electrode configuration.

D.1.4 grid(x, y, z, resX=100, resY=100)

used by cont_plot for plotting.

D.1.5 gramSchmidt(L)

Takes in a number of electrodes L and returns $L-1$ othornormal basis vectors found by Gram Schmidt Orthonormalization with the starting vector $(-1, 1, , 0 \cdots , 0)$ of length L.

D.1.6 create_R_sigma(sigma,L,Z,mesh)

Takes in a FEniCS expression sigma, a number of electrode L, a contact impedance vector Z of length L, and a mesh. It return the matrix $\mathbf{R}_\sigma$.

D.1.7 Frechet(sigma,L,Z,mesh)

Takes in a FEniCS expression sigma, a number of electrode L, a contact impedance vector Z of length L, and a mesh. It returns the matrix representation of the Fréchet derivative $\mathbf{R}'_\sigma$ without the approximation (3.36).

D.1.8 Frechet_point(sigma,L,Z,mesh)

The same as Frechet(sigma,L,Z,mesh), but with the approximation (3.36).

D.1.9 OperatorNorm(A)

Calculates the operator norm of A by returning the largest eigenvalue.

D.1.10 CEMLibrary_full.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 30 06:53:28 2015

@author: ec0di
"""

import numpy as np
from numpy import linalg as LA
from dolfin import *
from mshr import *
import matplotlib.cm as cm
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.colors import LogNorm
from matplotlib.ticker import MultipleLocator
import scipy.sparse as sps
from matplotlib.mlab import griddata
import matplotlib as mpl
#from h_functions import *

path = 'Full/'

def solver(sigma,L,I,Z,mesh):
    # def 2 pi function
    def twopiarctan(x):
        val=np.arctan2(x[1],x[0])
        if val<0:
            val=val+2*pi
        return val

    # Define length of each electrode (uniform length is assumed)
    e_l=pi/L
    d_e=2*pi/L-e_l

    class theta_values(Expression):
        def eval(self,theta,x):
            theta[0]=np.arctan2(x[1],x[0])

```

```
    # Define subdomain mesh
    subdomains = FacetFunction("size_t", mesh)
    subdomains.set_all(0)

    # Define subdomains
    class e(SubDomain):
            def inside(self, x, on_boundary):
                theta=twopiarctan(x)
                return on_boundary and theta>=theta1 and theta<=theta2

    R = FunctionSpace(mesh,"R",0)
    H1 = FunctionSpace(mesh,"CG",1)

    spacelist = []

    for i in range(1, L+1):
        theta1 = (i-1)*(e_l+d_e)
        theta2 = theta1+e_l
        e1 = e()                    # create instance
        e1.mark(subdomains,i)       # mark subdomain
        spacelist.append(R)

    spacelist.append(H1)
    spacelist.append(R)
    # Create funciton space
    V = MixedFunctionSpace(spacelist)

    # Define new measures associated with the boundaries
    dS = Measure('ds', domain=mesh)[subdomains]

    # Define trial and testfunctions
    u = TrialFunction(V)
    v = TestFunction(V)

    # Pre-define f
    f = 0*dS(1)

    B = sigma*inner(nabla_grad(u[L]), nabla_grad(v[L]))*dx
    for i in range(L):
        B += 1/Z[i]*(u[L]-u[i])*(v[L]-v[i])*dS(i+1)
        B += (v[L+1]*u[i]/assemble(1*dS(0)))*dS(0)
        B += (u[L+1]*v[i]/assemble(1*dS(0)))*dS(0)
        f += (I[i]*v[i]/assemble(1*dS(0)))*dS(0)

    # Compute solution
    q = Function(V)
    solve(B == f, q)

    Q = q.split(deepcopy=True)

    # Split solution
    U = []
    for i in range(L+1):
        U.append(Q[i])

    #u = Q[L]
    return U;

def cont_plot(x,y,z):    # Makes contour plot from (x,y,z) and returnes
    figure
    # define the colormap
    cmap = plt.cm.get_cmap('RdYlGn')
    # extract all colors from the .jet map
    cmaplist = [cmap(j) for j in range(cmap.N)]
    # create the new map
```

