Image Reconstruction on Electrical Resistance Tomography Measurements in a Settling Process

by

Siegfried Herholdt

A thesis submitted to the

Department of Electrical Engineering
University of Cape Town

for the degree of

MASTER OF SCIENCE IN ENGINEERING

19 October 2004

Statement of originality

I declare that the work presented in the thesis is, to the best of my knowledge and belief, original and my own work, except as acknowledged in the text, and that the material has not been submitted, either in whole or in part, for a degree at this or any other university.

Signed by candidate

Siegfried Herholdt

Acknowledgments

I would like to thank in first place my supervisor, Prof. Jon Tapson, who supported my thesis and guided me whenever necessary.

Many thanks also to Victor Balden whose advice and help on understanding the Finite Element Method has been invaluable.

Thank you to Prof. John Greene for his patience on helping me understand regularisation and for letting me use his code of the Adaptive Mutation Breeder Genetic Algorithm.

Thank you to Dr. Wilkinson, for his advice on overcoming some hurdles in implementing the Newton-Raphson algorithm.

Abstract

This thesis uses Electrical Resistance Tomography (ERT) to measure a settling process. The measurement hardware for this system was developed during previous work carried out by the author. This research is the continuation of this work by focusing on image reconstruction from the measured data.

The reconstruction algorithm attempts to find the conductivity distribution of the interior of a rig from measurements made on its boundary. The method used in this thesis is the Newton-Raphson (NR) Algorithm, which employs iterative solutions of the Finite Element Method (FEM) in order to converge to a solution for conductivity. FEM is a numerical simulation method that discretises a region into elements and performs a global optimisation or minimisation in order to arrive at a solution. The NR-method is similar in principle to Newton's method, but is designed to operate on complex data and matrix equations in multiple dimensions.

This thesis discusses the methods in reasonable detail and highlights several issues, including ill-conditioning, and regularisation as a method to improve the conditioning of the data. The results contain images of succesful reconstruction of a settling process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Introduction

This thesis deals with the subject of Electrical Impedance Tomography (EIT), or more specifically, Electrical Resistance Tomography (ERT). Electrical Tomography is a technology whereby electrical boundary measurements are made of an object or a region of interest, and that data is used to extract information on the object interior, commonly called image reconstruction. The particular application envisaged for this research project is the measuring of a settling process using ERT.

Settling processes are part of various mining operations and serve the aim of separating liquid and solid components as efficiently as possible. Slurry is pumped into large settling tanks, where the liquid component overflows on the top while the mud settles to the bottom and is pumped away by special underflow pumps. Special chemicals called flocculants are added to speed up the liquid-solid separation.

There exists an incentive to optimise this process, by determining the exact inflow and underflow rate together with just the right amount of flocculant that would maximise throughput rate. Probably the most important input information to this optimisation problem is the solid-liquid interface level, however this information is difficult to establish. This thesis seeks to investigate the possibility of using ERT to acquire information about the state of settling at any given moment.

1.2 Background to Project

This research project was funded by the De Beers Mining Group and was carried out under the supervision of Prof. J. Tapson. It was started by Gavin Teague [1] and taken over by the author as an undergraduate project. The undergraduate project dealt mainly with the measurement hardware and the data acquisition system.

1.3 Aims and Objectives

Since this thesis is a continuation of the work started by the author in his undergraduate project [2], the main aim is to complete the "circle" by achieving image reconstruction from the data recorded by the measurement circuit. The full set of objectives are stated below:

1. Achieve image reconstruction from measurement data.

2. Evaluate the quality and extent of information acquired from the reconstruction results.

3. Comment on the usefulness of this particular method.

1.4 Scope and Limitations

This research project investigated and sought to understand the theory of the chosen reconstruction methods in reasonable detail. Detailed explanations of the methods used are given where possible. The software developed for this thesis is not fully optimised, since that was not the objective. However, where practicality and ease of use demanded, ways of speeding up code execution were investigated and implemented. Therefore it should also be noted that the developed ERT system is not a real time system in its current state. This could be the objective of further development.

1.5 Layout and Plan of Development

This document starts by introducing some background information to the general subject, also by referencing relevant literature. The hardware system as developed during previous work is introduced and documented for completness.

The fourth chapter introduces and discusses the theory behind a core part of the research, the Finite Element Method. The Newton-Raphson reconstruction algorithm is discussed next as a solution to the inverse problem. The following chapters describe some practical aspects, such as data calibration and the structure of the software. Chapter eight presents the results, followed by conclusions and recommendations.

Chapter 2

Background

This chapter seeks to present a background overview of electrical impedance tomography.

2.1 Hardware and Data Collection

2.1.1 Measurement Strategies

The aim of Electrical Impendance Tomography is to establish the impedance distribution of an object, by making measurements on its boundary. There are several ways to approach this aim, most of which are extensively recorded in literature. Almost all strategies inject current using some boundary electrodes, while the other electrodes are used to measure the potential difference. Any measurement strategy should aim to make as many independent measurements about a region as possible. Webster [3] compiled a literature study that lists most measurement strategies. The most relevant to this thesis appear below for completness and reference purposes.

Neighbouring Method

Also called the adjacent pair method, this strategy injects current between two adjacent electrodes, while measuring the voltage difference between other adjacent electrodes along the boundary. This method has superior sensitivity on the periphery of the region, but does not contain much information on the resistivity in its centre. The neighbouring method is depticed in figure 2.1.

Cross Method

Hua *et al* (1987) [4] used a method of injecting current between electrodes a certain distance from each other. Alternate electrodes are used to measure voltage difference

with respect to a reference electrode. This method has the advantage of better sensitivity over the entire region, although some sensitivity is lost at the periphery.

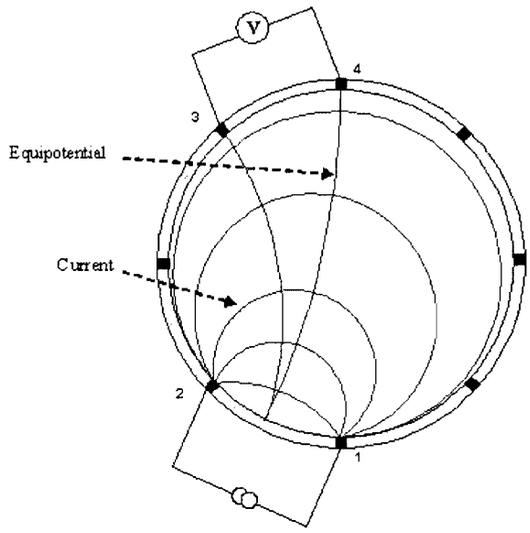

Figure 2.1: The Neighbouring Method. The diagram shows a circular rig with eight electrodes spaced at equal distances around the boundary. The lines represent the current lines and equipotential voltage lines respectively.

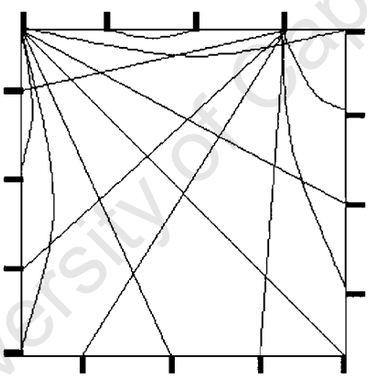

Figure 2.2: The Cross Method. The diagram shows two-dimensional view of a square rig, and the lines represent current lines between electrodes, following the protocol outlined in subsection 2.1.1.

Opposite Method

This method, also used by Hua *et al* (1987) [4], injects current between two directly opposite electrodes over the region, and measures the voltage difference between all other electrodes and the reference electrode, which is chosen adjacent to the current injecting electrodes.

Multireference Method

This method is slightly different in principle to those above, which all inject current through one pair of electrodes, while measuring the voltage across another pair. The multireference method, also used by Hua *et al* (1987) [4], selects one reference electrode and grounds it, and injects a current into all other electrodes simultaneously. The diagram in figure 2.3 shows the basic setup. Current from the injection electrodes exits through the reference electrode. Each current source can be varied from 0 to 5 mA. The voltage between each current injecting electrode and the reference electrode is then measured. These voltages make up the data points. The reference electrode is then changed to the next electrode, and the voltage measurements repeated. This process is repeated in order to make each electrode a reference electrode in turn. That way a complete data set is obtained.

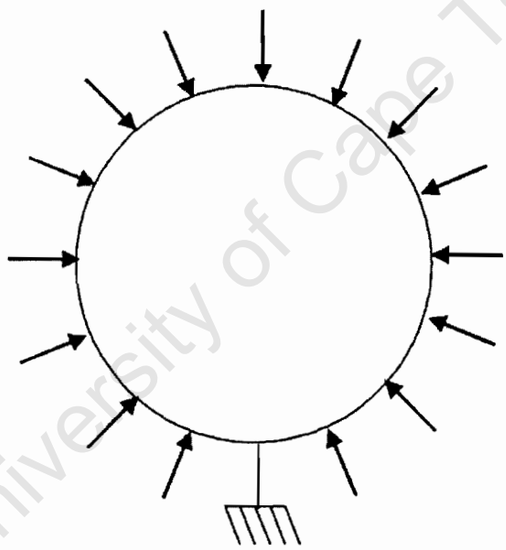

Figure 2.3: The Multireference Method. The arrows represent current injection electrodes. Current is injected into all electrodes, except for the grounded one. This process is repeated by grounding each electrode in succession and letting all others be current injection electrodes.

Adaptive Method

Not much different to the multireference method, this method was suggested by Gisser et al (1987) [5]. Again current is sunk through one grounded reference electrode, while being injected through all other electrodes. However, the current from the injecting electrodes is varied between -5mA and 5mA, based on the resistivity distribution. The objective is to maximise the sensitivity over the entire region. The current is optimised after several measurement cycles.

2.1.2 Sensor Circuitry

This section discusses issues relating to the measurement circuitry, and is largely a summary from the author's undergraduate project [2] and is included for completness.

Electrodes and Skin Impedance

When DC current is passed from an electrode through an electrolyte solution, a counter emf develops that interferes significantly with current flow. Thus all electrical tomography systems use some sort of alternating current to combat this phenomenon. Cilliers *et al* (2001) [6] developed a system that uses a bi-directional DC current pulse system as a source in order to avoid the effects of skin impedance, while most other systems use simple alternating current.

The interface between electrodes and any medium is not a perfect conductor, and has been studied and modelled by Szczepanik *et al* (2000) [7]. He also made a frequency analysis of the contact impedance.

Voltage Source vs. Current Source

Dickin and Wang [8] argued for the use of a current source. They said that a good current source has high output impedance, reducing the effects of unwanted impedances such as skin impedance. Also, since tomography has traditionally been studied for medical research, a monitored current source is an important consideration for a patient's safety.

Other authors have opted for voltage sources [9, 10], arguing that they are much easier to implement than current sources. If current levels were an issue, such as for medical purposes, the solution is to monitor the current levels constantly. For industrial purposes, safe current levels are not as crucial as in medical applications. The main benefit of using voltage sources remains the ease of implementation.

Data Acquisition

Part of any tomography system is the data acquisition system. The data needs to be synchronised and sampled using a personal computer or microprocessor or both. Once acquired, the data is to be used for further computation and analysis, and ultimately for image reconstruction. Usually in the form of an ISA or PCI interface card or more recently a USB device, it acts as an interface between computer and measurement circuitry. The circuit can be software controlled and the data is sampled with A-to-D converters on the data aquisition card.

Some authors have opted for DSP chips that feature onboard demodulation of the signals, instead of making the demodulation a part of the measurement circuit [8].

2.2 Image Reconstruction and Software Algorithms

As already mentioned, measurements are taken at the boundary of the region, but the aim is to obtain some information of the impedance distribution of the interior region. To achieve this involves the use of mathematical reconstruction algorithms. Since the governing differential equation that describes current flow in a medium cannot be solved analytically for resistivity, it needs to be solved numerically employing an inverse method. All inverse methods need some sort of forward method to obtain a solution.

2.2.1 Data Calibration

Data calibration must be done after the data has been sampled to a PC but before data processing begins. It is supposed to account for inaccuracies that are difficult or even impossible to correct. Examples of such are discrepancies between electrode position in the simulation model and in practice, unexpected resistances such as electrode skin impedance, or tolerance of electronic components. Any tomography system needs to include some sort of data calibration [11, 12, 13].

Calibration is often done by comparing simulated measurements against actual measurements for the case of a homogenous region of interest, e.g. tap water. Well considered calibration coefficients are then solved for using various approaches.

2.2.2 The Governing Equation

The differential equation that governs current flow in a medium is Poisson's Equation:

$$\nabla \cdot (\sigma \cdot \nabla U) = I$$

where U is voltage, I is current and σ is the resistivity. The resistivity cannot be solved for analytically, giving rise to the inverse problem. When solving for U or C with a known conductivity distribution σ, we refer to the forward problem. The derivation of Poisson's equation is explained in subsection 4.2.1 on page 17.

2.2.3 The Forward Problem

The forward problem is concerned with predicting boundary measurements from a known resisitivity distribution. This usually involves the Finite Element Method (FEM) [14, 15] or a similar method, the Finite Difference Method (FDM). Both are numerical methods. For FEM, the region is divided into discrete elements and some sort of minimisation or optimisation is performed over the entire region to obtain a solution. For a detailed explanation of FEM as used for this thesis see chapter 4 on page 16.

2.2.4 The Inverse Problem

The inverse problem is concerned with solving for the internal resistivity distribution for a given set of boundary measurements, and often, but not always, employs iterative solutions of the forward problem. All inverse methods need to employ the forward problem in order to obtain a solution . This is due to the fact that the governing equation can not be solved analytically for conductivity. Some popular inverse methods are mentioned below [3].

Basic Backprojection Method

Basic backprojection attempts to determine the conductivity distribution by back-projecting the potential difference between two equipotential lines on the periphery of the object to the conductivity value in the interior, between these two equipotential lines [16, 3]. An explanation of the method can be found in the references and will not be outlined here.

Perturbation Method

This method is based on applying constant voltages and measuring the current magnitude in response to the conductivity distribution. A "perturbation matrix" is calculated by changing the conductivity for one element in the interior and calculating the change in current flowing from the current exit electrodes [17]. The proportional change is then backprojected using the perturbation matrix.

Newton-Raphson Method

This inverse method is considered the theoretically most sound method for image reconstruction in EIT and was described by Yorkey *et al* (1986) [18] and compared to other reconstruction algorithms [19]. The method is described in more detail in section 5.1 on page 27 as it is the chosen method of reconstruction for this thesis.

2.2.5 Ill-Conditioning and Regularisation

Tomographic measurements are by nature ill-conditioned. This means that the boundary measurements are relatively insensitive to resistivity changes in the interior of the region, or in other words, a large change in the interior resistivity distribution causes only a small change to measurements on the periphery. This makes reconstruction problematic. For one, noise can have a large effect on reconstruction accuracy. Small inaccuracies such as electrode placement and stray impedance also affect reconstruction significantly.

The idea of intervention prior to reconstruction in order to improve conditioning of the data is a popular approach in EIT image reconstruction [20, 21, 22, 19]. Regularisation provides for the inclusion of prior information in order to improve accuracy of the results.

Regularisation is somewhat of a tweaking procedure. Based on the extent of the ill-conditioning and expected inaccuracies, the magnitude of regularisation differs. This concept is thoroughly dealt with in section 5.2 on page 29.

Chapter 3

Hardware and Sensor System

3.1 System Overview

The system diagram in figure 3.1 presents a block diagram overview of the entire Tomography system. Measurements of the rig made by the sensor circuitry are converted into digital form by the data acquisition card plugged into and controlled by a personal computer. The PC also hosts the software for image reconstruction, the development of which is the main aim of this thesis.

3.2 Previous Work

In the authors undergraduate project, the aim was to develop the sensor hardware for an electrical resistance tomography rig used to measure a settling process. A measurement system including sensor circuitry, a data acquisition system for reading data to a PC, and a suitable Graphical User Interface (GUI) for viewing the data in various formats was produced. It was shown intuitively and by explanation that relevant data was obtained from the sensors, based on an actual settling process (figure 3.8).

3.2.1 The Rig

The rig in figure 3.2 is the original settling rig that was built by Gavin Teague [1] for the purpose of this project. It is rectangular in shape and has eight point electrodes on two opposing walls (arranged vertically), thus giving a total of 16 electrodes. The rig differs to conventional ERT/EIT geometries in that it is not circular; also the area to be sampled is a vertical plane, as opposed to the more conventional horizontal arrangement of electrodes.

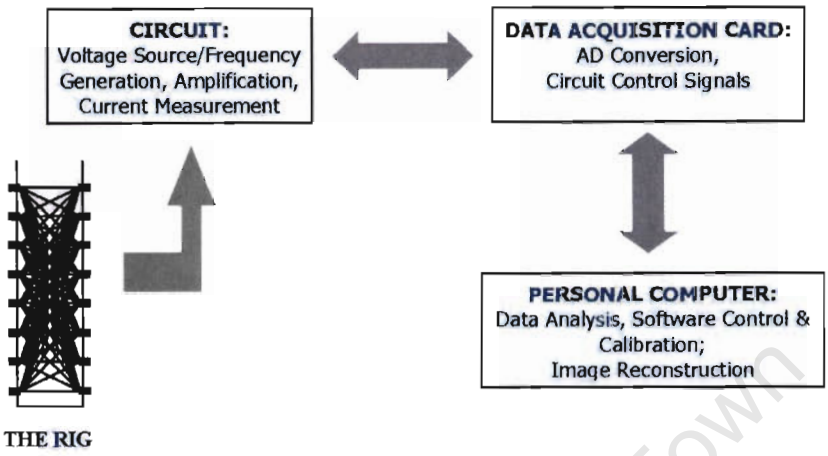

Figure 3.1: System Overview.