```
105     cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)
106
107     # define the bins and normalize
108     bounds = np.linspace(-0.1,0.1,20)
109     norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
110     cmap.set_over('green')
111     cmap.set_under('red')
112     fig, ax = plt.subplots(1,1)
113     ax.set_axis_bgcolor('grey')
114     # create grid
115     Xx, Yy, Zz = grid(x, y, z)
116     # make the scatter
117     ax.contourf(Xx, Yy, Zz, cmap=cmap, norm=norm)
118
119     divider = make_axes_locatable(ax)
120     # create a second axes for the colorbar
121     cax = divider.append_axes("right", size="10%", pad=0.05)
122     mpl.colorbar.ColorbarBase(cax, cmap=cmap, norm=norm,
123                                     spacing='proportional',
124                                     ticks=bounds, boundaries=[-10]+bounds
    +[10],
125                                     extend='both',
126                                     format='%.3f')
127     return fig
128
129 def cont_plot2(x,y,z):    # Makes contour plot from (x,y,z) and returnes
    figure
130     # define the colormap
131     cmap = plt.cm.get_cmap('RdYlGn')
132     # extract all colors from the .jet map
133     cmaplist = [cmap(j) for j in range(cmap.N)]
134     # create the new map
135     cmap = cmap.from_list('Custom cmap', cmaplist, cmap.N)
136
137     # define the bins and normalize
138     bounds = np.linspace(-0.4,0.4,50)
139     tickmarks = np.linspace(-0.4,0.4,20)
140     norm = mpl.colors.BoundaryNorm(bounds, cmap.N)
141     cmap.set_over('green')
142     cmap.set_under('red')
143     fig, ax = plt.subplots(1,1)
144     ax.set_axis_bgcolor('grey')
145     # create grid
146     Xx, Yy, Zz = grid(x, y, z)
147     # make the scatter
148     ax.contourf(Xx, Yy, Zz, cmap=cmap, norm=norm)
149
150     divider = make_axes_locatable(ax)
151     # create a second axes for the colorbar
152     cax = divider.append_axes("right", size="10%", pad=0.05)
153     mpl.colorbar.ColorbarBase(cax, cmap=cmap, norm=norm,
154                                     spacing='proportional',
155                                     ticks=tickmarks, boundaries=[-10]+bounds
    +[10],
156                                     extend='both',
157                                     format='%.3f')
158     return fig
159
160 def load_frechet():
161     print "Loading frechet derivative..."
162     M = np.loadtxt('Matrices/Frechet_point_sigma=1_full.txt')
163     print "Loaded"
164     return M
165
166 def grid(x, y, z, resX=100, resY=100):
```

```
            "Convert 3 column data to matplotlib grid"
            xi = np.linspace(min(x), max(x), resX)
            yi = np.linspace(min(y), max(y), resY)
            Z = griddata(x, y, z, xi, yi, 'linear')
            X, Y = np.meshgrid(xi, yi)
            return X, Y, Z

def gramSchmidt(L):
    def proj(u,v):  # Pojection of v on u for Gram-Schmidt
        return np.inner(u,v)/np.inner(u,u)*u



    V = np.zeros((L, L-1))
    U = np.zeros((L, L-1))

    V[0,:] = 1
    for i in range(L-1):
        V[i+1,i] = -1

    U[:,0] = V[:,0]
    # Orthogonalize
    for i in range(1,L-1):
        projSum = 0.
        for j in range(0,i):
            projSum += proj(U[:,j],V[:,i])
        U[:,i] = V[:,i] - projSum

    # Normalize
    for i in range(L-1):
        U[:,i] = U[:,i]/np.sqrt(np.inner(U[:,i],U[:,i]))

    return U

def create_R_sigma(sigma,L,Z,mesh):
    # create basis vectors
    Imat=gramSchmidt(L)

    U = np.zeros((L, L-1))

    for i in range(L-1):
        u=solver(sigma,L,Imat[:,i],Z,mesh)
        for j in range(L):
            U[j,i] = u[j]

    R_sigma = np.zeros((L-1, L-1))

    for i in range(L-1):
        for j in range(L-1):
            R_sigma[i,j] = np.dot(U[:,j],Imat[:,i])

    return R_sigma

# Frechet derivative matrix for a given conductivity
def Frechet(sigma,L,Z,mesh):
    # number of pixels
    N = mesh.num_entities(2)
    # create basis vectors
    Imat=gramSchmidt(L)

    g_w=[]
    for i in range(L-1):
        u=solver(sigma,L,Imat[:,i],Z,mesh)
        g_w.append(grad(u[L]))

```

```
    face_domains = FaceFunction('size_t',mesh)

    faces = [ Face(mesh, i)
                    for i in range(N) ]

    for face in faces:
        face_domains[face] = face.index()

    dsf = Measure('dx', domain=mesh)[face_domains]

    M = np.zeros(((L-1)*(L-1),N))
    # create matrix of functions [i,j]
    for k in range(N):
        vals = np.zeros((L-1, L-1))
        for i in range(L-1):
            for j in range(L-1):
                vals[i,j] = -assemble(inner(g_w[i],g_w[j])*dsf(k))
        vals = np.reshape(vals,((L-1)*(L-1)))
        M[:,k] = vals
    return M


def Frechet_point(sigma,L,Z,mesh):
    N = mesh.num_entities(2)
    print "Calculation frechet derivative with "+str(N)+" pixels"
    Imat=gramSchmidt(L)
    V_g=VectorFunctionSpace(mesh,'CG',1)

    g_w=[]
    for i in range(L-1):
        u=solver(sigma,L,Imat[:,i],Z,mesh)
        grad_w=project(nabla_grad(u[L]),V_g)
        g_w.append(grad_w)

    M = np.zeros(((L-1)*(L-1),N))
    # create matrix of functions [i,j]
    for k in range(N):
        if(np.mod(k,N/99)==0):
            print str((k+0.0)/N*100)+"%"
        vals = np.zeros((L-1, L-1))
        for i in range(L-1):
            for j in range(L-1):
                point=MeshEntity(mesh,2,k).midpoint()
                vals[i,j] = -g_w[i](point).dot(g_w[j](point))*Face(mesh,k).
    area()
        vals = np.reshape(vals,((L-1)*(L-1)))
        M[:,k] = vals
    print "Done with Frechet_point"
    return M

def OperatorNorm(A):
    b=np.dot(A,np.transpose(A))
    e,v=np.linalg.eig(b)
    return sqrt(max(e))
```

D.2 h_functions

This script contains analytical expressions for small perturbations.

h_functions.py

```
# -*- coding: utf-8 -*-
"""
```

```
Created on Thu Jun  4 07:42:00 2015

@author: ec0di
"""
from numpy import *

# circle with radius 1/3
def h_circle(x,y):
    r1=1.0*1/3
    if(x**2+y**2<=r1**2):
        return 0.3
    else:
        return 0

# circle, square and triangle
def h_geometry(x,y):
    r1=0.5
    r2=0.25
    x0=r1/sqrt(2)
    #cirkel top left corner
    if((x-(-x0))**2+(y-x0)**2<=r2**2):
        return 0.3
    #square top right corner
    if(x>=x0-r2 and x<=x0+r2 and y>=x0-r2 and y<=x0+r2):
        return 0.3
    #triangle bot middle
    if(y>=-(r1+r2) and y<=x-r2 and y<=-x-r2):
        return 0.3
    else:
        return 0

# doughnut
def h_hole(x,y):
    r1=1.0*1/3
    r2=1.0*2/3
    if(x**2+y**2<=r1**2):
        return 0
    if(x**2+y**2<=r2**2):
        return 0.3
    else:
        return 0

# square
def h_square(x,y):
    r1=0.3
    #Square in the middle
    if(x>=-r1/2 and x<=r1/2 and y>=-r1/2 and y<=r1/2):
        return 0.3
    else:
        return 0

# banana shape
def h_banan(x,y):
    r1=0.3
    if(y<=-x**2+r1/2 and y>=-0.7*x**2 and x>=-r1 and x<=r1):
        return 0.3
    else:
        return 0

# two half circles with diffent sign
def h_halfcircles(x,y):
    x1=-0.3
    x2=0.3
    if(x<0 and y<=0):
        return h_circle(x-x1,y)
```

```
    if(x>=0 and y>=0):
        return -h_circle(x-x2,y)
    else:
        return 0
```

D.3 reconstruct_moved_object

Code for reconstructing perturbations for different center values (x_0, y_0). They are compared to the analytical functions by calculating the relative norm between them.

reconstruct_moved_object.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun  9 06:19:28 2015

@author: ec0di
"""

from CEMLibrary_half import *
from h_functions import *


# Nr of electrodes


if(path=='Quarter/'):
    L = 10
else:
    L = 20

# generate orthonomal basis
Imat=gramSchmidt(L)

# Define vector of contact impedances
z = 0.1
Z = []
for i in range(L):
    Z.append(z)

# Define domain (Circle)
R = 1   # radius of circle
n = 300     # number of polygons to approximate circle
F = 50  # fineness of mesh
mesh = generate_mesh(Circle(Point(0,0),R,n),F)  # generates mesh
N = mesh.num_entities(2)
print N

# Define conductivity
class sigma_fun(Expression):
    def eval(self,values,x):
        values[0] = 1

# Define h
# h is defined from file h_functions
def h_fun(x,y):
        return h_circle(x-x0,y-y0)

# Define sigma+h
class sigma_h_fun(Expression):
    def eval(self,values,x):
        values[0] = sigma(x)+h_fun(x[0],x[1])