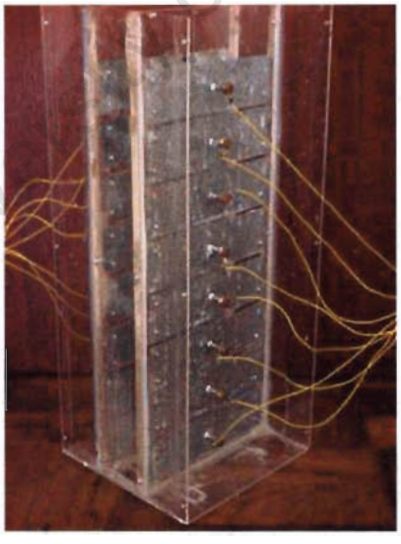

Figure 3.2: The "old" settling rig as built Gavin Teague [1]. One can identify the point-like electrodes used. Note that the side walls containing the electrodes are spaced only a very small distance apart.

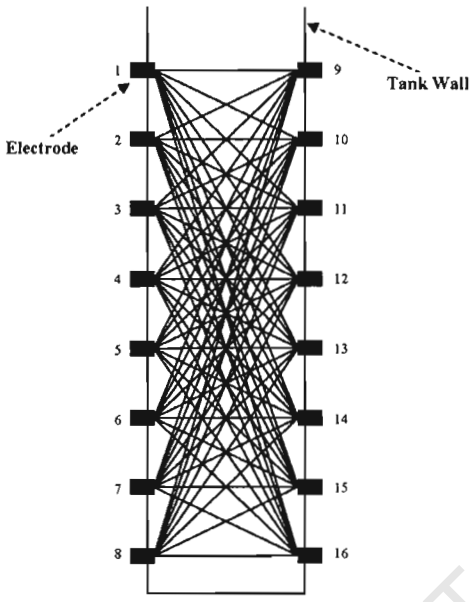

Figure 3.3: Measurement Strategy: Adaptation of the Multireference Method. The lines joining the electrodes are simplified current lines and show the many possible current paths for the particular measurement strategy used. The current lines are an indication of how much information the measurements gather on the region.

3.2.2 Measurement Strategy

The measurement strategy used for this thesis is basically the multireference method (see subsection 2.1.1 on page 5). However, there are a few subtle differences. The author decided to use a voltage source for injecting current instead of a current source as is typically used with the multireference method. Since the voltage is known, it is the current magnitude between electrode pairs that has to be measured. The multireference method, on the other hand, measures the voltage difference between electrode pairs and where current magnitude is known. In the end, both voltage and current measurements contain the same information about the region, only the implementation differs somewhat.

The measurement cycle starts by applying a sinusoidal voltage at electrode 1 on the left hand side of the rig (fig. 3.3). Electrodes 9 to 16 on the right hand side of the rig are held at virtual earth by opamp action and an eight-channel synchronous detector circuit measures the current sunk by those electrodes. Next, the sinusoidal voltage is applied to electrode 2, while current being sunk by electrodes on the right hand side, all at virtual earth, is measured. The sinusoidal voltage is applied to each electrode on the left hand side in succession up to electrode eight. While the voltage is applied to an electrode, the other electrodes on the left hand side remain floating.

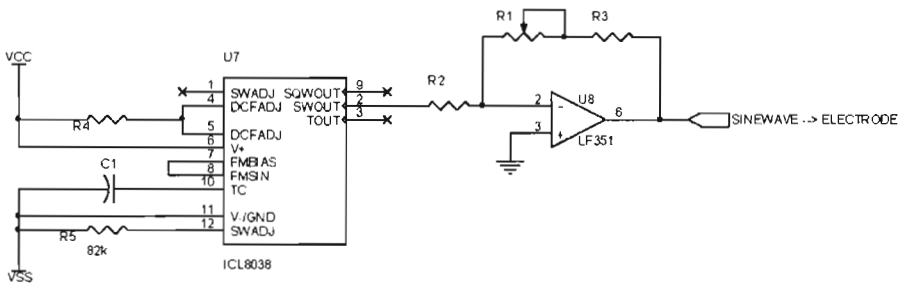

Figure 3.4: Function Generator Circuit Diagram.

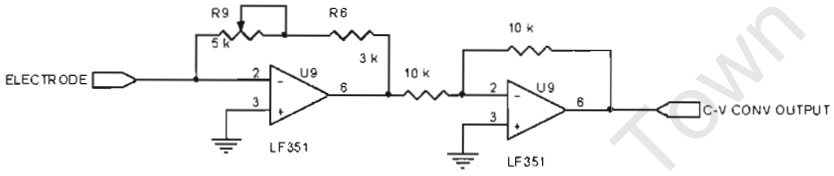

Figure 3.5: Current-to-Voltage Converter.

3.2.3 Circuit Design

The measurement circuit consists of a sinusoidal voltage source and eight synchronous detectors [2]. Eight relays are used to switch the sinusoidal voltage source between the electrodes. A circuit diagram of the function generator is shown in figure 3.4. An ICL8038 chip has been used to generate the sinusoidal voltage, feeding into a buffer circuit before being applied to the electrodes.

At the time, using a voltage source instead of a current source seemed the more attractive option, although this was offset by the need to build several synchronous detectors in order to measure the current magnitude between each electrode pair. If the author were faced with the task of re-designing the circuit in retrospect, those previous judgements would possibly be reconsidered. However, at the time the circuit was delivering satisfactory results and there was no need to consider a different design.

Each grounded, current sinking electrode is connected to a a current-to-voltage converter, which feeds the voltage into a demodulation circuit (synchronous detector) that converts the amplitude into a DC voltage. The DC voltage is directly proportional to the current flowing into each individual channel. A diagram of the current-to-voltage converter is shown in figure 3.5, while the synchronous detector is shown in figure 3.6. The output of the demodulation circuit connects to the data acquisition system (DAS) which samples each voltage into digital form.

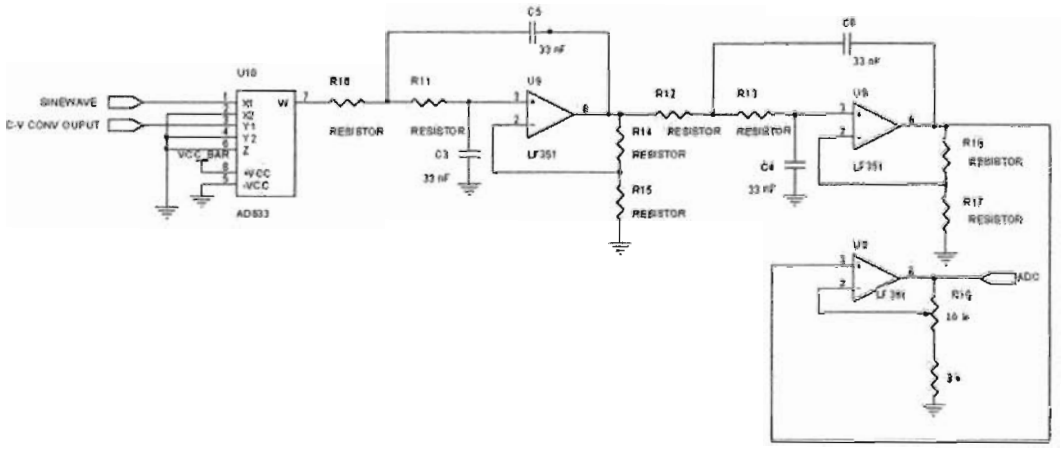

Figure 3.6: Demodulation Circuit (Synchronous Detector).

An overview of the circuit in block form, showing how the individual components make up the measurement circuit is shown in figure 3.7.

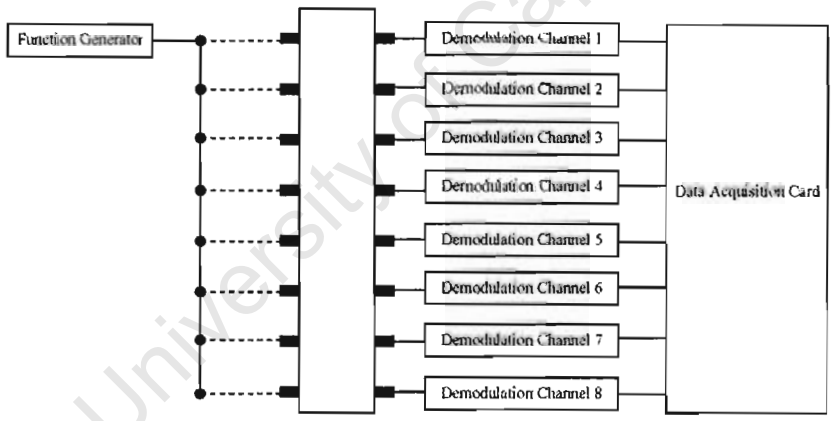

Figure 3.7: Complete circuit in block diagram form

3.2.4 Data Acquisition

An Eagle PC30B Data Acquisition (ISA) Card is used to sample the measurements and to control the switching of the relays. The Card interfaces to MATLAB and driver functions are supplied that allow for reading and controlling of the card. Measurements are stored in computer memory before being retrieved by the reconstruction software.

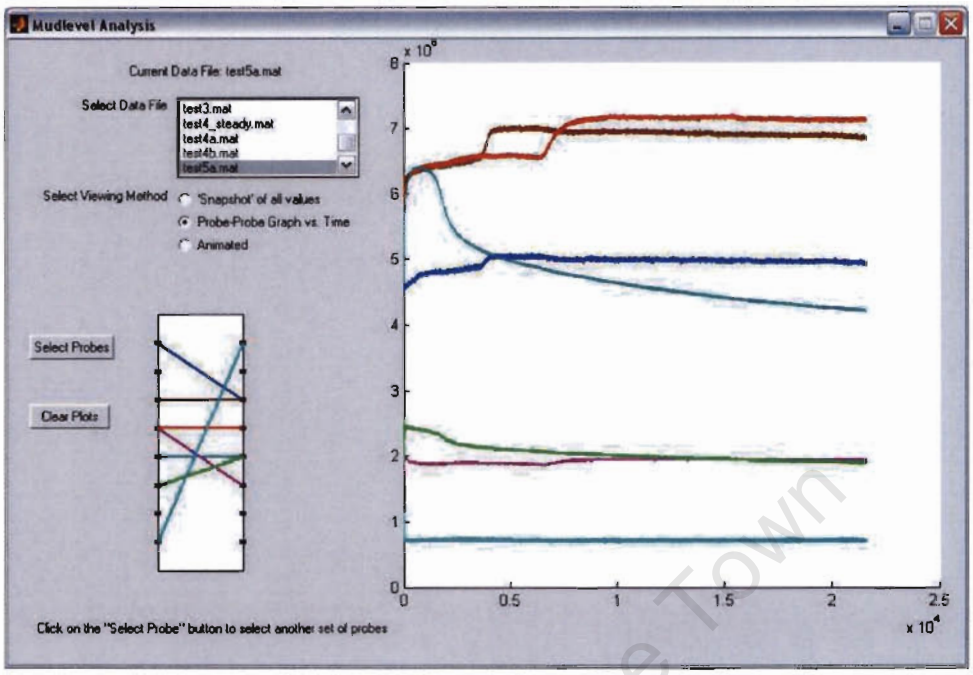

Figure 3.8: GUI for Viewing Measurement Data prior to Reconstruction.

3.2.5 Previous Results

A sample of the measurement results for an experimental settling process is shown in figure 3.8, using the hardware described above and a GUI written for convenient viewing purposes. A diagram of the rig in the bottom left corner of the GUI enables the user to select the pair of electrodes for which current measurement must be shown in the adjacent graph. The current lines and their corresponding graphs are colour-coded. The graphs represent the current magnitude over a period of time, based on the selected measurement set (top left corner in GUI).

Chapter 4

The Forward Problem: Finite Element Method

4.1 Introduction

4.1.1 General

Differential equations govern many physical phenomena in engineering, e.g. electromagnetic fields, current flow, stresses and heat flow, to name just a few. Methods exist to solve these equations on a small scale, but what happens if the problem becomes large and we need to analyse a phenomenon over a large area or volume? The finite element method (FEM) is a numerical method that can solve differential equations accurately. FEM is most useful when combined with the power of a digital computer.

For the case of modelling current flow, a defined region is divided into discrete elements (e.g. triangles), their size depending on the resolution required. The problem is then expressed in a matrix equation, and a minimisation or optimisation procedure is performed over the entire domain. The solution contains voltage values and boundary currents at discrete points in the region that is being modelled.

This chapter explains the finite element method specifically for solving the forward problem in the context of this thesis.

4.1.2 Word of caution

As with any modelling or simulation problem, it is of absolute importance to stay in touch with the real problem. The slightest error could render a modelling solution completely useless. Thus an exceptional amount of effort should go into verifying the correctness and validity of the model, while always keeping in mind that the

simulation environment is ideal. Many errors may go unnoticed, because the results "look right", which makes it even more difficult to spot errors [15]. The author encountered various obstacles during implementation, often due to careless and seemingly insignificant errors. A sound understanding of the method is very helpful when it comes to effective debugging.

4.2 The Governing Equation

4.2.1 Derivation of Differential Equation

The equations that describe current flow in a medium can be derived as follows [23]:

Current density in a conductive medium subject to an electric field can be described by the following formula:

$$J = \sigma \cdot E \tag{4.1}$$

Where J is the current density in (A/m^2), σ is the conductivity (S/m) and E is the electric field (V/m).

For any conductor that isn't a current source, all current entering a point must also leave that same point. This law can be described using the divergence operator:

$$\nabla \cdot J = 0 \tag{4.2}$$

The following equation relates electric field E to potential U:

$$E = -\nabla U \tag{4.3}$$

When we combine the three equations we get Poisson's equation in vector form:

$$\nabla \cdot (\sigma \cdot \nabla U) = 0 \tag{4.4}$$

The governing equation in eq. 4.4 can also be expanded to the following form for the two-dimensional case:

$$f(x,y) = \frac{\partial}{\partial x}\left(\sigma_x(x,y)\frac{\partial U(x,y)}{\partial x}\right) + \frac{\partial}{\partial y}\left(\sigma_y(x,y)\frac{\partial U(x,y)}{\partial y}\right) \tag{4.5}$$

where $\sigma(x,y)$ is the general form of conductivity, allowing for description of conductivity that is a function of x and/or y. For our case however, σ is in fact a scalar that is uniformly defined for each element and not a function of x and y as shown above.

4.2.2 Boundary Conditions

For eq. 4.5, we define the essential boundary conditions as

$$U = U_0 \text{ on } \partial\Lambda \tag{4.6}$$

where U_0 is the voltage on the boundary ∂A, and the natural boundary conditions are expressed as

$$\sigma\frac{\partial U}{\partial x} + \sigma\frac{\partial U}{\partial y} = I_0 \text{ on } \partial\Lambda \tag{4.7}$$

where I_0 is the current injected or sunk on boundary ∂A [24]. The meaning and application of the boundary conditions will become clear further on in this chapter, as more concepts are introduced and loose ends are tied up.

4.3 Elements

The region is divided into discrete elements. We use triangular elements since they can be described with simpler shape functions than other element types. For each element we need the node data, as shown in figure 4.1.

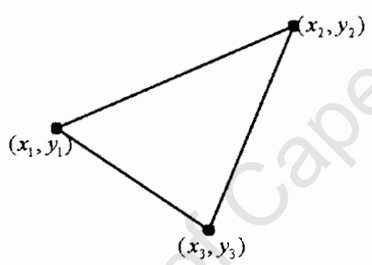

Figure 4.1: A triangular element, consisting of three nodes with coordinates (x_1, y_1), (x_2, y_2) and (x_3, y_3) for nodes 1, 2 and 3 respectively.

The entire mesh used for the rig is shown in fig. 6.2 on page 38. There are 32 vertical levels and 20 elements for each level. It is a rather fine mesh, however the FEM code was found to be fast enough and thus did not require a compromise in terms of mesh resolution.

4.3.1 The Trial Solution

Since FEM solves for voltages at the nodes only, the voltage inside an element is approximated using interpolation functions between nodes and is described by the

trial solution:

$$\tilde{U}(x,y) = \sum_{j=1}^{N=3} \phi_j(x,y) v_j \tag{4.8}$$

where $\phi_j(x,y)$ is the shape function or interpolation function and v_j represents the value of the trial solution at the corresponding node in the element and where j is the node index. The shape function can be of any order, depending on the nature of the problem. For our case, a linear shape or interpolation function is sufficient.

4.3.2 Triangle Element Formulation

The following sources contributed to derivation below: [24, 3, 14].

Consider the trial solution in eq. 4.8. The shape function (or interpolation function) for a triangle can be described by the following general equation [also include diagram explaining interpolation function]:

$$\phi_j(x,y) = \varphi_j + \psi_j x + \omega_j y \tag{4.9}$$

The function $\phi_j(x,y)$ is a interpolation function that has the value 1 at the jth node and 0 at the two other nodes. To determine φ_j, ψ_j and ω_j in $\phi_j(x,y)$ for $j = \{1,2,3\}$ we solve for the following set of equations:

At node 1

$$\tilde{U}(x_1,y_1) = \sum_{j=1}^{N=3} \phi_j(x,y) v_j = v_1 \tag{4.10}$$

which implies

$$\begin{aligned} \phi_1(x,y) &= \varphi_1 + \psi_1 x_1 + \omega_1 y_1 = 1 \\ \phi_2(x,y) &= \varphi_2 + \psi_2 x_1 + \omega_2 y_1 = 0 \\ \phi_3(x,y) &= \varphi_3 + \psi_3 x_1 + \omega_3 y_1 = 0 \end{aligned} \tag{4.11}$$

Similar equations to eq. 4.11 can be found for nodes 2 and 3 repectively, where

$$\tilde{U}(x_2,y_2) = \sum_{j=1}^{N=3} \phi_j(x,y) v_j = v_2 \tag{4.12}$$

$$\tilde{U}(x_3,y_3) = \sum_{j=1}^{N=3} \phi_j(x,y) v_j = v_3 \tag{4.13}$$

Solving the resulting system of equations and after considerable algebraic manipulation and grouping, we get the solution

$$\varphi_j = \frac{a_j}{2\Delta}$$

$$\psi_j = \frac{b_j}{2\Delta} \tag{4.14}$$

$$\omega_j = \frac{c_j}{2\Delta}$$

where

$$\begin{aligned} a_1 &= x_2y_3 - x_3y_2 & a_2 &= x_3y_1 - x_1y_3 & a_3 &= x_1y_2 - x_2y_3 \\ b_1 &= y_2 - y_3 & b_2 &= y_3 - y_1 & b_3 &= y_1 - y_2 \\ c_1 &= x_3 - x_2 & c_2 &= x_1 - x_3 & c_3 &= x_2 - x_1 \end{aligned} \tag{4.15}$$

and

$$\Delta = Area = \frac{1}{2}\left[x_2y_3 - x_3y_2 + x_3y_1 - x_1y_3 + x_1y_2 - x_2y_1\right] \tag{4.16}$$

The shape function can then be expressed as

$$\phi_j(x,y) = \frac{a_j + b_jx + c_jy}{2\Delta} \tag{4.17}$$

4.4 Derivation of FEM Equations

It is not essential to fully understand the derivation of the FEM matrix equations for successful implementation. Of importance is the final result of the derivation (equation 4.27). However, a broad understanding of the derivation helps to maintain an overview of the method, which is very helpful when debugging software - something which can turn into a nightmare if one has been following a mindless recipe!

4.4.1 The Galerkin Method of Weighted Residuals

The method of weighted residuals works by finding values for v_j (in eq. 4.8) such that the residual $R(x)$ is as close to zero as possible for all values of x over the entire domain (Ω) [24]. From eq. 4.5 we define the residual as follows:

$$R(x,y) = \frac{\partial}{\partial x}\left(\sigma\frac{\partial U(x,y)}{\partial x}\right) + \frac{\partial}{\partial y}\left(\sigma\frac{\partial U(x,y)}{\partial y}\right) - f(x,y) = 0 \tag{4.18}$$

The residual is summed over the entire domain, and multiplied by a weighting function. For the Galerkin method, the shape functions are used as the weighting functions. For a single element, we thus consider the following integral:

$$\int\int_{\Omega} R_{\Omega}(x,y)\phi_i(x,y)d\Omega = 0 \text{ for } i = 1,2,....N \tag{4.19}$$

Next we substitute eq. 4.18 into eq. 4.19 to get

$$\int\int_{Area}\left[\frac{\partial}{\partial x}\left(\sigma\frac{\partial U(x,y)}{\partial x}\right)+\frac{\partial}{\partial y}\left(\sigma\frac{\partial U(x,y)}{\partial y}\right)-f(x,y)\right]\phi_i(x,y)dxdy = 0 \text{ for } i = 1,2,3 \tag{4.20}$$

Multiplying out, gives

$$\int\int_{Area}\frac{\partial}{\partial x}\left(\sigma\frac{\partial U(x,y)}{\partial x}\right)\phi_i(x,y)dxdy+\int\int_{Area}\frac{\partial}{\partial y}\left(\sigma\frac{\partial U(x,y)}{\partial y}\right)\phi_i(x,y)dxdy-$$

$$-\int\int_{Area} f(x,y)\phi_i(x,y)dxdy = 0 \text{ for } i = 1,2,....N \tag{4.21}$$

Using the integration by parts rule ($\int_{x_a}^{x_b}\frac{du(x)}{dx}v(x) = [u(x)v(x)]_{x_a}^{x_b} - \int_{x_a}^{x_b}\frac{dv(x)}{dx}u(x)$) and considering the first term of eq. 4.21,

$$\frac{\partial}{\partial x}\left(\sigma\frac{\partial U(x,y)}{\partial x}\phi_i(x,y)\right) = \frac{\partial}{\partial x}\left(\sigma\frac{\partial U(x,y)}{\partial x}\right)\phi_i(x,y)+\frac{\partial\phi_i(x,y)}{\partial x}\sigma\frac{\partial U(x,y)}{\partial x} \tag{4.22}$$

and after applying the same rule to the second term, and then substituting back into eq. 4.21 we get the following expression:

$$-\int\int_{Area}\left(\frac{\partial}{\partial x}\left(\sigma\frac{\partial U(x,y)}{\partial x}\phi_i(x,y)\right)+\frac{\partial}{\partial y}\left(\sigma\frac{\partial U(x,y)}{\partial y}\phi_i(x,y)\right)\right)dxdy+$$

$$+\int\int_{Area}\left(\frac{\partial\phi_i(x,y)}{\partial x}\sigma\frac{\partial U(x,y)}{\partial x}+\frac{\partial\phi_i(x,y)}{\partial y}\sigma\frac{\partial U(x,y)}{\partial y}\right)dxdy =$$

$$=\int\int_{Area} f(x,y)\phi_i(x,y)dxdy = 0 \text{ for } i = 1,2,3 \tag{4.23}$$

If we now substitute the trial solution ($\tilde{U}(x,y) = \sum_{j=1}^{N=3}\phi_j(x,y)v_j$) for all instances of $U(x,y)$ in the second integral of eq. 4.23 above, then the equation becomes (expressed below in shortened form to enhance readability):

$$\sum_{j=1}^{N=3}\left[\int\int_{Area}\frac{\partial\phi_i}{\partial x}\sigma\frac{\partial\phi_j}{\partial x}+\frac{\partial\phi_i}{\partial y}\sigma\frac{\partial\phi_j}{\partial y}dxdy\right]v_j = \int\int_{Area} f(x,y)\phi_i(x,y)dxdy+$$

$$+\int\int_{Area}\frac{\partial}{\partial x}\left(\sigma\frac{\partial U}{\partial x}\phi_i\right)+\frac{\partial}{\partial y}\left(\sigma\frac{\partial U}{\partial y}\phi_i\right)dxdy \text{ for } i = 1,2,3 \tag{4.24}$$

Notice that we take $\int\int_{Area}\frac{\partial}{\partial x}\left(\sigma\frac{\partial U}{\partial x}\phi_i\right)+\frac{\partial}{\partial y}\left(\sigma\frac{\partial U}{\partial y}\phi_i\right)dxdy$ to the other side of the equation, and we also don't substitute the trial solution for $U(x,y)$. This is done intentionally, since this term represents the flux boundary conditions (current flow). Leaving the term as it is allows us to simply substitute the boundary values from eq. 4.7. For nodes with no bounday current the term remains zero. This will become clear in the matrix equation formulation below.

4.4.2 Matrix Formulation

When we put eq. 4.24 into matrix form, we get the following system for a single element:

$$\sigma \begin{bmatrix} \int\int_{Area} \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_1}{\partial x} + \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_1}{\partial y}dxdy & \int\int_{Area} \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_2}{\partial x} + \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_2}{\partial y}dxdy & \int\int_{Area} \frac{\partial\phi_1}{\partial x}\frac{\partial\phi_3}{\partial x} + \frac{\partial\phi_1}{\partial y}\frac{\partial\phi_3}{\partial y}dxdy \\ \int\int_{Area} \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_1}{\partial x} + \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_1}{\partial y}dxdy & \int\int_{Area} \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_2}{\partial x} + \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_2}{\partial y}dxdy & \int\int_{Area} \frac{\partial\phi_2}{\partial x}\frac{\partial\phi_3}{\partial x} + \frac{\partial\phi_2}{\partial y}\frac{\partial\phi_3}{\partial y}dxdy \\ \int\int_{Area} \frac{\partial\phi_3}{\partial x}\frac{\partial\phi_1}{\partial x} + \frac{\partial\phi_3}{\partial y}\frac{\partial\phi_1}{\partial y}dxdy & \int\int_{Area} \frac{\partial\phi_3}{\partial x}\frac{\partial\phi_2}{\partial x} + \frac{\partial\phi_3}{\partial y}\frac{\partial\phi_2}{\partial y}dxdy & \int\int_{Area} \frac{\partial\phi_3}{\partial x}\frac{\partial\phi_3}{\partial x} + \frac{\partial\phi_3}{\partial y}\frac{\partial\phi_3}{\partial y}dxdy \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} =$$

$$= \begin{bmatrix} \int\int_{Area} f(x,y)\phi_1(x,y)dxdy \\ \int\int_{Area} f(x,y)\phi_2(x,y)dxdy \\ \int\int_{Area} f(x,y)\phi_3(x,y)dxdy \end{bmatrix} + \begin{bmatrix} \int\int_{Area} \frac{\partial}{\partial x}\left(\sigma\frac{\partial U}{\partial x}\phi_1\right) + \frac{\partial}{\partial y}\left(\sigma\frac{\partial U}{\partial y}\phi_1\right)dxdy \\ \int\int_{Area} \frac{\partial}{\partial x}\left(\sigma\frac{\partial U}{\partial x}\phi_2\right) + \frac{\partial}{\partial y}\left(\sigma\frac{\partial U}{\partial y}\phi_2\right)dxdy \\ \int\int_{Area} \frac{\partial}{\partial x}\left(\sigma\frac{\partial U}{\partial x}\phi_3\right) + \frac{\partial}{\partial y}\left(\sigma\frac{\partial U}{\partial y}\phi_3\right)dxdy \end{bmatrix} \tag{4.25}$$

or in matrix notation,

$$\mathbf{Y}_{i,j}\mathbf{v}_j = \mathbf{I}_i \tag{4.26}$$

for each element, where $\mathbf{Y}$ is the element matrix (also known as the stiffness matrix or resistance matrix), $\mathbf{v}$ is the voltage vector and $\mathbf{I}$ the current vector. $\mathbf{I}$ is zero everywhere except at boundary nodes where there is a net in- or outflow of current.

By using the expressions in eqs. 4.15 and 4.16 we can express the element matrix $\mathbf{Y}_{i,j}$ in eq. 4.25 in a more compact form:

$$\mathbf{y} = \frac{\sigma}{2\Delta}\begin{bmatrix} b_1^2 + c_1^2 & b_2b_1 + c_2c_1 & b_3b_1 + c_3c_1 \\ b_2b_1 + c_2c_1 & b_2^2 + c_2^2 & b_3b_2 + c_3c_2 \\ b_3b_1 + c_3c_1 & b_3b_2 + c_3c_2 & b_3^2 + c_3^2 \end{bmatrix} \tag{4.27}$$

The element matrix $\mathbf{Y}_{i,j}$has been renamed to $\mathbf{y}$ to avoid confusion.

4.5 Global Matrix Assembly

Once an element matrix has been defined for each triangle, these have to be combined to form a global master matrix.

4.5.1 Global Node Numbering

We first need to distinguish between local and global node numbering. For local node numbers, we consider all triangles of the mesh as if they were disconnected (disjointed). Each vertex of a triangle receives a unique number as for the example in figure 4.2. To solve the actual problem requires the triangles to form a connected mesh. In mesh form a single node could be the vertex of three triangles where they

are joined. For a connected mesh, we consider a global numbering scheme where each node has a unique number, as shown in figure 4.3.

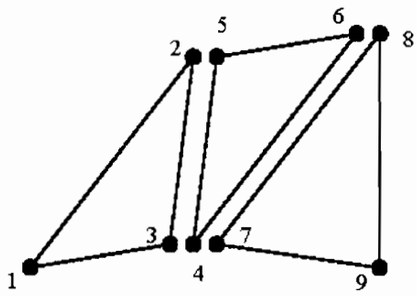

Figure 4.2: Example of local numbering

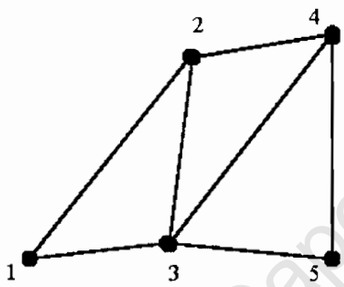

Figure 4.3: Example of global numbering

4.5.2 Matrix Assembly

Each element matrix (eq. 4.27) is combined as in eq. 4.28 to form the global master matrix $\mathbf{Y}$ of size $n \times n$:

$$\mathbf{Y}_{dis} = \begin{bmatrix} \mathbf{y}_1 & & & & \\ & \mathbf{y}_2 & & & \\ & & \ldots & & \\ & & & \ldots & \\ & & & & \mathbf{y}_n \end{bmatrix} \tag{4.28}$$

where n is the number of elements. This matrix, however, represents the master matrix in its disjointed form. The matrix needs to be modified in order to represent the connected mesh, thus we introduce a transformation matrix.

4.5.3 Transformation matrix

The transformation matrix transforms the global master matrix $\mathbf{Y}$ from its disjointed form to a connected form, by relating local node numbering to global numbering. Let us consider the transformation matrix $\mathbf{C}$ [14]. It's function can be explained as follows:

$$\mathbf{v}_{dis} = \mathbf{C}\mathbf{v}_{con} \tag{4.29}$$

In the case of the example in figures 4.2 and 4.3, the transformation matrix would operate as follows

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix}_{dis} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & 1 & & \\ 1 & & & & \\ & & & 1 & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}_{con} \tag{4.30}$$

relating the local node numbers to their global numbering convention.

4.5.4 Modified Master Matrix

A similar transformation to eq. 4.30 also applies to $\mathbf{I}_{dis}$. Equation 4.26 can also be written as

$$\mathbf{Y}\mathbf{v}_{dis} = \mathbf{I}_{dis}$$

or

$$\mathbf{C}^T\mathbf{Y}\mathbf{C}\mathbf{v}_{con} = \mathbf{I}_{con}$$

Therefore

$$\mathbf{Y}^\star = \mathbf{C}^T\mathbf{Y}\mathbf{C} \tag{4.31}$$

The master equation used for calculation then becomes

$$\mathbf{Y}^\star\mathbf{v}_{con} = \mathbf{I}_{con} \tag{4.32}$$

Instead of transforming $\mathbf{Y}$ using the transformation matrix $\mathbf{C}$, $\mathbf{Y}^\star$ can also be assembled using a recursive routine [3]. This would be faster than first compiling the master matrix and then applying the transformation. However, the author

found that the introduction of the transformation matrix aided the understanding of the main equations.

4.6 Final Equations

For the case that the injection and exit current magnitudes are known and the voltage distribution $\mathbf{v}$ is unknown, we would solve for the unknown voltages by the calculation:

$$\mathbf{v} = \mathbf{Y}^{-1}\mathbf{I}$$

However, when the boundary voltages are known and the injection and exit currents are unknown, we need a slightly different arrangement of the final equation.

4.6.1 Applying the Boundary Conditions

Let us first consider the voltage vector $\mathbf{v}$. We shall from now on refer only to the connected case. At some nodes in our mesh, the voltages are prescribed by the boundary conditions. The other majority of nodes are however "free to vary" [14]. Since they all need to be included in the same vector, we need to arrange $\mathbf{v}$ in such a way as to keep the "prescribed" and "free-to-vary" node potentials seperate. Let us enter the "free-to-vary" voltages first, and the prescribed voltages last. Then v looks like this:

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_f \\ \mathbf{v}_p \end{bmatrix}$$

The same applies to $\mathbf{I}$. Since those nodes with essential boundary conditions also have natural boundary conditions, the numbering of $\mathbf{I}$ will be the same as for $\mathbf{v}$. For the current however, we know already that $\mathbf{I}_f$ will consist of zeros only, since at any node within the region, the net current will be equal to current zero (eq. 4.2). $\mathbf{I}_p$ will contain the current at the boundary nodes where electrodes are attached and current is either injected or exits.

The modified master matrix has to match the numbering of $\mathbf{v}$ and $\mathbf{I}$, so we now have the following matrix equation:

$$\begin{bmatrix} \mathbf{Y}_{ff} & \mathbf{Y}_{fp} \\ \mathbf{Y}_{pf} & \mathbf{Y}_{pp} \end{bmatrix} \begin{bmatrix} \mathbf{v}_f \\ \mathbf{v}_p \end{bmatrix} = \begin{bmatrix} \mathbf{I}_f \\ \mathbf{I}_p \end{bmatrix} \tag{4.33}$$

4.6.2 Solving for unknown values

Equation 4.33 can also be expressed as two seperate equations:

$$\mathbf{Y}_{ff}\mathbf{v}_f + \mathbf{Y}_{fp}\mathbf{v}_p = \mathbf{I}_f \tag{4.34}$$

and

$$\mathbf{Y}_{pf}\mathbf{v}_f + \mathbf{Y}_{pp}\mathbf{v}_p = \mathbf{I}_p \tag{4.35}$$

Now, if the boundary voltages are known beforehand and we would like to calculate the unknown boundary currents for a given conductivity distribution, we first rearrange eq. 4.34 to solve for $\mathbf{v}_f$:

$$\mathbf{v}_f = \mathbf{Y}_{ff}{}^{-1}(\mathbf{I}_f - \mathbf{Y}_{fp}\mathbf{v}_p) \tag{4.36}$$

and then subsitute the solution of $\mathbf{v}_f$ into eq. 4.35 to get $\mathbf{I}_p$.

Chapter 5

The Inverse Problem

Since the governing equation (eq. 4.4) cannot be analytically solved for conductivity σ, it becomes an inverse problem. Several methods of approaching this problem exist, however Yorkey *et al* [19] found the Newton-Raphson Method to yield the best results among several such inverse methods. The method has since proved a popular method among authors in solving EIT problems [21, 20, 25, 13]. It is for this reason that the author chose to use the Newton-Raphson Method to solve the inverse problem.

5.1 The Newton-Raphson Method

The Newton-Raphson (NR) method can be compared in principle to Newton's method. Consider a function as shown in figure 5.1 and suppose that we would like to find the x-value of the global minimum. We start by making a guess for the x-value, then we take the tangent at that point and update our new guess to the x-value where the tangent cuts the x-axis. Then the same is repeated for the updated guess, until the guess converges to the true solution, or some value close enough.

As with any iterative method, convergence might not be possible for various reasons. For one, a solution might not exist, for example if measurement errors are present. Regularisation of the data seeks to address this problem (see section 5.2 below) . Another common reason for non-convergence is a bad initial guess, if for example, the initial guess causes the iterative algorithm to converge to a local minimum of the particular curve instead of the global minimum.