```

```
52
53 # Define string names for later print
54 hstr = "h=0.3_circle"
55
56 # Define H1 room
57 H1=FunctionSpace(mesh,'CG',1)
58
59 # Initiate functions
60 sigma = sigma_fun(element=H1.ufl_element())
61
62 print "Creating R_sigma"
63 R_sigma = create_R_sigma(sigma,L,Z,mesh)
64
65 if(L==10):
66     M = load_frechet_L10()
67 else:
68     M = load_frechet()
69
70 # Create svd
71 print "Doing SVD..."
72 U, s, V = LA.svd(M)
73
74 # Saving midpoints for later
75 print "Saving midpoints..."
76 x = [ MeshEntity(mesh,2,i).midpoint().x()
77                          for i in range(N) ]
78 y = [ MeshEntity(mesh,2,i).midpoint().y()
79                          for i in range(N) ]
80
81 # Getting ready to loop over different h functions
82 y0_max=0.60
83 y0_step=0.05
84 y0_vec=np.arange(0, y0_max+y0_step, y0_step)
85 print y0_vec
86 k_vec=np.zeros(len(y0_vec))
87 error_vec=np.zeros(len(y0_vec))
88 for k in range(len(y0_vec)):
89
90     # Define new y-value for midpoint of inner cirkel
91     y0=y0_vec[k]
92     # Defining x0 to the right configuration
93     if(L==10):
94         x0=y0
95     else:
96         x0=0
97     sigma_h = sigma_h_fun(element=H1.ufl_element())
98
99     print "Creating R_sigma_h for the: "+str(k)+"th time"
100    R_sigma_h = create_R_sigma(sigma_h,L,Z,mesh)
101
102    b=np.reshape(R_sigma_h-R_sigma,((L-1)*(L-1)))
103
104
105    """
106    # Create noise
107    alpha = 0.01
108    sd = alpha*np.max(np.abs(b))
109
110    print "Make some noise!!!"
111    noise = np.random.normal(0, sd, len(b))
112
113    b=b+noise
114    """
115
116    # Create analytical stacked solution
```

```
    h_anal = np.zeros((N))

    for i in range(N):
        midPoint=MeshEntity(mesh,2,i).midpoint()
        h_anal[i]=h_fun(midPoint.x(),midPoint.y())

    h_rec = 0*V[0,:]
    idx_min_error=0
    min_error=1000000
    for i in range(len(s)):
        h_rec=h_rec+np.dot(b,U[:,i])/s[i]*V[i,:]
        if(LA.norm(h_rec-h_anal)<=min_error):
            min_error=LA.norm(h_rec-h_anal)
            idx_min_error=i+1
    print "idx_min_error is: "+str(idx_min_error)
    # Calculate relative error
    rel_error = min_error/LA.norm(h_anal)
    error_vec[k]=rel_error

    # Generating reconstructed best solution
    h_rec = 0*V[0,:]
    for i in range(idx_min_error):
        h_rec=h_rec+np.dot(b,U[:,i])/s[i]*V[i,:]

    # Plotting
    fig=cont_plot2(x,y,h_rec)
    fig.suptitle('Number of singular vectors: '+str(idx_min_error)+" for y0
    ="+str(y0_vec[k]), fontsize=20)

    folder = path+"L="+str(L)+"/Reconstruct/Moved_Object/"+hstr+"/"
    y0_string = "%.2f" %y0_vec[k]
    name = "y0="+y0_string+"_Reconstructed.png"
    # save image
    print "Saving image as: "+name
    fig.savefig(folder+name)
    plt.close(fig)

    # Analytical
    fig=cont_plot2(x,y,h_anal)
    fig.suptitle('Analytical for y0='+str(y0_vec[k]), fontsize=20)
    name = "y0="+y0_string+"_Analytical.png"
    # save image
    print "Saving image as: "+name
    fig.savefig(folder+name)
    plt.close(fig)

print error_vec
# Safe error_vec
name="error_vec_"+hstr
np.savetxt(folder+name,error_vec)
# Plotting
plt.plot(y0_vec,error_vec)
plt.title("Error Plot of "+hstr)
if(L==10):
    plt.xlabel('center value of x0=y0')
else:
    plt.xlabel('center value y0')

plt.ylabel('relative error')
name="error_plot_"+hstr+".png"
plt.savefig(folder+name)
plt.show()
```

D.4 R_sigma

Code for testing the Fréchet derivative. Both the linearization error and the convergence plot are made in here.

R_sigma.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 27 11:12:40 2015

@author: anders

code for creating R_sigma in circle domain
"""


from CEMLibrary_full import *
from h_functions import *
# Define number of electrodes
L = 20

# Define input currents
#I = [-2,5,-3]

Imat=gramSchmidt(L)
#print Imat


# Define vector of contact impedances
z = 0.1
Z = []
for i in range(L):
    Z.append(z)

# Define domain (Circle)
R = 1   # radius of circle
n = 300    # number of polygons to approximate circle
F = 30  # fineness of mesh
mesh = generate_mesh(Circle(Point(0,0),R,n),F)  # generates mesh
N = mesh.num_entities(2)
print N
H1=FunctionSpace(mesh,'CG',1)

#plot(mesh)

# Define conductivity in domain
class sigma_fun(Expression):
    def eval(self,values,x):
        values[0] = 1

sigmastr = "sigma=1"
# make instance of sigma
sigma = sigma_fun(element=H1.ufl_element())

# create R_sigma
R_sigma = create_R_sigma(sigma,L,Z,mesh)

count=0
# values of h function
hs = [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5]
norms = []