5.1.1 Derivation

The NR method recognises the fact that we cannot solve for the conductivity distribution of a region directly. Thus the approach is to start with an arbitrary

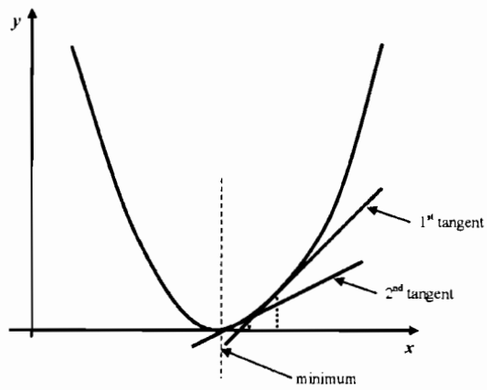

Figure 5.1: Example of Newton's Method. The figure shows a parabola, whose minimum we would like to find using Newtons Method.

conductivity distribution guess. For any specified conductivity distribution, it is possible to calculate the voltage or current response by solving the forward problem using FEM.

Since we can measure the actual current response for our region, it might seem plausible that one could compare the measurements due to our known conductivity, to those based on a conductivity guess. It should thus be possible, using an algorithm, to find a conductivity distribution which results in a similar current response (determined by FEM) to our measured currents. This assumes a one-to-one mapping between conductivity and the resulting current measurements.

We define the mean square error between measured and computed current measurements [3, 19] as:

$$\Phi(\sigma) = \frac{1}{2}(f(\sigma) - \mathbf{I}_0)^T(f(\sigma) - \mathbf{I}_0) \tag{5.1}$$

where $f(\sigma)$ is the computed response for a conductivity distribution σ and $\mathbf{I}_0$ are the actual current measurements. The idea is to find the values for σ that minimize the error function $\Phi(\sigma)$.

To do this we set the derivative of Φ to zero,

$$\Phi'(\sigma) = [f'(\sigma)]^T(f(\sigma) - \mathbf{I}_0) = 0 \tag{5.2}$$

where $[f'(\sigma)]_{i,j} = \frac{\partial f_i}{\partial \sigma_j}$ is called the Jacobian matrix.

Next, we take the Taylor series expansion of $\Phi'(\sigma)$ about σ_k,

$$\Phi'(\sigma_{k+1}) = \Phi'(\sigma_k) + \Phi''(\sigma_k)\Delta\sigma_k = 0 \tag{5.3}$$

keeping only the linear terms. Note that

$$\sigma_{k+1} = \sigma_k + \Delta\sigma_k \tag{5.4}$$

where $\Delta\sigma_k$ is the conductivity update.

We call the second derivative $\Phi''(\sigma^k)$ the Hessian matrix.

$$\Phi'' = [f']^T f' + [f'']^T \{\mathbf{V} \otimes [f - \mathbf{I}_0]\} \tag{5.5}$$

The $\otimes$ is the Kronecker matrix product and is omitted, since f'' is relatively small and also difficult to calculate. Thus the second derivative is used as follows,

$$\Phi'' = [f']^T f' \tag{5.6}$$

If we substitute eqs. 5.3 and 5.6 into the Taylor expansion (eq. 5.3) and change the subject of the equation, we get the following update equation for σ,

$$\Delta\sigma_k = -\left[[f'(\sigma_k)]^T f'(\sigma_k)\right]^{-1} \left[[f'(\sigma_k)]^T [f(\sigma_k) - \mathbf{I}_0]\right] \tag{5.7}$$

In fact, the above method is called the modified Newton-Raphson method. A flow chart of the algorithm is shown in fig. 5.2.

5.2 Ill-conditioning and Regularisation

The inverse problem is by nature ill-conditioned, which effectively means that large changes in the interior conductivity distribution cause comparatively small changes in the boundary measurements. The degree of ill-conditioning also depends on the measurement strategy and whether a large number of linearly independent measurements of the rig interior are made. Unfortunately the geometry of the rig chosen for this thesis limits boundary measurements to the sides only, whereas other tomography systems using circular rigs are able to make measurements all around.

This makes our system very sensitive to noise or measurement errors, or even small discrepancies between the real and modelled rig. One can also say that the boundary measurements are not sensitive enough to conductivity changes in some parts of the region. This is refered to as low sensitivity. The Jacobian in the Hessian matrix (eq. 5.6) is in effect a sensitivity matrix, since it contains information on the sensitivity of the measurements with respect to change in interior conductivity. In the update equation (eq. 5.7) the inverse of the Hessian is computed. If small errors due to measurement or other factors have been introduced, the inversion will be inaccurate, causing large reconstruction errors.

One very popular way of stabilising such an ill-posed system is by regularising the data, or introducing a stabilising bias to the data. Regularisation is a manual and

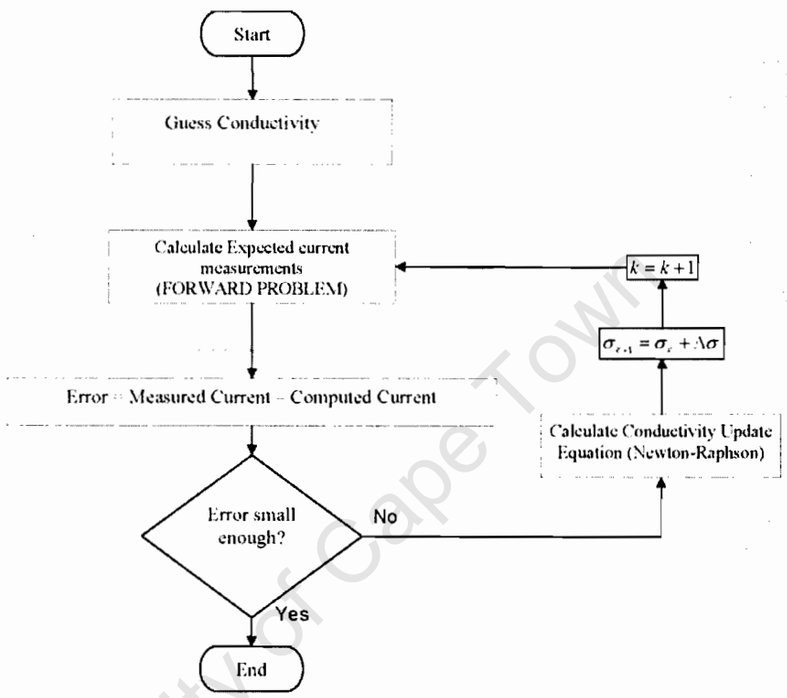

Figure 5.2: Newton-Raphson algorithm [3]

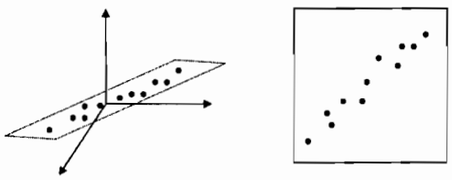

Figure 5.3: A set of partially correlated data. The graph on the left shows the data points centered around the origin, while the diagram on the right shows the same points in a top view.

somewhat crude interference, but it works exceptionally well. The technique has been perfected and various adaptations exist, but we will consider basic regularisation (or ridge regression), which returned satisfactory results the first time round.

A very effective and intuitive explanation has been put forward by J. R. Greene [26], since it captures the essence of regularisation better than highly theoretical explanations.

5.2.1 A Simplified Regularisation Example (Ridge Regression)

Consider a set of data points $\mathrm{Z} = \{x, y\}$, as shown in figure 5.3. The same data can also be recorded in a matrix:

$$\begin{array}{ccc} x & y & Z \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \ldots & \ldots & \ldots \\ a_n & b_n & c_n \end{array}$$

The data is centred about the origin. One should be able to see from figure 5.3 that the data lies more or less around a line and is not evenly spread out, i.e. the data is partially correlated. Or, in other words, the x- and y- values are "approximately linearly dependent". Imagine we would like to fit a plane to that set of data. Also consider the case if the x- and y-values contained some noise errors. Due to the data points being positioned more or less around a line, that plane would be very unstable, and a well defined tilt for the plane could not be found. It should be appreciable that the effect of the noise error on the tilt of the plane introduces a significant error.

One way to achieve an error-free tilt of the plane would be to eliminate errors completely, but since that is only attainable in an ideal system that is not an option. The regularisation approach would be to introduce a bias to the data, by manually adding some data points. Consider adding data points on each axis, close

to the origin. The modified data set would then look as follows:

$$\begin{matrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \dots & \dots & \dots \\ a_n & b_n & c_n \\ \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{matrix}$$

λ being a small constant, typically to the order of 10^{-3} for normalised data [26]. Although this modification could also introduce potential errors, the new data points would have the effect of stabilising the tilt of the plane by reducing the slope in all directions, as long as λ remains small. For directions in which the plane is strongly defined, the regularisation constant would have very little effect, but it would reduce the effect of small data errors. Overly large a value for λ would distort the data.

5.2.2 Regularising Tomographic Data

Regularising our tomographic data can also be explained in the context of the plane-fitting example above. Our data set consists of numerous current measurements for a particular conductivity distribution, $\sigma = \{I_1, I_2, I_3, ..., I_n\}$. The particular conductivity distribution is an unknown, for which we only have the current measurements. We can find ideal measurements for any σ using the FEM code, but we can not solve for σ analytically. Thus the inverse solution attempts to find a σ whose set of expected ideal measurements $\{I_1, I_2, I_3, ..., I_n\}$ closest match the real measurements. This is analogous to "fitting the plane".

We will modify eq. 5.1 to include the regularisation parameter:

$$\Phi(\sigma) = \frac{1}{2}(f(\sigma) - \mathbf{I}_0)^T(f(\sigma) - \mathbf{I}_0) + \lambda\Sigma \tag{5.8}$$

where Σ is the identity matrix.

The update equation (eq. 5.7) then becomes:

$$\Delta\sigma_k = -\left[[f'(\sigma_k)]^T f'(\sigma_k) + 2\lambda\Sigma\right]^{-1}\left[[f'(\sigma_k)]^T[f(\sigma_k) - \mathbf{I}_0]\right] \tag{5.9}$$

As can be seen, this results in the regularisation parameter being added to the diagonal of the Hessian matrix $\left[[f'(\sigma_k)]^T f'(\sigma_k)\right]$.

The effect of varying magnitudes for λ on image reconstruction has been explored and is discussed in the chapter on results (chapter 8) .

5.3 Expected Resolution for Reconstruction

Each conductivity unknown σ_k represents one "pixel" on the final image. The resolution of the reconstructed image is determined by the number of unknowns that can be solved for by the Newton-Raphson method. This depends on the number of linearly independent measurements taken by the measurement strategy. In order to converge to a solution, the Hessian $\left[[f'(\sigma_k)]^T f'(\sigma_k)\right]$ in eq. 5.7 must be of full rank [19].

It was found empirically that the highest number of unknowns that still result in a consistent full rank for the Hessian was 16 (with regularised data only). Thus the highest vertical resolution for image reconstruction that the particular measurement strategy supports is 16 levels for the electrode configuration and measurement strategy employed for this thesis.

Chapter 6

Software

6.1 Platform

The platform of choice is MATLAB. Since the ERT system is still in a research stage, MATLAB seemed to offer all the tools to facilitate effective implementation, such as a powerful matrix computation engine and an abundance of useful functions. This allowed the author to focus on the project while being able to rely on a well-established research platform. It was always kept in mind that a practical online implementation of such a system would require a more optimized platform than MATLAB.

6.1.1 General

The purpose of the software was not to develop an online or real time tomography system. The structure and extent of the software was developed in such a way as to allow for practical exploration of the possibilities and implications of using ERT for measuring a settling process.

Basic Operation

Circuit measurements can be controlled and read in MATLAB, and are saved in memory before the reconstruction code is applied to them. The next step is calibration, for which suitable routines have been written (chapter 7. The calibrated data is then fed to the Newton-Raphson routine, which iteratively computes a conductivity distribution employing a combination of the inverse and forward solutions.

Execution Speed

In the beginning some attention had to be given to code optimisaton in order to improve execution speed. For example, it was realised that sparse matrix methods

improved the execution time of the finite element code significantly, from the order of minutes to a couple of seconds. Subsequent calculation of the Jacobian increased execution time by several orders. Some ways of improving this were explored and are mentioned below. However, it should be noted that since code optimization wasn't the main aim of this thesis, the exploration of faster methods was only pursued if experimentation became impractical due to excessive execution times. Of course the development of a real time system would require further attention to be given to code optimization.

6.2 Newton Raphson Main Routine

The main calling routine for reconstructing an image from measured data is the function called *newt_rap*. It calls all other functions that are needed for reconstruction. Input arguments to the *newt_rap* function are an initial conductivity guess, as well as current measurements from the circuit for a particular state of the rig and its contents. Figure 6.1 shows a basic overview of the main code and its subroutines.

6.2.1 The Jacobian Subroutine

The Jacobian is an important input to the conductivity update equation. The *jacob_nr* subroutine is very computationally intensive, since iterative solutions of the finite element code are required to make up the matrix. For the case that only one iteration of the Newton-Raphson algorithm is computed [27], a popular approach has been to pre-compute the Jacobian, store it in memory, and retrieve it for the first iteration. This speeds up the algorithm significantly and is very useful in an online system, where measured data is fed to a reconstruction algorithm in real time. Since the author experimented with different numbers of iterations and other parameters, pre-computing of the Jacobian was not integrated as a permanent feature in the code.

6.2.2 The Update Equation Subroutine

The *update_sigma* subroutine has as its input the Jacobian matrix, the most recent update to the conductivity guess, the measured current data as well as the expected current data calculated by the finite element code, based on the last conductivity update. It computes eq. 5.7, adds the results (updates) to the conductivity guess, and returns the new conductivity guess to the main *newt_rap* routine.

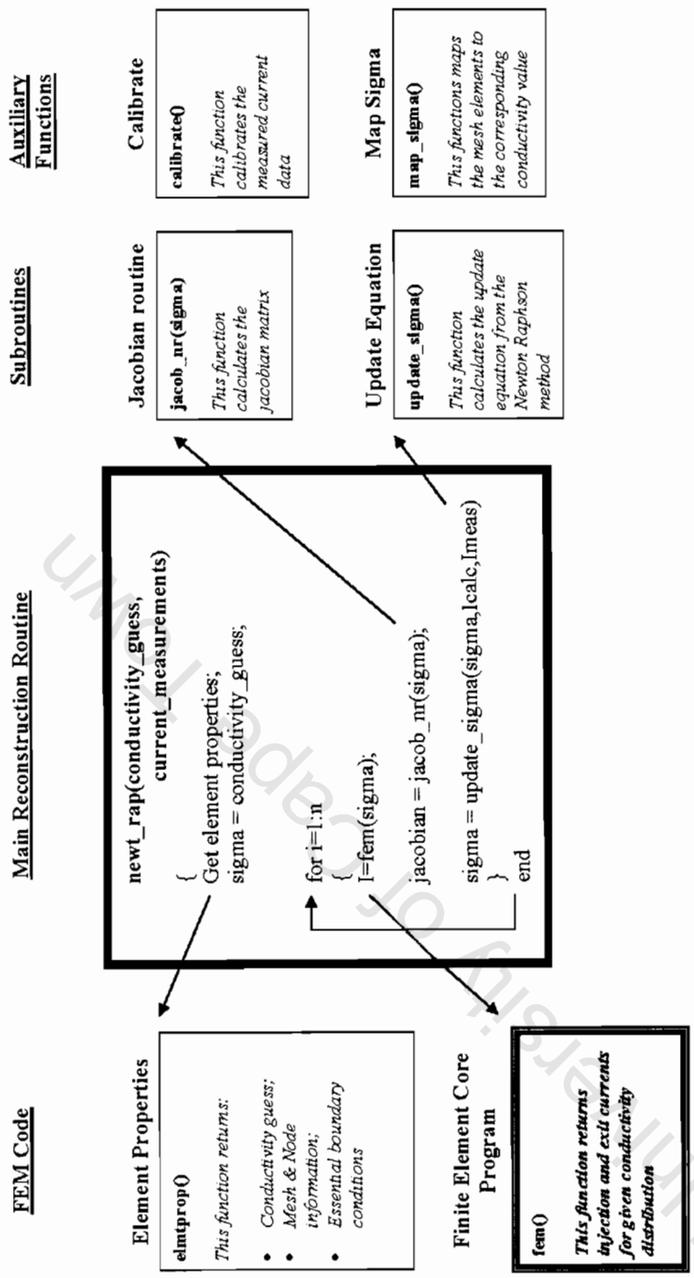

Figure 6.1: Basic code overview.

6.3 Finite Element Code

Initially, the author investigated various commercial and open source software packages to implement the finite element method. An obvious option was the commercial package FEMLAB from *Consol*, which runs on the MATLAB platform. It is a very extensive package and was seriously considered. Other options were the open source software FEMM [28] and EIDORS [29] (An open source package compiled by various researchers).

After some contemplation, it became evident that each of these packages still required a sound understanding of the finite element method since they would need to be modified to suit this thesis. Despite this the learning curve for each software package itself seemed steep. At the time, the author found that resources would be better spent acquiring a more in-depth understanding of the finite element method than spending that amount of time learning how to use new software. It was in this spirit that the author opted to write the FEM code from scratch. In retrospect, the knowledge and experience gained seem highly valuable and time well-spent, although opting for an already established package might have yielded good results in less time.

A first attempt at writing the code from scratch yielded good results, but the code was bulky and had no inherent structure, and thus had to be improved. It was rewritten at a later stage, based in principle on the structure introduced by Victor Balden [24]. The author even extended the functionality of the code in order to use square elements instead of triangle elements. This involves Gauss-Legendre integration in order to formulate the shape functions for the square elements. However, it turned out that this functionality wasn't necessary for the purpose of this thesis, therefore this aspect has not been explained in this document. The full code is printed in the appendix, however.

6.3.1 Defining the mesh

Meshing Tool in Matlab One major part of any modelling or simulation, is the digital description of the real "thing". With the finite element method this is done by describing the geometry and dividing it into elements. Matlab offers some basic triangle meshing capability as part of its PDE toolbox. MATLAB's *pdetool* was used to define the geometry of the rig. Since we were assuming two-dimensionality, a rectangle corresponding to the measurements of the rig was defined. The function "poimesh" generates a regular triangle mesh for rectangular geometries and returns the mesh description data. A regular mesh is important in this case, since we are interested in a vertical conductivity distribution. It is thus useful if the nature of the conductivity distribution is reflected in the mesh structure.

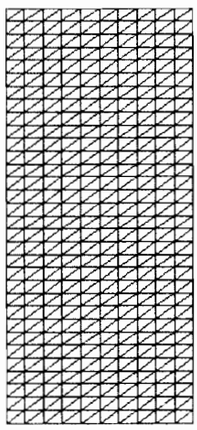

Figure 6.2: Triangular mesh for the settling rig showing the triangles that discretise the interior of the rig. Each triangle is defined by its three node numbers and their respective coordinates, as shown in figure 4.1 on page 18.

The mesh is shown in fig. 6.2. There are 32 vertical levels and 20 triangles for each horizontal level. The vertical resolution was chosen to be divisible by 16, for reasons already stated (see section 5.3 on page 33). It is a rather fine mesh for our purposes. However, the need to optimise didn't arise, since code execution speed wasn't a problem at this stage. In addition, a finer mesh improves modelling accuracy.

Mesh Description Data The mesh information is described by means of three matrices, called P, E and T. They hold information on the node coordinates, node numbering, triangle data and element numbering. The conventions are defined under MATLAB and was further adapted for this thesis. Of the three matrices, only P and T are of significance for our purposes. P holds the point, or node data, while T contains information on the triangles and element numbers.

For our problem, the author chose to add some information to the respective vectors. P, holding node data, was further extended to hold information about current injection and exit nodes, as well as the voltage boundary conditions This is explained by means of table 6.1. T was extended to also contain the conductivity for each element. Examples for the use of the P and T vector are shown in tables 6.1 and 6.2 below.

column no.	1	2	3	4	5
row 1: x-coordinate	0	0.7	0.6	1.8	1.8
row 2: y-coordinate	0	1	0.1	1.2	0
row 3: type of boundary cond.	0	1	0	0	1
row 4: boundary condition value	0	4.2	0	0	0

Table 6.1: Example of node data (P matrix).

The example in table 6.1 refers to figure 4.3 on page 23. Here node 2 at $(0.7, 1)$ has a voltage of 4.2V applied to it, while node 5 at $(1.8, 0)$ is grounded. The other nodes have no boundary conditions defined. The node numbers are implicitly defined as the column numbers of the P matrix.

column no.	1	2	3
row 1: node 1	1	3	3
row 2: node 2	2	2	4
row 3: node 3	3	4	5
row 4: subdomain no. (not used)	-	-	-

Table 6.2: Example of element data (T matrix).

An example of the element matrix for figure 4.3 is shown in table 6.2. The first three describe the three nodes that make up each element. The node numbers are a reference to the column numbers in the P matrix, while the element number is implicitly described by the column number of the T matrix.

6.3.2 The FEM Code

The finite element method is implemented as described in chapter 4. MATLAB's advanced matrix indexing capabilities were exploited to manipulate the large data structures. Figure 6.3 shows the basic structure of the finite element code by means of a flowchart. The code follows the steps introduced and explained in chapter 4. The full code is printed in the Appendix.

6.4 Auxiliary Functions

Before the data is used in calculations, it has to be calibrated and put into the format which will be used in the calculations. Several seperate auxiliary functions were written for this purpose.

6.4.1 Calibration

The *calibration* function serves the purpose of calibrating raw data from the measurement circuit and is discussed in more detail in the next chapter.

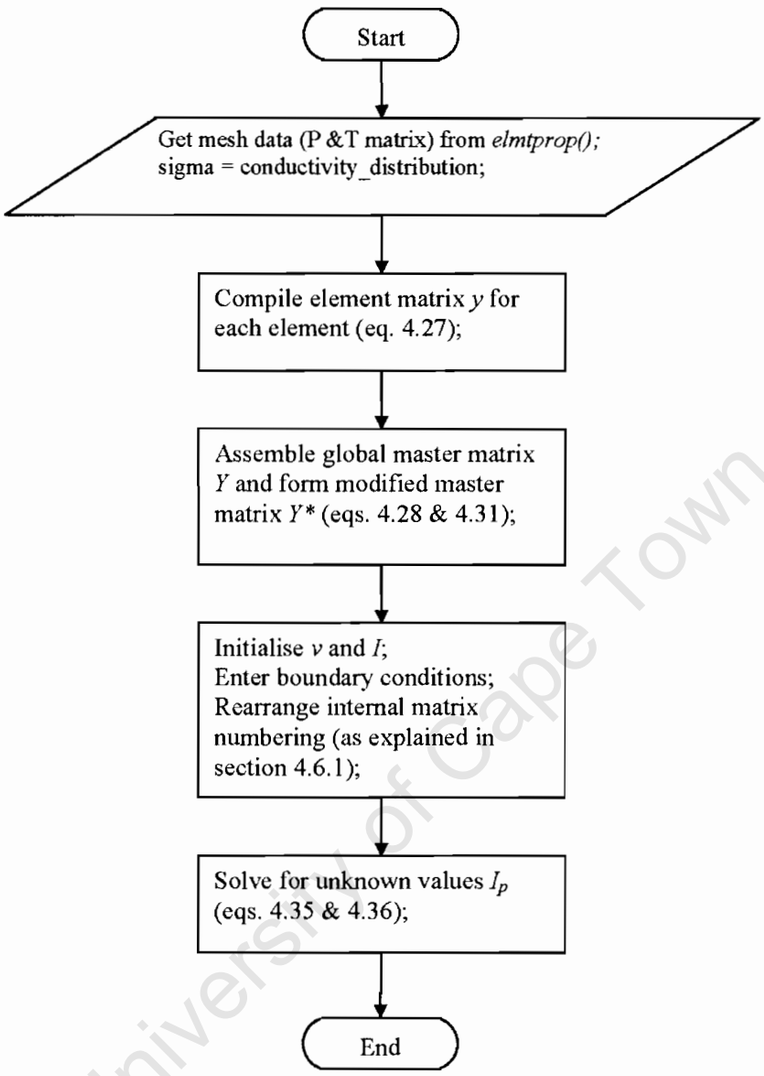

Figure 6.3: Flowchart depicting FEM code structure.

6.4.2 Parsing

The *hard2soft* function serves the purpose of parsing data from the measurement hardware into a form useful to the reconstruction code, as it is not in the correct form when it is read into the PC and saved by the measurement routine. This subroutine takes into account matrix numbering conventions and configurations as required by the reconstruction code.

6.4.3 Mapping function

The *map_sigma* function is used to map the conductivity values used by the *newt_rap* function to their corresponding element numbers as used in the FEM code. This is necessary since the reconstruction resolution is different to the mesh resolution. The so-called "mesh grouping" that is performed by this function is principally the same as in the mesh grouping method used by Kyung Ho Cho *et al* [25].

Chapter 7

Acquiring Real Data

7.1 New Rig

7.1.1 Comparing real and modelled data

The task of verifying the accuracy of the foward problem solver involved comparing the computed results with real, measured data. Using the obvious test case of a homogenous medium (tap water in this case) a set of measurements was taken from the rig shown in fig. 3.2, while another set was computed using the two-dimensional finite element code. The two sets did not match even slightly.

Measurement and circuit errors were ruled out after comprehensive testing of the hardware. The author was also quite confident that the code delivered correct results. There was a strong suspicion that the two-dimensional FEM model was an oversimplification of the system, and that this was the reason for the strong discrepancy.

In the two-dimensional model, current fringing into the third dimension is not modelled by the code. Since the electrodes used were point electrodes, fringing into the third dimension was significant. Figure 7.1 demonstrates the effect of current fringing for a three-dimensional system and shows how the two-dimensional case can be approximated by minimising current fringing and using line electrodes to achieve this.

In order to verify this suspicion, the area to the sides of the electrodes was filled with polystyrene foam to force current into a narrow path in line with the electrodes, and thereby approximating the two-dimensional case. The measurements for this experiment closely matched the computed ones within a 6% error margin, thus confirming the suspicion.

In an effort to address the problem, the possibility of upgrading the FEM code to a three-dimensional system was investigated. It soon became clear that this was

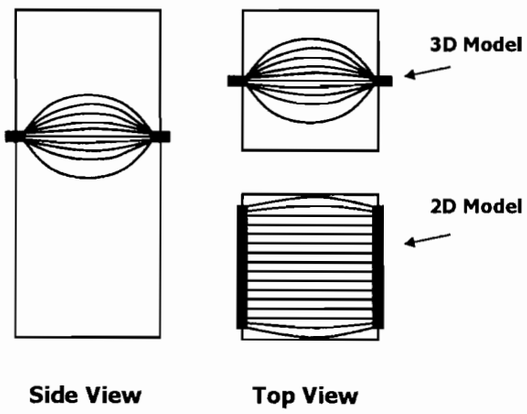

Figure 7.1: Current fringing. The diagram shows a side view of the current lines as the are assumed to be for both the 3-D and 2-D case. The fringing effect that differentiates the 3-D case from the 2-D case becomes evident in the top view, where the current lines for point electrodes and line electrodes are shown.

not an attractive route to follow, since adding a third dimension meant a significant increase in the execution speed of the code, from currently a few seconds, to the order of several minutes. This was not desired, and another solution seemed possible.

7.1.2 Improved rig

Another possibility to achieve matching results was to make the real rig more similar to the simulated one. This would involve minimizing the current fringing between the electrodes. One way of doing this is to use long electrodes, causing the current to flow in parallel lines and only having a minimal fringing effect on the sides. Figure 7.1 also shows the current lines for a rig using long electrodes, and it can be observed that the fringing effect is minimal. The new rig design is shown in fig. 7.2.

Fig. 7.3 shows a photo of the new rig after it was built.

7.2 Data Calibration

The current measurements from the settling rig are not ideal and contain sources of errors due to component tolerance, electrode placement and stray resistance and capacitance. Small errors can have a large effect on reconstruction accuracy (see section 5.2 on ill-conditioning). Therefore it is desirable to perform some sort of

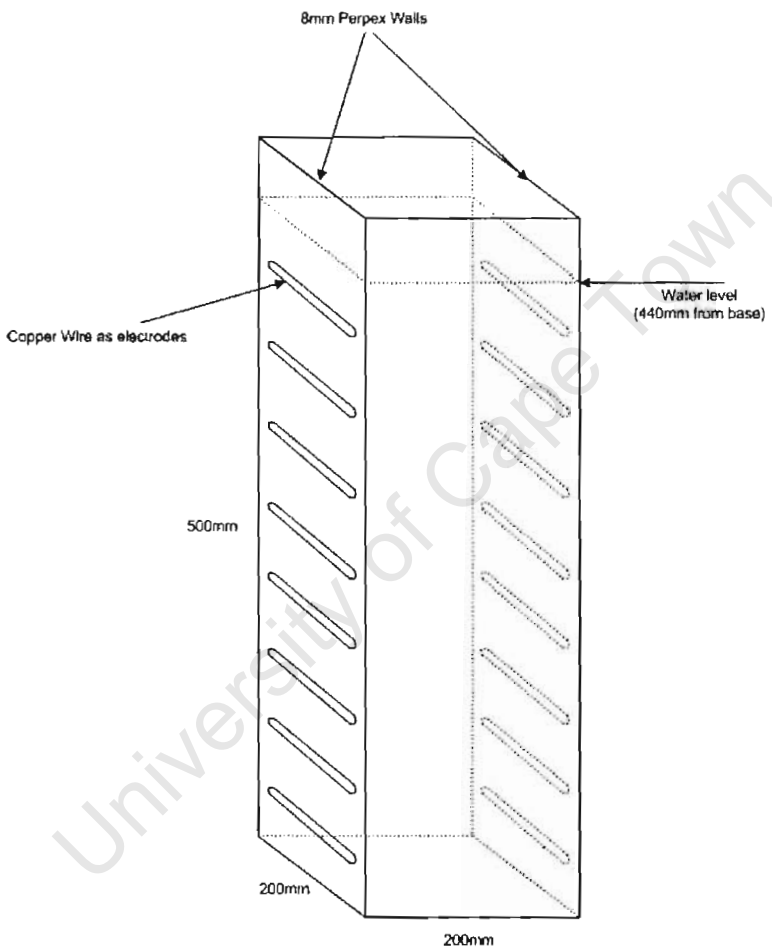

Figure 7.2: New rig design

Figure 7.3: Picture of new rig on a scale of 1:8.

calibration to the data once it is recorded in digital form, in order to reverse the effect of such errors as good as possible.

As a benchmark, we use the ideal current response as computed by the FEM code. The current response for a homogenous medium is computed. A real set of measurements for the rig are made, where the rig is filled with tap water (a homogenous medium). The measurements from the real rig are recorded in an 8x8 matrix, where the rows are measurements due to the same current source and the eight grounded electrodes are varied. The columns contain measurements from the same grounded electrode, with the current injecting electrodes varied between the eight sources.

For calibration coefficients we introduce 24 unknowns. 16 of these are weighting factors that are supposed to reverse the effect of gain errors introduced by the eight measurement channels and the eight current injecting electrodes. The remaining eight unknowns are offset constants that cater for offset errors introduced by the current measurement channels (connected to the grounded electrodes) and the AD conversion. The idea is that the calibration coefficients will reduce the effect of the abovementioned error sources, thus contributing to better reconstruction results. See fig. 7.4 for a depiction of these coefficients.

7.2.1 Calibration Code

Computation of unknowns We regard the computed current measurements as ideal, and solve for the unknown calibration coefficients such that they make

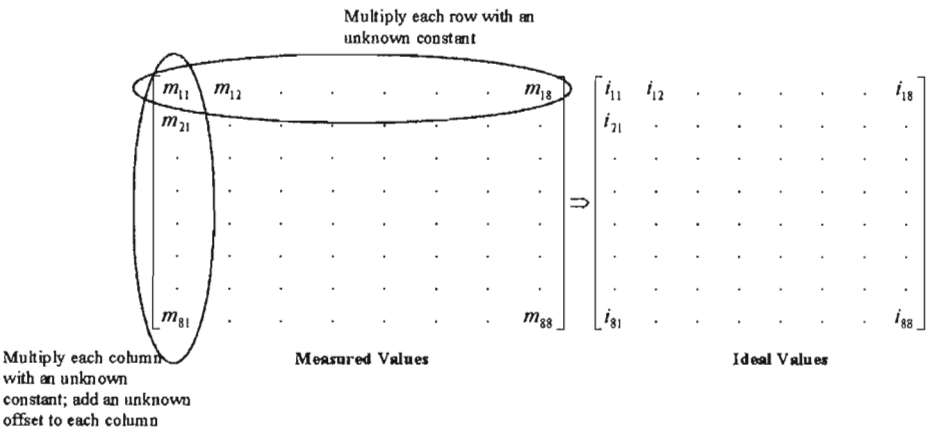

Figure 7.4: Explanation of calibration coefficients showing how the arrangement of electrodes are reflected in the rows and columns, and how this determines the choice of calibration coefficients. Each set of measurements or "capture instant" is recorded in an 8x8 matrix.

the real measurements match the ideal values as closely as possible. Solving for the unknowns was achieved using a breeder genetic algorithm [30] where the aim is to find values for the unknowns that minimize the error between the normalised calibrated data and normalised ideal data as computed by the FEM code. A breeder genetic algorithm written by J. R. Greene [31] was used and adapted only slightly. The subroutine is called *calibval* and has as input the uncalibrated data and the ideal data, each in the format of an 8x8 matrix. The subroutine *calibfit* is part of the code and has the function of calculating the mean-squared error between the ideal data and calibrated data. The main routine *calibval* then attempts to find the calibration coefficients that result in the least error between calibrated and ideal data. The code is printed in the appendix for completness.

The *calibrate* routine calls all the necessary subroutines to perform calibration. It takes as input the raw uncalibrated data and returns an average of the 10 best sets of coefficients. It was found that averaging several solutions of the breeder genetic algorithm in fact returned better results than any single solution. However, this can be interpreted as just another way of "breeding" solutions and could have been incorporated into the original breeder code.

Auxiliary Function The function *soft2hard* is used by the *calibration* routine and parses current measurements that are returned from the FEM code into an 8x8 matrix, which is the same format as that of the stored data which is sampled from the measurement hardware.

Calibration Using the calculated calibration coefficients, the function *adjust* performs the actual calibration or "correction adjustments" on the raw data. Inputs are the uncalibrated 8x8 matrix as well as the calibration coefficients. The function returns an 8x8 matrix with the calibrated data.

7.2.2 Calibration Results

Figure 7.5 shows a graphical representation of normalised uncalibrated data for the special case of a homogenous medium inside the rig (tap water), and next to it the normalised ideal data as computed by the finite element code. The shape of the graphs do not mean anything - the graphs are merely a graphical representation of the data, depicted by means of points joined by lines. This representation was chosen because it is easier for the human mind to grasp basic features of a large data set by visual means, than having the tedious task of reading through a large number of floating point values.

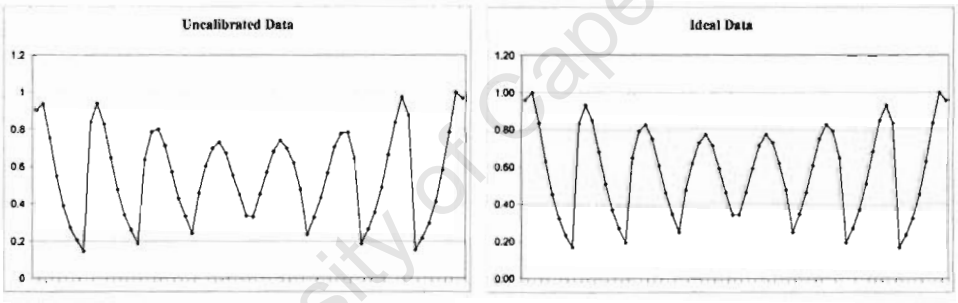

Figure 7.5: Uncalibrated and Ideal Data. The shape of the graphs are not what we are interested in - the graphs merely allow for convenient comparisons between different data sets (see figures 7.6 and 7.7 below). The data points are joined by lines to provide for a visual overview of the data.