M = load_frechet()
```

```
58
59 count=0
60
61 for h_inside in hs:
62     count=count+1
63     # Define h
64     def h_fun(x,y):
65         r1=1.0*1/3
66         if(x**2+y**2<=r1**2):
67             return h_inside
68         else:
69             return 0
70
71     hstr = "h="+str(h_inside)+"_inside"
72     class sigma_h_fun(Expression):
73         def eval(self,values,x):
74             values[0] = sigma(x)+h_fun(x[0],x[1])
75
76     sigma_h = sigma_h_fun(element=H1.ufl_element())
77
78     print "Creating R_sigma_h for the "+str(count)+"th time"
79     R_sigma_h = create_R_sigma(sigma_h,L,Z,mesh)
80
81     diff_matrix = R_sigma_h-R_sigma
82
83     # calculate analytical function
84     h = np.zeros((N))
85     for i in range(N):
86         midPoint=MeshEntity(mesh,2,i).midpoint()
87         h[i]=h_fun(midPoint.x(),midPoint.y())
88
89     Mh = M.dot(h)
90
91     Rm_sigma_h = np.reshape(Mh,(L-1,L-1))
92
93     # matrices to be plotted
94     M1 = diff_matrix
95     M2 = Rm_sigma_h
96     M3 = np.abs(diff_matrix-Rm_sigma_h)
97
98     """
99     # plot matrices
100    name = "Matrices/Comparing_with_difference/"+typ+"_M1-"+sigmastr+"_"+
       hstr+"_L="+str(L)+"_N="+str(N)+"_F="+str(F)+"_n="+str(n)+".txt"
101    # Dump matrix to file
102    print "Saving M1 as: "+name
103    np.savetxt(name, M1)
104
105    name = "Matrices/Comparing_with_difference/"+typ+"_M2-"+sigmastr+"_"+
       hstr+"_L="+str(L)+"_N="+str(N)+"_F="+str(F)+"_n="+str(n)+".txt"
106    # Dump matrix to file
107    print "Saving M2 as: "+name
108    np.savetxt(name, M2)
109
110    name = "Matrices/Comparing_with_difference/"+typ+"_M3-"+sigmastr+"_"+
       hstr+"_L="+str(L)+"_N="+str(N)+"_F="+str(F)+"_n="+str(n)+".txt"
111    # Dump matrix to file
112    print "Saving M3 as: "+name
113    np.savetxt(name, M3)
114    """
115
116    print "plotting..."
117
118    fig, (ax1, ax2, ax3) = plt.subplots(1,3)
119
```

```
    fig.suptitle('Comparing with linearization ('+str(typ)+')', fontsize
    =20)

    ax1.set_title('$R_{\sigma+h}-R_{\sigma}$')
    # Display image, 'aspect='auto'' makes it fill the whole 'axes' (ax3)
    im1 = ax1.matshow(M1, interpolation='nearest', vmin=M1.min(), vmax=M1.
    max(), cmap='jet')
    # Create divider for existing axes instance
    divider1 = make_axes_locatable(ax1)
    # Append axes to the right of ax3, with 20% width of ax3
    cax1 = divider1.append_axes("right", size="20%", pad=0.05)
    # Create colorbar in the appended axes
    # Tick locations can be set with the kwarg 'ticks'
    # and the format of the ticklabels with kwarg 'format'
    cbar1 = plt.colorbar(im1, cax=cax1, format="%.2e")
    # hide axes
    ax1.xaxis.set_visible(False)
    ax1.yaxis.set_visible(False)

    ax2.set_title('$R\'_{\sigma}[h]$')
    im2 = ax2.matshow(M2, interpolation='nearest', vmin=M2.min(), vmax=M2.
    max(), cmap='jet')
    divider2 = make_axes_locatable(ax2)
    cax2 = divider2.append_axes("right", size="20%", pad=0.05)
    cbar2 = plt.colorbar(im2, cax=cax2, format="%.2e")
    ax2.xaxis.set_visible(False)
    ax2.yaxis.set_visible(False)

    ax3.set_title('Abs. difference')
    im3 = ax3.matshow(M3, interpolation='nearest', vmin=M3.min(), vmax=M3.
    max(), cmap='jet')
    divider3 = make_axes_locatable(ax3)
    cax3 = divider3.append_axes("right", size="20%", pad=0.05)
    cbar3 = plt.colorbar(im3, cax=cax3, format="%.2e")
    ax3.xaxis.set_visible(False)
    ax3.yaxis.set_visible(False)

    plt.tight_layout()
    # Make space for title
    plt.subplots_adjust(top=1)

    name = "Matrices/Comparing_with_difference/"+typ+"_img-"+sigmastr+"_"+
    hstr+"_L="+str(L)+"_N="+str(N)+"_F="+str(F)+"_n="+str(n)+".png"
    # save image
    print "Saving image as: "+name
    fig.savefig(name)
    plt.close(fig)

    # calculate norm
    x = OperatorNorm(M3)/h_inside
    norms.append(x)
    print str(1.0*count/len(hs)*100)+"%"

print hs
print norms

hs=np.array(hs)

# plot convergence
plt.plot(np.log(1/hs),norms)
plt.ylabel(r'$\frac{1}{\Vert h \Vert_{L^{\infty}}} \Vert R_{\sigma+h} -R_{\
    sigma} -R_{\sigma}^{,}[h] \Vert_{op}$',fontsize=15)
plt.xlabel(r'$\log{\frac{1}{\Vert h \Vert_{L^{\infty}}}}$',fontsize=15)
plt.title('Convergence Plot')
plt.savefig('Figures/convergencePlot_'+str(typ)+'_full.png')
```

```
#plt.show()
```

D.5 squaredomainCEM

Code for solving the setup in Figure 2.1.

squaredomainCEM.py

```
# -*- coding: utf-8 -*-
"""
Created on Mon Mar  9 05:17:48 2015

@author: ec0di
"""

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

Z=0.1;
I=[2,-2]
x0=0; y0=0; x1=1; y1=1
L=2
mesh=RectangleMesh(x0,y0,x1,y1,10,10)
class sigma_fun(Expression):
    def eval(self,values,x):
        values[0]=1

sigma = sigma_fun()

class Left(SubDomain):
    def inside(self, x, on_boundary):
        return near(x[0], 0.0)

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

# Initialize sub-domain instances
left = Left()
right = Right()

# Initialize mesh function for boundary domains
boundaries = FacetFunction("size_t", mesh)
boundaries.set_all(0)
left.mark(boundaries, 1)
right.mark(boundaries, 2)
#top.mark(boundaries, 2)
#bottom.mark(boundaries, 4)

dS = Measure('ds', domain=mesh)[boundaries]

# Define function space and basis functions
R = FunctionSpace(mesh,"R",0)
H1 = FunctionSpace(mesh,"CG",1)
mixedspaces=[R,R,H1,R]

V = MixedFunctionSpace(mixedspaces)

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

# Pre-define f
```

```
f = 0*dS(1)

B = sigma*inner(nabla_grad(u[L]), nabla_grad(v[L]))*dx
for i in range(L):
    B += 1/Z*(u[L]-u[i])*(v[L]-v[i])*dS(i+1)
    B += (v[L+1]*u[i]/assemble(1*dS(i+1)))*dS(i+1)
    B += (u[L+1]*v[i]/assemble(1*dS(i+1)))*dS(i+1)
    f += (I[i]*v[i]/assemble(1*dS(i+1)))*dS(i+1)

# Solution found and shown
q = Function(V)
solve(B == f, q)

Q = q.split(deepcopy=True)

# Split solution
U = []
for i in range(2):
    U.append(Q[i])

u = Q[2]


for i in range(2):
    print("U"+str(i+1)+": "+str(U[i].compute_vertex_values()[0]))

plot(u)
interactive()
```

D.6 SVD

Code for making SVD of the Fréchet derivative and plotting the singular values and vectors.

SVD.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue May 12 15:02:02 2015

@author: anders
"""
from CEMLibrary_full import *
L = 20

# create basis vectors
Imat=gramSchmidt(L)

# create vector of contact impedances
z = 0.1
Z = []
for i in range(L):
    Z.append(z)

# Define domain (Circle)
R = 1   # radius of circle
n = 300     # number of polygons to approximate circle
F = 50  # fineness of mesh
mesh = generate_mesh(Circle(Point(0,0),R,n),F)  # generates mesh
N = mesh.num_entities(2)
print N

H1=FunctionSpace(mesh,'CG',1)
```

```
28
29 # Define conductivity in domain
30 class sigma_fun(Expression):
31     def eval(self,values,x):
32         values[0] = 1
33 sigma = sigma_fun()
34
35 M = load_frechet()
36
37 print "Doing SVD..."
38 U, s, V = LA.svd(M)
39 print "Done"
40
41 print "plotting singular values..."
42 fig = plt.figure()
43 fig.suptitle('Singular values', fontsize=20)
44 plt.xlabel('$i$')
45 plt.ylabel('log$_{10}$($\sigma_i$)')
46 x = np.arange(1,s.size+1,1)
47 y = np.log10(s)
48 x_lab = 0
49 while(y[x_lab]>-14):
50     x_lab = x_lab+1
51 x_lab = x_lab-1
52 plt.scatter(x,y)
53 lab = x_lab+1
54 plt.annotate(
55         'i = %d' % lab,
56         xy = (x_lab+1,y[x_lab]), xytext = (60, 10),
57         textcoords = 'offset points', ha = 'right', va = 'bottom',
58         bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow', alpha = 0.5)
    ,
59         arrowprops = dict(arrowstyle = '->', connectionstyle = 'arc3,rad=0'
    ))
60
61 # Define saving folder
62 folder = path+"L="+str(L)+"/Basisvectors/"
63 # save image
64 name = "Singular_values"
65 print "Saving image in:   "+folder+"  as  "+name
66 fig.savefig(folder+name+".png")
67 np.savetxt(folder+name+".txt",s)
68
69 # save midpoints
70 x = [ MeshEntity(mesh,2,i).midpoint().x()
71                         for i in range(N) ]
72 y = [ MeshEntity(mesh,2,i).midpoint().y()
73                         for i in range(N) ]
74
75 # number of images to be saved
76 n_images = str(len(s))
77 for i in range(len(s)):
78
79     fig = cont_plot(x,y,V[i,:])
80     fig.suptitle('Singular vector: '+str(i+1), fontsize=20)
81     name = "v_"+str(i+1)+".png"
82     # save image
83     print "Saving image "+str(i+1)+" of "+n_images+" in:   "+folder+"  as
    "+name
84     fig.savefig(folder+name)
85     plt.close(fig)
```