Figure 7.6 shows the same data sets overlapping each other. It can be seen that the real uncalibrated data is similar to the ideal data set, however with some errors.

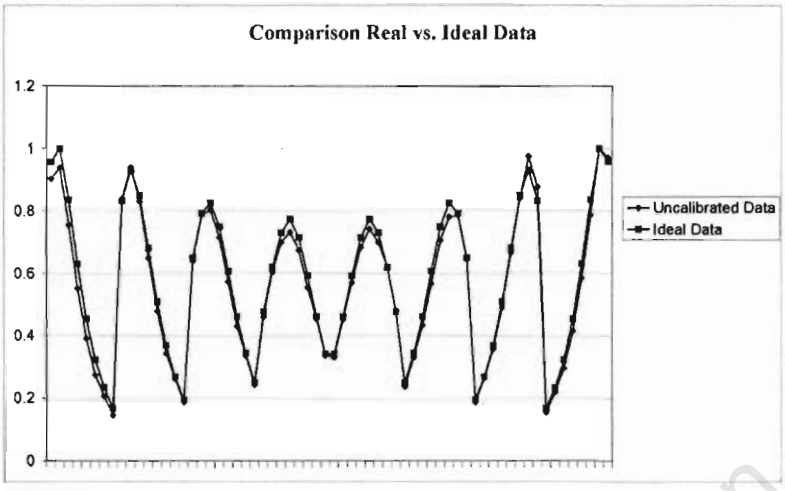

Figure 7.6: A comparison between raw uncalibrated data and an ideal data set.

After solving for the calibration coefficients and calibrating the raw data, the numbers match the ideal data closely. An overlapping comparison between the calibrated and ideal data is shown in figure 7.7. As they match closely, it is difficult to distinguish between the two sets on the graph.

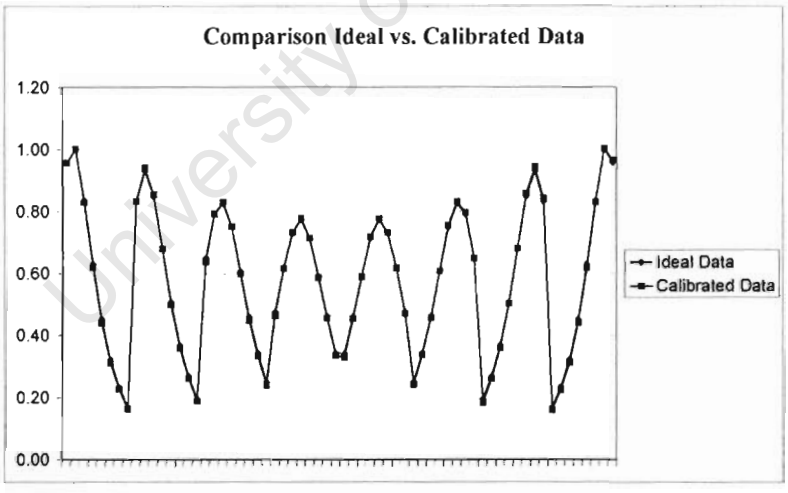

Figure 7.7: A comparison between calibrated data and an ideal data set.

Chapter 8

Results

This chapter introduces and discusses the results obtained from the system as laid out in the previous chapters. Several sets of measurements of a settling slurry have been made, in order to show how the reconstruction results relate to what is visually happening inside the rig. The effect of varying degrees of regularization is discussed and an empirical demonstration of the maximum resolution for reconstruction is included.

8.1 Regularisation Parameter

Due to the inherent ill-conditioning of the data, regularising the data was an important part of reconstruction (see section 5.2). Since the optimal regularisation parameters depend on the solution requirements, such as the degree of smoothing required, an optimal range of values had to be found by experimentation. Please note that the magnitude of regularization parameters λ in this text apply only to measurement data normalised to values between 0 and 1.

Figure 8.1 shows a typical reconstruction result for the slurry in a settled state. The colour map is only one possible visual representation of the reconstruction result. A graph of the same results is shown in figure 8.2.

The results in figures 8.1 and 8.2 use a regularisation parameter of $\lambda = 3$ (see subsection 5.2.1 for further explanation). It was found that a λ of this magnitude provided satisfactory results.

Figure 8.3 shows the reconstruction results for the same data, using different regularisation parameters $\lambda = 0.05$ for the graph on the left and $\lambda = 0.1$ for the graph on the right. It can be observed that the reconstruction results become highly irregular and the desired conductivity pattern is almost unrecogniseable.

Too much regularisation is also not desirable, as shown in figure 8.4. Here the desired parameter $\lambda = 20$ was used on reconstruction. It can be seen that especially

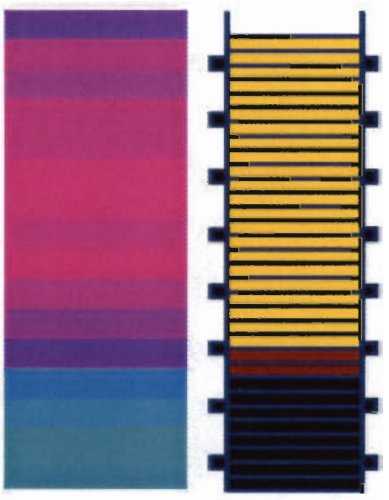

Figure 8.1: Typical reconstructed solution for a settled slurry in the rig. The diagram on the left shows the reconstructed conductivity profile of the rig contents. Light blue colours represent low conductivity values, while the darker purple colours represent higher conductivity values on the other end of the scale. The diagram on the right is a drawing of the actual settled state of the slurry, as was measured and recorded at the same time as each data capture.

on the edges (top and bottom region of rig) conductivity information is lost.

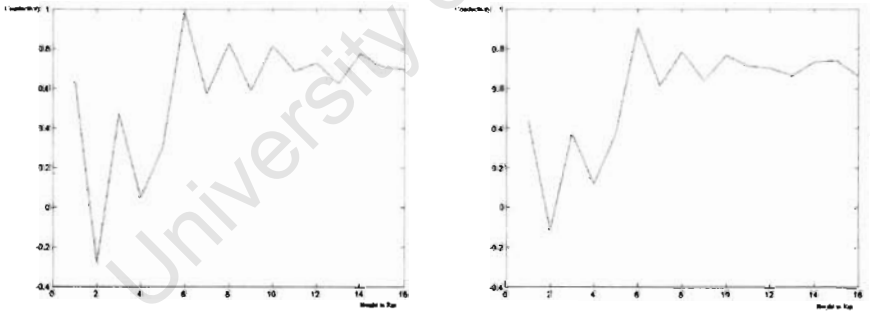

Figure 8.3: Reconstruction of the same data, using $\lambda = 0.05$ and $\lambda = 0.1$ respectively. Both results are quite distorted and a regular conductivity profile is barely evident. This is due to the ill-posedness of the data and a lack of regularisation.

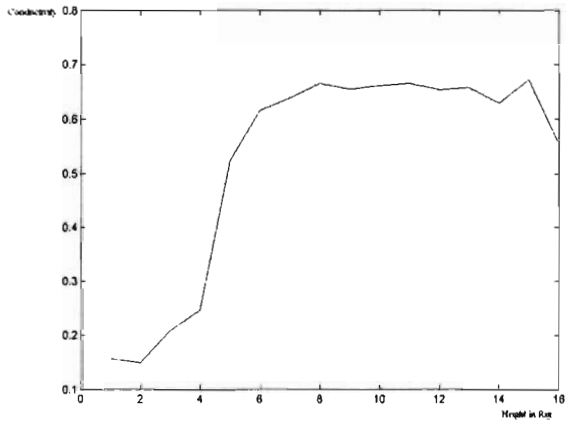

Figure 8.2: Graphical representation of the conductivity distribution in figure 8.1. The x-axis corresponds to height in the rig. Low x-values refer to the bottom of the rig, while high x-values refer to the top of the rig. The y-axis corresponds to conductivity. This reconstruction was done with $\lambda = 3$.

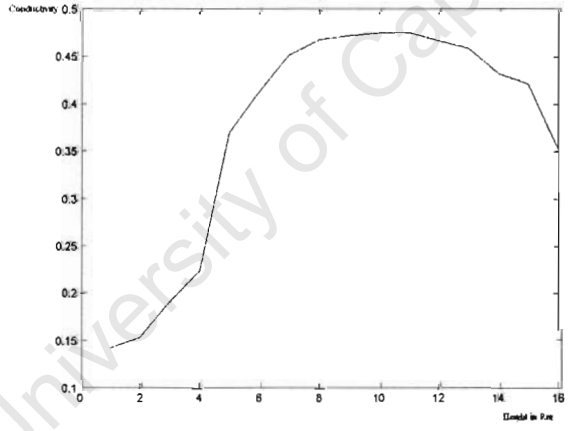

Figure 8.4: Reconstruction of the same data, using $\lambda = 20$. The data becomes "too smooth" and some of the information about the conductivity as it changes with height is lost. Also note how the range of the y-axis has become very small compared to the well-regularised figures.

8.1.1 Discussion

Higher values of λ has the effect of smoothing the data to the point that information is lost. It is also important to note that small inaccuracies in the data cause highly irregular results when no regularization is applied, thus it is useful to be able to "desensitise" the reconstruction process to measurement errors to the extent that the data becomes useable. Therefore the degree of regularisation depends strongly

on the unique properties of the problem, and on the particular requirements for reconstruction.

The value of $\lambda = 3$ is still quite high if compared to similar cases in literature [19, 26, 20, 3]. The author attributed this to the nature of the measurement strategy employed. Since several measurements made are linearly dependent, the system is overdetermined. As the reconstruction algorithm in principle searches for a "best fit" conductivity profile, it could be argued that small changes to the data do not affect reconstruction results significantly, thus the need for higher regularisation parameters.

It should also be noted that regularisation smoothes the results for conductivity in the centre region of the rig, while slightly distorting the results for top and bottom conductivity, as can be observed in figure 8.4. This might be because the centre part of the rig is well defined from the measurements, while the sensitivity of the measurements for the top and bottom regions is significantly less, as not much current flows through these regions. Where the sensitivity of the data is low due to the measurement strategy the high value for λ is shown to distort the results even more.

The many redundant measurements contribute a lot of noise and errors as well, making the system slightly unstable and thus reconstruction without any form of regularisation seems impossible. Therefore, the an optimal value had to be found by experimentation. For the remainder of the results, a regularization parameter of magnitude $\lambda = 3$ has been used, since it proved to produce the most satisfactory results.

8.2 Vertical Resolution

In section 5.3 the maximum reconstruction resolution that can be expected was explained by means of the rank of the Hessian matrix (section 5.3). In figure 8.5 it is shown empirically that a higher resolution than 16 vertical levels does not provide good reconstruction results. The effect of the increased resolution is evident from the graph and also from the irregular colour pattern that can be observed in the image of the results in figure 8.5. When reconstructing for a higher number of unknowns than 16, the Hessian matrix is not of full rank, therefore an inverse can not be found (eq. 5.7) and it follows that an update $\triangle\sigma_{k+1}$ can not be solved for. Even when the data is regularised and the Hessian matrix is of full rank, the results are irregular.

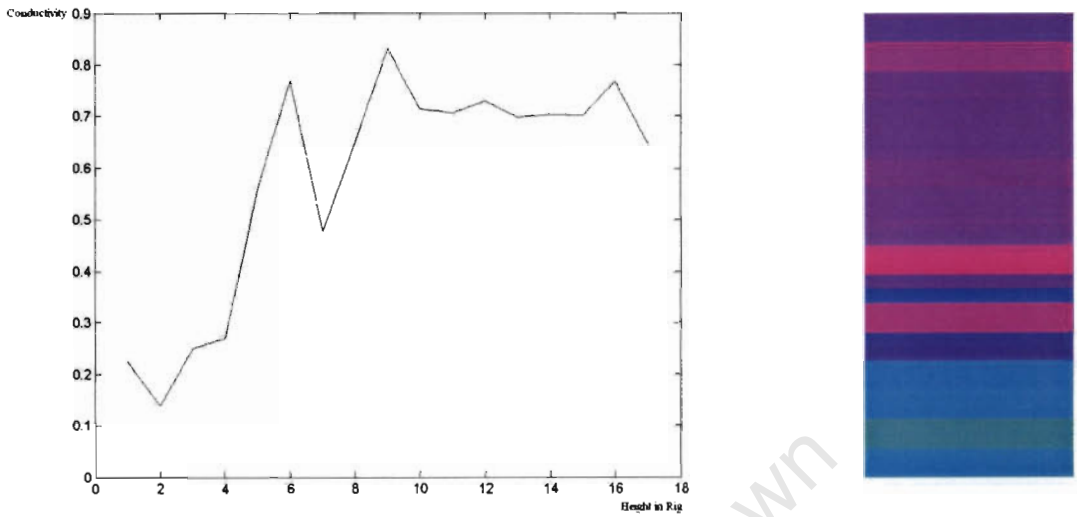

Figure 8.5: Example of results when using a higher resolution than allowed. An irregularity can be observed in the graph on the left for the "pixels" at higher resolution. An image of the same data is shown on the right. This observation supports the predication of a maximum of 16 levels.

8.3 Calibrated vs. Uncalibrated Image Reconstruction

When performing image reconstruction on uncalibrated measurements and comparing the results to reconstruction from calibrated measurements, the difference is surprisingly little. Figure 8.6 shows the results in graph form, displayed on the same set of axes. The lower curve represents the results for uncalibrated data, while the upper curve is based on the same data, but calibrated.

This can be attributed to the nature of the measurement strategy. The same argument used for motivating the higher than usual regularisation parameter applies here as well. As the system is overdetermined by nature of the measurement strategy, small changes to the data do not affect reconstruction significantly.

8.4 Number of Iterations for the Newton-Raphson Algorithm

The reconstruction results when performing several iterations have been compared to the results for the same data after the first iteration. Figure 8.7 shows both results in graph form, displayed on the same set of axes. The upper curve shows the results after ten iterations, while the lower curve shows the results after the first

iteration. Reconstruction has been performed using a regularisation parameter of $\lambda = 3$.

Notice that the result for several iterations shows more clearly defined conductivity boundaries, while the curve for results after the first iteration is still very smooth. For the results after ten iterations, the conductivity in the bottom and top regions has become more distorted. An explanation for this was given in subsection 8.1.1. However, the actual improvement from one iteration to ten iterations does not seem worth the additional computational expense. The results after one iteration display clearly the nature of the data, and the liquid-solid interface, which is what we are interested in, is also evident. Therefore the results for the settling processes in section 8.5 have been reconstructed using one iteration only.

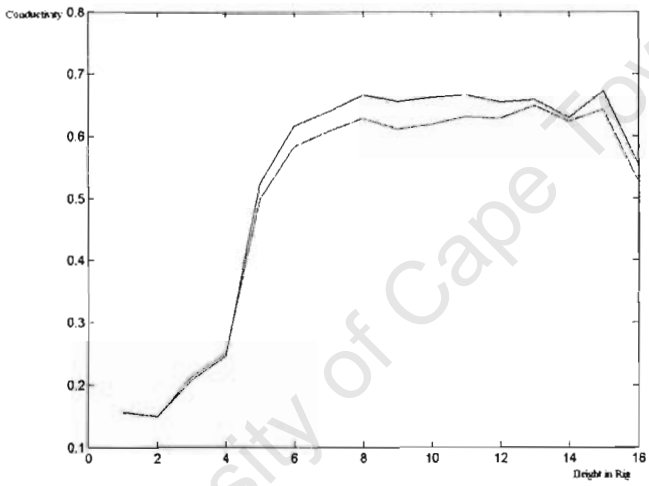

Figure 8.6: A comparison of reconstruction results for calibrated and uncalibrated data. The reconstruction from uncalibrated data is represented by the lower curve, while the upper curve represents the calibrated case. No significant difference between the two results can be detected.

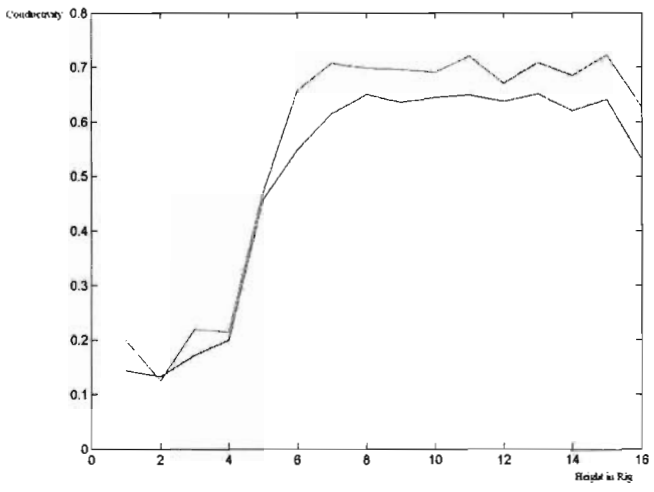

Figure 8.7: A comparison of reconstruction results after ten iterations vs. results after one iteration. The upper curve shows the reconstruction results after ten iterations, while the lower curve shows the conductivity results after the first iteration.

8.5 Reconstructing a Settling Process

A couple of measurements were recorded for a settling process occuring over a longer period of time, in order to establish what information can be extracted by the reconstruction software afterwards.

Figure 8.8 shows the results and corresponding drawings of the actual state of settling occuring over two hours. The sludge consists of three types of mud: (1) the main component which is the high density mud. It settles almost instantly and is almost impossible to keep suspended while stirring; (2) a finer sand component that settles fully after about an hour; (3) a fine "dust"-like component that can be regarded negligible for measurements. Type 3 only settles after about 24 hours after which the water becomes clear.

Mud type 1 is clearly represented in the reconstructed images by the light blue colour level. This corresponds to the black-coloured levels in the drawings below. The level does not change over the period of settling.