D.7 SVD_reconstruct_h

Code for reconstructing a perturbation.

SVD_reconstruct_h.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 13 08:18:47 2015

@author: ec0di
"""

from CEMLibrary import *
from h_functions import *
# Nr of electrodes
L = 20

# generate orthonomal basis
Imat=gramSchmidt(L)

# Define vector of contact impedances
z = 0.1
Z = []
for i in range(L):
    Z.append(z)

# Define domain (Circle)
R = 1   # radius of circle
n = 300     # number of polygons to approximate circle
F = 50  # fineness of mesh
mesh = generate_mesh(Circle(Point(0,0),R,n),F)  # generates mesh
N = mesh.num_entities(2)
print N

# Define conductivity
class sigma_fun(Expression):
    def eval(self,values,x):
        values[0] = 1

# Define h
# h is defined from file h_functions
def h_fun(x,y):
    return h_hole(x,y)

# Define sigma+h
class sigma_h_fun(Expression):
    def eval(self,values,x):
        values[0] = sigma(x)+h_fun(x[0],x[1])

# Define string names for later print
sigmastr = "sigma=1"
hstr = "h=0.3_hole_cond"
typ = "point"

# Define H1 room
H1=FunctionSpace(mesh,'CG',1)

# Initiate functions
sigma = sigma_fun(element=H1.ufl_element())
sigma_h = sigma_h_fun(element=H1.ufl_element())

# Create matrices
R_sigma = create_R_sigma(sigma,L,Z,mesh)
R_sigma_h = create_R_sigma(sigma_h,L,Z,mesh)

Y=np.reshape(R_sigma_h-R_sigma,((L-1)*(L-1)))

# Create Frechet derivate in matrix form (L-1)^2 X N
```

```
M = Frechet_point(sigma,L,Z,mesh)

# Create svd
print "Doing SVD..."
U, s, V = LA.svd(M)

# Define midpoint coordinates for cells
print "Saving midpoints..."
x = [ MeshEntity(mesh,2,i).midpoint().x()
                          for i in range(N) ]
y = [ MeshEntity(mesh,2,i).midpoint().y()
                          for i in range(N) ]
h = 0*V[0,:]
counter_5=1
for i in range(len(s)-5): # minus 5 since the last values in s are so small,
                          #  so h=h+... takes infi long time.
    h=h+np.dot(Y,U[:,i])/s[i]*V[i,:]

    if(counter_5==0):
        #plotting
        condNr=s[0]/s[i]
        fig = plt.figure()
        fig.suptitle('Antal basisvektorer: '+str(i+1)+", Cond: "+str(condNr),
                                                                  fontsize=20)
        Xx, Yy, Zz = grid(x, y, h)
        plt.contourf(Xx, Yy, Zz)
        plt.colorbar()
        name = "Matrices/h_reconstruct/"+hstr+"/h_"+str(i+1)+"_L="+str(L)+"_N="+str(N)+"_F="+str(F)+"_n="+str(n)+"v_"+str(i+1)+"_"+typ+"_"+sigmastr+"_"+hstr+".png"
        # save image
        print "Saving image as: "+name
        fig.savefig(name)
        plt.close(fig)
    # Update counter
    counter_5=counter_5+1
    counter_5=np.mod(counter_5,5)
```

D.8 SVD_reconstruct_h_noise

Script for reconstructing a perturbation with added noise on the data and also the analytical perturbation.

SVD_reconstruct_h_noise.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 13 08:18:47 2015

@author: ec0di
"""

from CEMLibrary_quarter import *
from h_functions import *
# Nr of electrodes
L = 20

# generate orthonomal basis
Imat=gramSchmidt(L)