The light and darker brown coloured levels in the drawings represent mud type 2. The question mark (?) in the bottom drawing in figure 8.8 indicates that no clear mud-water interface could be observed in the rig at that early stage of settling. From the reconstructed images it can be observed how the conductivity drops in the region just above mud type 1, as mud type 2 accumulates in that area. However, the boundary between the mud levels and the water is not as clearly defined in the reconstructed images as was actually observed at the time of data capture. This

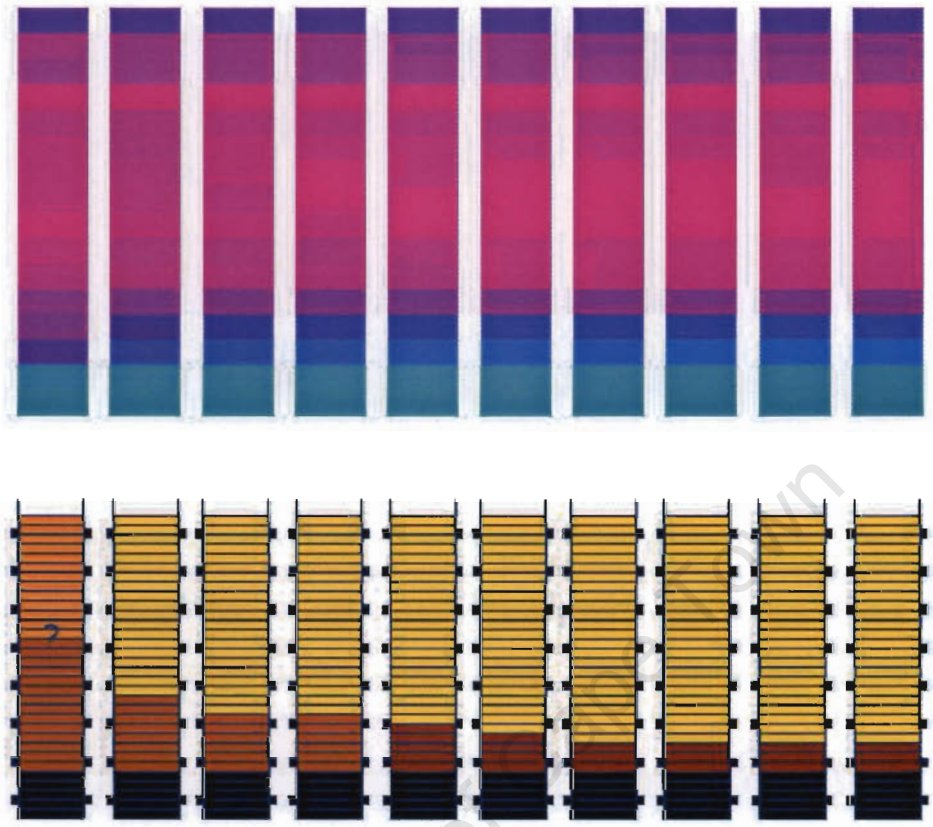

Figure 8.8: Results for settling occuring over two hours. The drawings below each image reflect the actual state of settling as observed and measured at the various stages of data capture. The main component of the sludge, the high density mud (black/light blue), has already settled when measurment started and could not be kept suspended by stirring.

is partly because of regularisation, which enforces a smoothness constraint on the results, preventing sudden changes in conductivity with changing height. Decreasing the regularisation parameter would result in more defined boundaries, but then the results would become irregular, as discussed in subsection 8.1.1.

Another set of measurements was taken with a considerable larger proportion of slurry compared to water in the rig. This time measurements were started during stirring. The results are shown in figure 8.9. Mud type 1 remains accumulated at the bottom of the rig even while stirring, as can be observed by the levels coloured in light blue. Due to the larger proportion of mud type 2, it can now be observed more clearly over the set of images how this mud accumulates above the mud type 1. This is shown by the darker blue-cloloured levels.

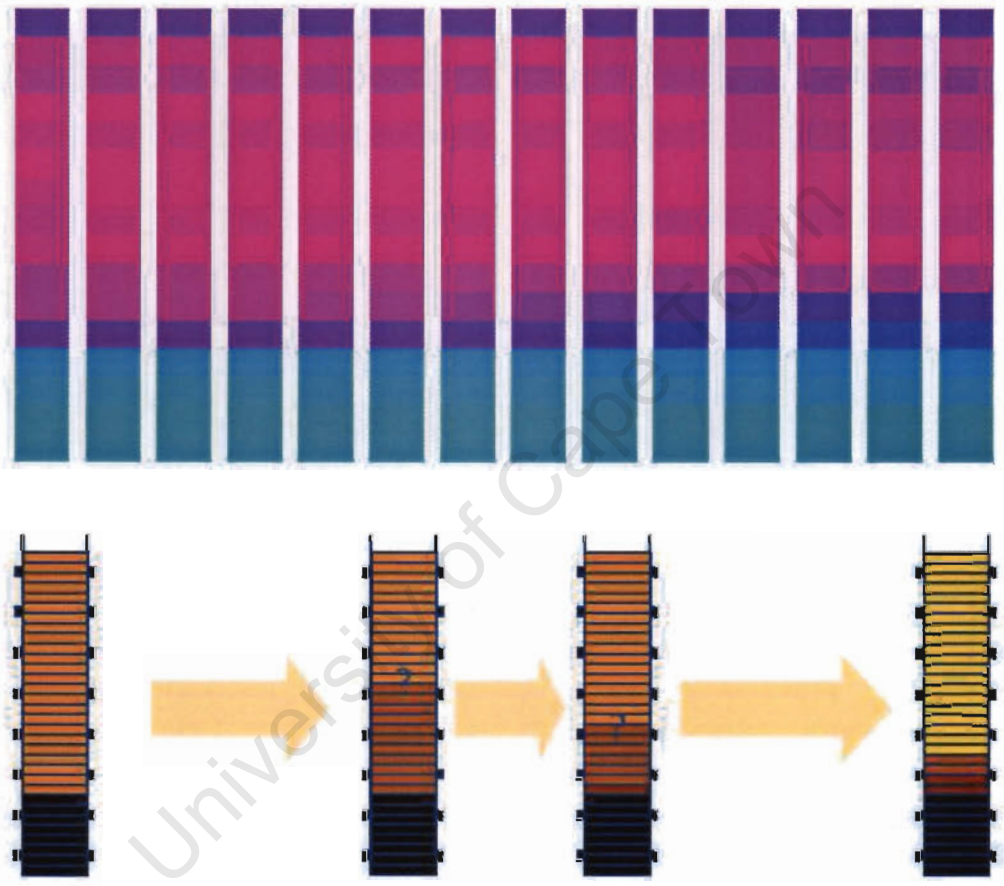

Figure 8.9: Results for settling over two hours. The rig contains a significant larger proportion of slurry than in figure 8.8. Measurements were started during stirring. The diagrams below show the progress of the settling as observed and measured during data capture.

Chapter 9

Conclusions

The following conclusions have been drawn:

1. Successful image reconstruction has been performed from tomographic measurements of a settling process.

Measurements from a suitable sensor were used to reconstruct the conductivity in the interior of a rig.

2. The vertical resolution for this particular configuration of electrodes and chosen measurement strategy is limited to 16 vertical levels.

This finding has been supported by making use of the rank of the Hessian matrix, and it has been shown empirically that a higher resolution does not produce satisfactory results when using the configuration as laid out in this thesis.

3. The liquid-solid interface of the settling slurry is clearly visible from the reconstructed images.

The liquid-solid interface level can be deducted unambiguously from the reconstructed data, and is only limited by the reconstruction resolution.

Chapter 10

Recommendations

The following recommendations are made for future work to be carried out in the context of this thesis:

1. Investigate the quality of reconstruction for different measurement strategies.

In particular, it should be investigated whether there is an improvement in reconstruction quality and accuracy if a measurement strategy based on a current driven source is used, instead of a voltage source. The measurement data used for reconstruction has been captured from a sensor system that uses a voltage source.

2. Study the implications of applying the method used in this thesis to a full-scale rig operating in a true mining environment.

Since the work for this thesis has been carried out exclusively in a laboratory environment, the same results can not be expected for a more practical environment. A study of the implications of applying this method to such an environment should be part of any further work.

3. Optimise the reconstruction code to produce an online and real-time system.

Further work should be carried out if this method were to be used in a real-time system. The data feed from the measurement circuit should be optimised, as well as revising the various parts of the reconstruction algorithm in order to increase code execution speed.

Bibliography

[1] G. Teague, "Using EIT for measuring a settling process," 2001, preliminary investigation.

[2] S. Herholdt, "Electrical impedance tomography sensor system for measuring of settling process," Undergrad. project, University of Cape Town, 2002.

[3] J. Webster, Ed., *Electrical Impedance Tomography.* Adam Hilger, 1990.

[4] P. Hua, J. G. Webster, and W. J. Tompkins, "Effect of the measurement method on noise handling and image quality of EIT imaging," *Proc. Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society*, vol. 9, pp. 1429–30, 1987.

[5] D. G. Gisser, D. Isaacson, and J. C. Newell, "Current topics in impedance imaging," *Clin. Phys. Physiol. Meas.*, vol. 8 Suppl. A, pp. 39–46, 1987.

[6] J. J. Cilliers, W. Xie, S. J. Neethling, E. W. Randall, and A. J. Wilkinson, "Electrical resistance tomography using a bi-directional current pulse technique," *Meas. Sci. Technol.*, vol. 12, pp. 997–1001, 2001.

[7] Z. Szczepanik and Z. Rucki, "Frequency analysis of electrical impedance tomography system," *IEEE Trans. Instrum. Meas.*, vol. 49, no. 4, pp. 844–851, 2000.

[8] F. Dickin and M. Wang, "Electrical resistance tomography for process applications," *Meas. Sci. Technol.*, vol. 7, pp. 247–260, 1996.

[9] F. J. Lidgey and Q. C. Zhu, "Electrode current determination from programmable voltage sources," *Clin. Phys. Physiol. Meas.*, vol. 13, pp. 43–46, 1992.

[10] W. Sansen, B. Geeraerts, W. V. Petegem, and M. Steyaert, "Electrical impedance tomography systems based on voltage drive," *Clin. Phys. Physiol. Meas.*, vol. 13, pp. 39–42, 1992.

[11] N. K. Soni, H. Dehgani, A. Hartov, and K. D. Paulsen, "A novel data calibration scheme for electrical impedance tomography," *Physiol. Meas.*, vol. 24, pp. 421–435, 2003.

[12] T. Naidoo, "Signal and image processing for electrical resistance tomography," Master's thesis, University of Cape Town, 2002.

[13] J. Li, "A method of reducing the error caused by boundary shape and electrode positions in electrical impedance tomography," *Physiol. Meas.*, vol. 15, pp. A169–A174, 1994.

[14] P. P. Silvester and R. L. Ferrari, *Finite Elements for Electrical Engineers*, 3rd ed. Cambridge University Press, 1996.

[15] R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt, *Concepts and Applications of Finite Element Analysis*, 4th ed. John Wiley and Sons, Inc., 2002.

[16] D. C. Barber, B. H. Brown, and I. L. Freeston, "Imaging spatial distributions of resistivity using applied potential tomography," *Elec. Lett.*, vol. 19, pp. 933–5, 1983.

[17] Y. Kim, J. G. Webster, and W. J. Tompkins, "Electrical impedance imaging of the thorax," *J. Microwave Power*, vol. 18, pp. 245–57, 1983.

[18] T. J. Yorkey, J. Webster, and W. J. Tompkins, "An optimal impedance tomographic reconstruction algorithm," *Proc. Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society*, vol. 8, pp. 339–42, 1986.

[19] T. J. Yorkey, J. G. Webster, and W. J. Tompkins, "Comparing reconstruction algorithms for electrical impedance tomography," *IEEE Trans. Biomed. Eng.*, vol. BME-34, pp. 843–852, 1987.

[20] P. Hua, E. J. Woo, J. G. Webster, and W. J. Tompkins, "Iterative reconstruction methods using regularization and optimal current patterns in electrical impedance tomography," *IEEE Trans. Med. Imaging*, vol. 10, no. 4, pp. 621–628, 1991.

[21] E. J. Woo, P. Hua, J. G. Webster, and W. J. Tompkins, "A robust image reconstruction algorithm and its parallel implementation in electrical impedance tomography," *IEEE Trans. Med. Imaging*, vol. 12, no. 2, pp. 137–146, 1993.

[22] M. Vauhkonen, D. Vadász, P. A. Karjalainen, E. Somersalo, and J. P. Kaipio, "Tikhonov regularization and prior information in electrical impedance tomography," *IEEE Trans. Med. Imaging*, vol. 17, no. 2, pp. 285–293, 1998.

[23] G. Tattersfield, *Electromagnetic Field Theory*, University of Cape Town, 1999.

[24] V. Balden, *Introduction to Finite Elements Course Notes*, University of Cape Town, CERECAM, 2003.

[25] K. H. Cho, S. Kim, and Y. J. Lee, "Impedance imaging of two-phase flow field with mesh grouping method," *Nuclear Engineering and Design*, pp. 57–67, 2001.

[26] J. R. Greene, *Neural, Fuzzy and Evolving Systems*, University of Cape Town, Dept. of Electrical Engineering, 2003.

[27] M. Cheney, D. Isaacson, J. Newell, S. Simske, and J. Goble, "NOSER: An algorithm for solving the inverse conductivity problem," *Int. J. Imaging Syst. Tech.*, vol. 2, pp. 66–75, 1990.

[28] "Finite element method magnetics," http://femm.foster-miller.net/index.html, last visited: 15 October 2004.

[29] "Electrical impedance tomography and diffuse optical tomography reconstruction software," http://eidors3d.sourceforge.net/, last visited: 15 October 2004.

[30] H. Muhlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm," *Evol. Comput.*, vol. 1, pp. 25–49, 1993.

[31] J. R. Greene, "Adaptive mutation breeder genetic algorithm (AMBA)," Personal Communication, 2004.