```

```
# Define vector of contact impedances
z = 0.1
Z = []
for i in range(L):
    Z.append(z)

# Define domain (Circle)
R = 1   # radius of circle
n = 300     # number of polygons to approximate circle
F = 50  # fineness of mesh
mesh = generate_mesh(Circle(Point(0,0),R,n),F)  # generates mesh
N = mesh.num_entities(2)
print N

# Define conductivity
class sigma_fun(Expression):
    def eval(self,values,x):
        values[0] = 1

# Define h
# h is defined from file h_functions
def h_fun(x,y):
    return h_geometry(x,y)

# Define sigma+h
class sigma_h_fun(Expression):
    def eval(self,values,x):
        values[0] = sigma(x)+h_fun(x[0],x[1])

# Define string names for later print
sigmastr = "sigma=1"
hstr = "h=0.3_geometry"

# Define H1 room
H1=FunctionSpace(mesh,'CG',1)

# Initiate functions
sigma = sigma_fun(element=H1.ufl_element())
sigma_h = sigma_h_fun(element=H1.ufl_element())


# Create matrices
print "Creating R_sigma"
R_sigma = create_R_sigma(sigma,L,Z,mesh)
print "Creating R_sigma_h"
R_sigma_h = create_R_sigma(sigma_h,L,Z,mesh)

Y=np.reshape(R_sigma_h-R_sigma,((L-1)*(L-1)))
print Y
# Create noise
alpha = 0.01
sd = alpha*np.max(np.abs(Y))
print sd

print "Make some noise!!!"
noise = np.random.normal(0, sd, len(Y))
print noise
print Y
print "Find relative error.."
Yn = Y + noise
rel_error = LA.norm(Y-Yn)/LA.norm(Y)
print rel_error

# Create Frechet derivate in matrix form (L-1)^2 X N
M = load_frechet()
```

```
81
82 # Create svd
83 print "Doing SVD..."
84 U, s, V = LA.svd(M)
85
86 # Define midpoint coordinates for cells
87 print "Saving midpoints..."
88 x = [ MeshEntity(mesh,2,i).midpoint().x()
89                             for i in range(N) ]
90 y = [ MeshEntity(mesh,2,i).midpoint().y()
91                             for i in range(N) ]
92 h = 0*V[0,:]
93 counter_5=1
94
95 h_anal = np.zeros((N))
96
97 for i in range(N):
98     midPoint=MeshEntity(mesh,2,i).midpoint()
99     h_anal[i]=h_fun(midPoint.x(),midPoint.y())
100
101 folder = path+"L="+str(L)+"/Reconstruct/Noise/"+hstr+"/"
102
103
104 h = 0*V[0,:]
105 idx_min_error=0
106 min_error=1000000
107 for i in range(len(s)):
108     h=h+np.dot(Y,U[:,i])/s[i]*V[i,:]
109     if(LA.norm(h-h_anal)<=min_error):
110         min_error=LA.norm(h-h_anal)
111         idx_min_error=i+1
112 print "idx_min_error is: "+str(idx_min_error)
113
114 # Generating reconstructed best solution
115 h = 0*V[0,:]
116 for i in range(idx_min_error):
117     h=h+np.dot(Y,U[:,i])/s[i]*V[i,:]
118
119
120 fig=cont_plot2(x,y,h)
121 fig.suptitle('Number of singular vectors: '+str(idx_min_error), fontsize
    =20)
122 name = "Reconstructed_no_noise.png"
123 # save image
124 print "Saving image as: "+name
125 fig.savefig(folder+name)
126 plt.close(fig)
127
128 fig=cont_plot2(x,y,h_anal)
129 fig.suptitle('Analytical', fontsize=20)
130 name = "Analytical.png"
131 # save image
132 print "Saving image as: "+name
133 fig.savefig(folder+name)
134 plt.close(fig)
```

Bibliography

[1] Adams, R. A., and Fournier, J. J. F. Sobolev Spaces. 2nd ed. Pure and Applied Mathematics 140. New York, NY: Academic Press. xiii, 305 p., 2003.

[2] Adler, A., et al. Whither lung eit: Where are we, where do we want to go and what do we need to get there? Physiological Measurement 33, 5 (2012), 679.

[3] Evans, L. C. Partial Differential Equations. Graduate studies in mathematics. American Mathematical Society, Providence (R.I.), 1998. Réimpr. avec corrections : 1999, 2002.

[4] Griffel, D. H. Applied Functional Analysis. Ellis Horwood, 1981.

[5] Hansen, P. Discrete Inverse Problems. Society for Industrial and Applied Mathematics, 2010.

[6] Karhunen, K., et al. Electrical resistance tomography imaging of concrete. Cement and Concrete Research 40, 1 (2010), 137 – 145.

[7] Logg, A., et al. FEniCS, `http://fenicsproject.org/`.

[8] Somersalo, E., Cheneyand, M., and Isaacson, D. Existence and Uniqueness for Electrode Models for Electric Current Computed Tomography. SIAM J. Appl. Math. 52, 4 (Aug. 1992), 1023–1040.

[9] Strang, G. The Fundamental Theorem of Linear Algebra. American Mathematical Monthly (1993), 848–855.

[10] van Rossum, G. Python (programming language). Available at `https://www.python.org/`.