Appendix A

Software Code

```
function [a_soln,F_soln] = fem(varargin)

global N gauss_order no_elm no_nodes p t

[sig] = elmtprop;


%EVALUATE THE INPUT ARGUMENTS
if nargin==0    % if no input arguments (standard case) then continue without changing anything, i.e. break
                % the if statement
    ;
elseif nargin==1 % if 1 input argument, it contains the values for sigma, hence update sig.
    sig = varargin{:}; % This case is used for calculation of jacobian for the newton-raphson method.
else
    error('incorrect input arguments');  % if any other number of input arguments, throw an error.
end

if size(sig,2)~=no_elm
    no_elm
    error('Either sigma is undefined or the number of entries in sigma do not correspond to the number of
elements')
end

if size(p,1) < 4
    error('No p defined or you probably forgot to define the boundary conditions in the p matrix')
end

K = spalloc(no_elm*N,no_elm*N,no_elm*N);
%F is initialised below.
%a is initialised below.



% EVALUATE & ASSEMBLE GLOBAL ISOPARAMETRIC STIFFNESS MATRIX & FORCE MATRIX

% CALCULATE STIFFNESS MATRIX K

if N==4
    %DETERMINE K USING GAUSS INTEGRATION

    [w_nk w_nl eta_nl neta_nk] = gauss(gauss_order);

    for elm = 1:no_elm

        K_local = zeros(N,N);

        for i = 1:N
            for j = 1:N

                % GAUSS INTEGRATION
                for k = 1:gauss_order
                    for l = 1:gauss_order

                        [de_phi dn_phi] = d_phi(eta_nl(l),neta_nk(k));

                        jacobian_e_n = jacob(elm,p,t,eta_nl(l),neta_nk(k));
                        inv_jacobian = inv(jacobian_e_n);
                        [dphi_i] = inv_jacobian * [de_phi(i);dn_phi(i)];  %dphi/dx is in row 1 and dphi/dy
                                                                          %is in row 2
                        [dphi_j] = inv_jacobian * [de_phi(j);dn_phi(j)];

                        %Apply Gauss-Legendre Rules (Numerical Integration)
                        K_local(i,j) = K_local(i,j) + w_nk(k)*w_nl(l) * sig(elm) * (dphi_i(1)*dphi_j(1) +
dphi_i(2)*dphi_j(2))*det(jacobian_e_n);

                    end
                end

            end
        end
        % ASSEMBLE GLOBAL STIFFNESS MATRIX
        place = ((elm-1)*N)+1:elm*N;
        K(place,place) = K_local;

    end
```

```
%/continue function "fem"

elseif N==3
    %DETERMINE K FOR TRIANGLE ELEMENTS

    for elm=1:no_elm
        K_local = zeros(N,N);
        % DETERMINE COORDINATES OF LOCAL NODES

        for localnode = 1:N
            x(localnode)=p(1,t(localnode,elm));
            y(localnode)=p(2,t(localnode,elm));
        end

        % CALCULATE THE ELEMENT MATRIX K_local

        area = abs((x(2)*y(3)-x(3)*y(2)+x(3)*y(1)-x(1)*y(3)+x(1)*y(2)-x(2)*y(1))/2);

        b1 = y(2)-y(3);
        b2 = y(3)-y(1);
        b3 = y(1)-y(2);

        c1 = x(3)-x(2);
        c2 = x(1)-x(3);
        c3 = x(2)-x(1);

        K_local(:,:) = (sig(elm) / (4*area)) .* [b1^2+c1^2   , b2*b1+c2*c1 , b3*b1+c3*c1
            b2*b1+c2*c1 , b2^2+c2^2   , b3*b2+c3*c2
            b3*b1+c3*c1 , b3*b2+c3*c2 , b3^2+c3^2];


        %ASSEMBLE GLOBAL MASTER MATRIX
        place = ((elm-1)*N)+1:elm*N;
        K(place,place) = K_local;
    end


end



%DEFINE MAPPING MATRIX C - IT MAPPES THE LOCAL NODES TO THE GLOBAL NODES
C = spalloc(no_elm*N,no_nodes,no_elm*N);
i=0;
for elnum=1:no_elm
    for node=1:N
        i = i+1;
        C(i,t(node,elnum))=1;
    end
end


%ASSEMBLE MODIFIED STIFFNESS MATRIX
Kmod = C.'*K*C;

%APPLY BOUNDARY CONDITIONS

%CREATE THE INDEX MATRICES TO BE USED FOR RE-ARRANGING
[row_idx col_idx] = find(p==1);  %find the indices of nodes where voltages are applied, i.e. set to "1" (see
                                 %explanation of p matrix in the file "elmtprop.m")
pre_idx_all = col_idx(find(row_idx==3)); %filter out the 1's that are in the 3rd row of the p matrix.
pre_idx_grnd = pre_idx_all(find(p(4,pre_idx_all)==0)); %now filter out all the indices of the pre_idx_all
                                                       %above whose nodes are zero, i.e. are grounded.

pre_idx_act = pre_idx_all(find(p(4,pre_idx_all)));  %here filter out all indices of pre_idx_all whose nodes
                                                    %are non_zero, i.e. have active voltages applied.

proj_idx = pre_idx_all(find(p(5,pre_idx_all))); %find the columns that contain a projection number in the
                                                %fifth row

no_proj = max(p(5,proj_idx));  %number of projections


%SOLVE THE FEM PROBLEM FOR EACH PROJECTION (EG. FOR EACH APPLIED VOLTAGE)
 for proj_num = 1: no_proj
```

```
    %/continue function "fem"

    currentidx_of_preidxact = find(p(5,pre_idx_act)==proj_num) ; %find the nodes with active voltages for
                                                                %each projection
    pre_idx = [pre_idx_act(currentidx_of_preidxact); pre_idx_grnd]; %concatenate the index of the node where
                                                    %the active voltage is applied and the grounded nodes
    num_pre = size(pre_idx,1);  %this variable contains the number of predetermined nodes, i.e. the number
                                %of nodes with boundary conditions.

    idx = [1:no_nodes]';    % create a basic index matrix
    idx(pre_idx) = 0;       % set all indices that are in pre_idx to zero.
    idx = idx(find(idx));   % remove all zero entries from the basic index matrix

    %INITIALISE FORCE VECTOR (CURRENT VECTOR)
    F = spalloc(no_nodes,1,num_pre);


    %INITIALISE "a" VECTOR (VOLTAGE VECTOR
    a = spalloc(no_nodes,1,num_pre);  %initialise a, the voltage vector
    a(pre_idx_act(currentidx_of_preidxact)) = p(4,pre_idx_act(currentidx_of_preidxact)); %write the boundary
                    %conditions into the a vector (from p matrix that describes the geometry and mesh)


    %RE_ARRANGE THE MATRICES APPROPRIATELY
    Kff = Kmod(idx,idx);
    Kfp = Kmod(idx,pre_idx);
    Kpf = Kmod(pre_idx,idx);
    Kpp = Kmod(pre_idx,pre_idx);

    Ff = F(idx);
    Fp = F(pre_idx);


    af = a(idx);
    ap = a(pre_idx);

    %SOLVE THE MATRIX EQUATIONS
    af = Kff\(Ff-Kfp*ap);
    Fp = Kpf*af + Kpp*ap;

    %REBUILD THE SOLUTION MATRICES
    a_soln([idx;pre_idx],proj_num) = [af;ap];
    F_soln([idx;pre_idx],proj_num) = [Ff;Fp];

end
```

```
function [sigma] = elmtprop

%
%   THIS FUNCTION IS WHERE DATA SPECIFIC TO EACH PROBLEM IS ENTERED.
%   THE OTHER FUNCTIONS ARE NOT MEANT TO BE CHANGED, UNLESS THE TYPE OF PROBLEM COMPLETELY CHANGES -
%   - IF THAT HAPPENS,NEW CASES NEED TO BE ADDED TO THE VARIOUS FUNCTIONS
%

global N gauss_order no_elm no_nodes p t



% MATLAB uses the following convention for defining meshes:
%
% "The matrices P, E, and T are the mesh data.
%  In the point matrix P, the first and second rows contain
%  x- and y-coordinates of the points in the mesh.
%
%  In the edge matrix E, the first and second rows contain indices
%  of the starting and ending point, the third and fourth rows contain
%  the starting and ending parameter values, the fifth row contains
%  the boundary segment number, and the sixth and seventh row
%  contain the left- and right-hand side subdomain numbers.
%
%  In the triangle matrix T, the first three rows contain indices to
%  the corner points, given in counter clockwise order, and the last
%  row contains the subdomain number."
%
% --> For this computer program, the author has added the following
%     definitions to the matrices described above:
%
%     In the point matrix P, the third and fourth row decribe the boundary
%     conditions.  The third row contains the type of boundary condition,
%     "0" for no boundary condition, "1" for essential boundary conditions
%     (e.g. applied voltage) and "2" for natural flux boundary conditions
%     (e.g. current).  The fourth row contains the value of the boundary
%     condition.
%
%     In the triangle matrix T (or element matrix), the first N rows contain the
%     indices to the corner points (nodes), the second-last row contains the subdomain
%     number while the last row contains the value for alpha (or sigma, in the case of
%     conductivity.



%% CURRENT EXPERIMENT IS BASED ON DATA BELOW  (DATE:18July2004)
N=3;
load petnew

p = pnew;
t = tnew;
clear pnew tnew enew

t(5,:) = ones(1,size(t,2));

p(3:5,:) = zeros(3,size(p,2));

p(3:5,23)= [1;11;1];
p(3:5,67)= [1;11;2];
p(3:5,111) = [1;11;3];
p(3:5,155)= [1;11;4];
p(3:5,199) = [1;11;5];
p(3:5,243)= [1;11;6];
p(3:5,287) = [1;11;7];
p(3:5,331)= [1;11;8];
p(3:5,33) = [1;0;0];
p(3:5,77) = [1;0;0];
p(3:5,121) = [1;0;0];
p(3:5,165) = [1;0;0];
p(3:5,209) = [1;0;0];
p(3:5,253) = [1;0;0];
p(3:5,297) = [1;0;0];
p(3:5,341) = [1;0;0];
```

```
%/continue function "elmtprop"

%
%
%            LEAVE THE CODE BELOW UNCOMMENTED
%
%
%

%DETERMINE NUMBER OF NODES, NUMBER OF UNKNOWNS ETC.
no_elm = size(t,2);
no_nodes = size(p,2);
if size(t,1) == N+2
    sigma = t(N+2,:);
end
```

```
function [de_shape,dn_shape] = d_phi(eta,neta)

%THIS FUNCTION IS USED IF FEM IS IMPLEMENTED WITH SQUARE ELEMENTS

global N

    de_shape = [0.25*(neta-1)
                0.25*(1-neta)
                0.25*(1+neta)
                0.25*(-1-neta)];

    dn_shape = [0.25*(eta-1)
                0.25*(-1-eta)
                0.25*(1+eta)
                0.25*(1-eta)];
```

```
function [w_nl,w_nk,eta_nl,neta_nk] = gauss(order)

%THIS FUNCTION IS USED IF FEM IS IMPLEMENTED WITH SQUARE ELEMENTS
% GAUSS-LEGENDRE INTEGRATION


%GAUSS ORDER = 2
if order == 2
    eta_nl = [-1/sqrt(3) 1/sqrt(3)];
    neta_nk= [1/sqrt(3) -1/sqrt(3)];

    w_nk = [1 1];
    w_nl = [1 1];
end
```

```
function [J] = jacob(k,p,t,eta,neta)

%THIS FUNCTION IS USED IF FEM IS IMPLEMENTED WITH SQUARE ELEMENTS
% CALCULATE JACOBIAN

global N

%FOR 2-DIMENSIONS


x = p(1,t(1:N,k));
y = p(2,t(1:N,k));

[de_phi dn_phi] = d_phi(eta,neta);

dxde = 0;
dyde = 0;
dxdn = 0;
dydn = 0;

for i=1:N
    dxde = dxde + x(i)*de_phi(i);
    dyde = dyde + y(i)*de_phi(i);
    dxdn = dxdn + x(i)*dn_phi(i);
    dydn = dydn + y(i)*dn_phi(i);
end

J = [dxde dyde
     dxdn dydn];
```

```
function [sigma_final] = newt_rap(varargin)

%Use this function as follows:  sigma_final = newt_rap(sigma_guess,current_measurements)


sigma_guess = elmtprop;
lambda = 3;

%EVALUATE THE INPUT ARGUMENTS
if nargin==0    % if no input arguments (standard case) then continue without changing anything, i.e. break
the if statement
    ;
elseif nargin==1 % if 1 input argument, it contains the values for sigma, hence update sigma_guess.
    if size(varargin{:},1)==1
        sigma_guess = varargin{:};
    elseif size(varargin{:},1)==8
        volts = varargin{:};
        I = hard2soft(volts);   % This function arranges the values into the appropriate form
    elseif size(varargin{:},1)>8
        I = varargin{:};
    end
elseif nargin==2
    sigma_guess = varargin{1};
    if size(varargin{2},1)==8
        volts = varargin{2};
        I = hard2soft(volts);   % This function arranges the values into the appropriate form
    elseif size(varargin{2},1)>8
        I = varargin{2};
    end
else
    error('incorrect input arguments');  % if any other number of input arguments, throw an error.
end

I_meas = reshape(I,size(I,1)*size(I,2),1);  % put all columns of I into one column.


%CALCULATE JACOBIAN AND SAVE IT - ONLY USE FOR ONE ITERATION
% jacobian = jacob_nr(sigma_guess);
% save jacobian_nr jacobian

for i=1:10
    %CALCULATE EXPECTED RESULTS FOR CURRENT SIGMA GUESS
    sigma_map = map_sigma(sigma_guess);
    [U I] = fem(sigma_map);
    I_calc = reshape(I,size(I,1)*size(I,2),1);  % put all columns of I into one column.

    jacobian = jacob_nr(sigma_guess);

    [sigma_guess d_sigma] = update_sigma(sigma_guess,I_calc,I_meas,jacobian,lambda);

    sig_error = (sum((I_meas-I_calc).^2))

    plot(sigma_guess);
    hold on

end
hold off
sigma_final = sigma_guess;
```

```
function [jacobian] = jacob_nr(varargin)

%CALCULATE JACOBIAN
%Use this function as follows: jacobian = jacob_nr(sigma_guess)

if nargin==0    % if no input arguments (standard case) then continue without changing anything, i.e. break
the if statement
    sigma_guess = elmtprop;
elseif nargin==1 % if 1 input argument, it contains the values for sigma, hence update sig.
    sigma_guess = varargin{:}; % This case is used for calculation of jacobian for the newton-raphson
method.
else
    error('incorrect input arguments for jacobian calculation (should be sigma)');  % if any other number of
                                                                                    %  input arguments, throw an error.
end


for elm = 1:length(sigma_guess)

    sigma_increm = sigma_guess;

    sigma_increm(elm) = sigma_guess(elm)+sigma_guess(elm)*0.00001;

    sigma_map = map_sigma(sigma_guess);
    [U,I] = fem(sigma_map);
    F1 = reshape(I,size(I,1)*size(I,2),1);

    sigma_map_incr = map_sigma(sigma_increm);
    [U,I] = fem(sigma_map_incr);
    F2 = reshape(I,size(I,1)*size(I,2),1);

    F = F2-F1;
    dFdsigma = F./(sigma_guess(elm)*0.00001);

    jacobian(:,elm) = dFdsigma;
end
```

```
function [sigma_new,d_sigma] = update_sigma(sigma_guess,I_calc,I_meas,J,lambda)
% THIS FUNCTION CALCULATES THE CONDUCTIVITY UPDATE
dia = diag(ones(size(J,2),1));
d_sigma = -(J'*J+(dia*lambda))\(J'*(I_calc-I_meas));
Hessian = full(J'*J);
sigma_new = sigma_guess + d_sigma';
```

```
function [bestT] = calibval(realmeas,idealmeas)

% ADAPTIVE MUTATION BREEDER-GENETIC ALGORITHM (AMBA)
% WRITTEN BY J. R. GREENE

nvars = 24;                                     % number of variables
pop = 500;                                      % population size (number of trial solutions)
maxgen = 100;                                   % limit on number of generations
countstop = 0;                                  % counter to determine when to stop
gen = 0;                                        % record number of generations

thr = round(pop*0.15);                          % 15% selection threshold
fitrec = [];

delta = 0.1;

T = rand(16,pop)*0.1+0.95*ones(16,pop);                      % initial trial solutions
T(17:24,:) = rand(8,pop)*-0.003;


while countstop < 50
    gen = gen+1;
    f = calibfit(T,realmeas,idealmeas);                      % return row vector with fitness



    flow = mean(f(2:100));                  % mean fitness with lower mutation
    fhigh = mean(f(101:200));               % mean fitness with higher mutation

    if flow > fhigh
        delta = 0.95*delta;
    else
        delta = 1.05*delta;
    end

    [f,i] = sort(f);                       % sort by fitness in ascending order
    [S] = [T(:,i)];                         % sorted trial solutions
    best = calibfit(S(:,1),realmeas,idealmeas);              % best fitness so far
    fitrec = [fitrec,best];                 % record it
    pool = S(:,1:thr);                      % survivors

    T(:,1) = S(:,1);                        % elitist insertion

    for i=2:200                             % construct rest of population
        R = randperm(thr);                  % pick two different parents at random
        rnd = rand;
        if rnd < 0.15                       % discrete recombination (recombination by taken random picks
                                             % from any two parents)
            mask = rand(nvars,1) > 0.5;     %returns a boolean vector with ones and zeros
            T(:,i) = mask.*pool(:,R(1)) + (~mask).*pool(:,R(2));
        else
            rr = -0.25 + 1.5*rand(nvars,1);  % volume recombination
            T(:,i) = rr.*pool(:,R(1)) + (1-rr).*pool(:,R(2));
        end



        % mutate lower/higher half of population at lower/higher rate
        r1 = 1 + floor(nvars*rand);          % random integer in range (1,nvars)
        if i < 101
            T(r1,i) = T(r1,i) + delta*(randn/1.1);
        else
            T(r1,i) = T(r1,i) + delta*(randn*1.1);
        end
    end



    if gen > 1 & abs(fitrec(gen)-fitrec(gen-1)) < 0.001
        countstop = countstop + 1;
    else
        countstop = 0;
    end

    disp(['generation     ','mutation rate     ','current best    '])
    disp([gen,delta,best])
    plot(fitrec)
    drawnow
end

bestT = [T(:,1)];
```

```
function [f] = calibfit(T,real,ideal)

%THIS FUNCTION CALCULATES THE MEAN-SQUARED ERROR FOR THE CALIBVAL FUNCTION

% Put T into respective vectors
V=T(1:8,:);
G=T(9:16,:);
d=T(17:24,:);


for k=1:size(T,2)

    Vrep = repmat(V(:,k),1,size(real,2));       %Replicate columns and rows respectively
    Grep = repmat(G(:,k)',size(real,1),1);
    drep = repmat(d(:,k)',size(real,1),1);

    idealtrial = real.*Vrep.*Grep+drep;

    f(k) = sum(sum((ideal-idealtrial).^2));     %Calculate mean-squared error
end
```

```
function [creal] = adjust(T,real)

%THIS FUNCTION APPLIES THE CALIBRATION FACTORS

V=T(1:8);
G=T(9:16);
d=T(17:24);

Vrep = repmat(V(:,1),1,size(real,2));
Grep = repmat(G(:,1)',size(real,1),1);
drep = repmat(d',size(real,1),1);

creal = real.*Vrep.*Grep+drep;
```

```
function[Tavg] = calibrate(cal)

% THIS FUNCTION PERFORMS THE CALIBRATION PROCEDURE, CALLING ALL NECESSARY FUNCTIONS

n=10;

% CALIBRATE
calnorm = cal./max(max(cal));  %Normalised between 0 and 1

%Calculate Ideal values
[U I] = fem;
ideal = soft2hard(I);
idealnorm = ideal./max(max(ideal));

for i = 1:n
T(:,i) = calibval(calnorm,idealnorm);
end

Tavg = sum(T,2)./n;
```

```
function [ideal] = soft2hard(I)

% THIS IS AN AUXILIARY PARSING FUNCTION

[r c] = find(I<0);

for i = 1:length(r)
    Irm(i-(max(c)*(c(i)-1)),c(i)) = I(r(i),c(i));
end

Irm = fliplr(Irm);
Irm = flipud(Irm);
Irm = Irm';

ideal = -Irm;
```

```
function [volts_new] = hard2soft(volts)

% THIS IS AN AUXILIARY PARSING FUNCTION

global p t no_nodes

[row_idx col_idx] = find(p==1);  %find the indices of nodes where voltages are applied, i.e. set to
                                 % "1" (see explanation of p matrix in the file "elmtprop.m")
pre_idx_all = col_idx(find(row_idx==3)); %filter out the 1's that are in the 3rd row of the p
                                          %matrix.
pre_idx_grnd = pre_idx_all(find(p(4,pre_idx_all)==0)); %now filter out all the indices of the
                                  %pre_idx_all above whose nodes are zero, i.e. are grounded.
pre_idx_act = pre_idx_all(find(p(4,pre_idx_all)));  %here filter out all indices of pre_idx_all
                                    %whose nodes are non_zero, i.e. have active voltages applied.

volts_new = spalloc(no_nodes,size(pre_idx_act,1),size(pre_idx_grnd,1)+1*size(pre_idx_act,1));

volts = volts';
volts = fliplr(volts);
volts = flipud(volts);

inject = sum(volts,1);

for j=1:size(volts,2)
    for i=1:size(volts,1)
        volts_new(pre_idx_grnd(i),j) = -volts(i,j);
    end

    volts_new(pre_idx_act(j),j) = inject(j);
end
```

```
function [sig_result] = getresult

% THIS FUNCTION LOADS THE DATA AND CALLS THE NECESSARY FUNCTIONS FOR
% IMAGE RECONSTRUCTION.  THIS FUNCTION MAY BE MODIFIED.

load cal_coeff02 Tav

load meas16


sig01 = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
%this step for results3/7 (meas16)
step = [18,30,50,70,90,110,150,190,300,500,800,1000,1500,2000];

for i = 1:length(step)
    meas(:,:,i) = volts(:,:,step(i))/max(max(volts(:,:,step(i))));  %Normalise
    meas(:,:,i) = adjust(Tav,meas(:,:,i));
    sig_result(i,:) = newt_rap(sig01,meas(:,:,i));
    hold on
    plot(sig_result(i,:))
    hold on
    pause(2)
end
```