An Investigation Into the Dynamic Implementation of a 16-electrode FDM Electrical Impedance Tomography System

by

Gareth Goldswain

Submitted to the Department of Electrical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science

at the

UNIVERSITY OF CAPE TOWN

July 2005

Signature of Author .....

Department of Electrical Engineering
22 July, 2005

Statement of Originality

I know the meaning of plagiarism and declare that all work in this document, save for that which is properly acknowledged, is my own.

To my grandfather, Terrence Millen. Town electrical engineer, Tzaneen, 1957 - 1979.

Acknowledgements

The author wishes to thank the following:

- The project supervisor, Professor J. Tapson for the continued financial support, as well as his guidance throughout this thesis.
- The DeBeers mining group, specifically DebTech, for their financial backing.
- The Postgraduate Funding Office at the University of Cape Town, for their financial assistance and regular cash advances.
- Dr. A. Wilkinson's ERT research group, in particular Tim Long, for the data supplied and assistance with data acquisition.
- Jordan Boyle, for his persistant proof reading, editing and friendship.
- Bex, for her loving support.

An Investigation Into the Dynamic Implementation of a 16-electrode FDM Electrical Impedance Tomography System

by

Gareth Goldswain

Submitted to the Department of Electrical Engineering
on 22 July, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis documents one phase of an ongoing project - a project which was initiated in 1999, and whose research goal is to investigate, and provide insight into, the possibility of using impedance tomography to measure the mass flow rate of an air-gravel-seawater mixture. The ultimate goal of the research is to develop an instrument which can be placed in-line with an industrial scale airlift, and used to monitor its flow characteristics.

Since the project's conception, a number of researchers have been involved with various stages of its development; the research of G. Teague produced the most noteworthy results and since their publication (in his Ph.D. thesis), the drive of the project has been to pursue his recommendations.

Teague implemented a dual-plane, 16-electrode, Frequency Division Multiplexed (FDM) Electrical Impedance Tomography measurement system using square-wave excitation, as well as Neural Network (NN) based image reconstruction algorithms. Research succeeding Teague's investigated the use of sine-wave excitation on an single-plane, 8-electrode system, and using the reconstruction software developed previously, demonstrated superior outcomes.

This thesis is specifically involved with the upgrading of the abovementioned 8-electrode system, to a 16-electrode system capable of measuring *real* flows. Only static environments had been investigated prior to this research thus, the project's dynamic implementation issues needed to be addressed - a crucial step in the system's industrialisation.

Besides the design and fabrication of a new, open-ended, 16-electrode electrode ring, hardware issues which needed to be addressed included: the development of a new data acquisition system and the upgrading of measurement hardware. There exists (now) measurement hardware capable of capturing EIT data from a 16-electrode rig using sine-wave excitation.

The electrode ring contains three sets of electrodes - one plane of 'line' electrodes and two planes of guard electrodes (positioned on either side of the line electrodes). This variety of electrodes was realised to enable an investigation into the effects of electrode shape and configuration on NN based reconstruction algorithms. The reason for investigating the use of line electrodes is that previous research had shown significantly improved results in 2-dimensional Finite Element Method (FEM) simulations of periphery voltages in EIT.

Regarding the system's reconstruction software and signal processing: a new reconstruction method has been proposed and is investigated. Kernel Ridge Regression (KRR) is an 'intelligent' generalisation technique which has been demonstrated to outperform classical NN type

approaches in similar problems. The experiments of this research benchmark its performance against the best performing Multi-Layer Perceptron (MLP) NN found in Teague's research.

Static training data was attained in much the same way as in forerunning phases of the research. Three data sets were captured; measurements obtained using line electrodes, point electrodes (realised using one plane of guards) and line + guard electrodes were recorded, normalised and split into training and test data. On all datasets KRR outperformed MLP NNs in predicting volume fractions of a static 2-phase mixture. Data captured using line electrodes displayed improved results to data captured using point electrodes, when using MLPs to predict volume fractions. The results of KRR were unchanged. The use of guard electrodes in conjunction with line electrodes showed a further improvement in MLP and KRR generalisation ability.

Dynamic experiments were performed using two techniques: firstly, a laboratory scale airlift was used to emulate flow regimes similar to that expected in the final application. Results of testing MLPs and KRR, trained using the static data of previous experiments, on real data acquired from the airlift were poor. Thus, simulated boundary measurements with added Gaussian noise were used as training data, the results of which demonstrated a strong correlation between mean predicted volume fractions and average recorded flow rate.

The second dynamic experiment involved obtaining real flow data, captured from a high velocity pumping loop, together with reconstructed conductivity 'images' of each data frame. The data was supplied, and reconstructed by, Dr. A. Wilkinson's world-class research group at the University of Cape Town. The volume fractions of each frame were used as target values in database population and were estimated by finding what percentage of pixels in the image fell below a certain conductivity threshold. These target values, together with the raw voltage measurements used to reconstruct each frame's image, were used as training and testing instances. Testing KRR and MLPs on the abovementioned data produced good results. Specifically, KRR's mean absolute prediction error was less than 1% of the vessel's cross-sectional area.

The findings of the research lead to the following conclusions and recommendations:

1. KRR consistently outperforms MLPs in EIT reconstruction.
2. Data collected using line electrodes, especially used concurrently with guard electrodes, is superior when using KRR or MLPs to predict volume fractions of 2-phase mixtures.
3. Boundary measurements simulated using the Finite Element Method (with added Gaussian noise), produce a more representative training database than the empirical training methods used previously.
4. KRR, using simulated training data, performs well in predicting volume fractions of a rising-air-bubbles flow.
5. Kernel Ridge Regression, when trained using conventional images of a flow, can be used to predict volume fractions of similar flows directly.
6. Simulated measurements should become the method of choice for training database population for the prediction of real flows.
7. The real-time implementation issues of kernel methods should be investigated.
8. The system should be tested on high velocity flows.

Contents

Chapter 1

Introduction

This thesis documents one phase of an ongoing project - a project which was initiated in 1999, and whose research goal is to investigate, and provide insight into, the possibility of using impedance tomography to measure the mass flow rate of an air-gravel-seawater mixture. The ultimate goal of the research is to develop an instrument which can be placed in-line with an industrial scale airlift, and used to monitor its flow characteristics. The airlift in question simply uses the vacuum created by rising air, injected near the mouth of a pipe, to suck gravel and alluvial deposits off the seabed and onto the shore. Such deposits are rich in precious material, and until recently have remained an untapped resource - hidden by the ocean from the eyes of man. It should thus be obvious why the ability to accurately monitor such a process is invaluable, and hence why this project exists.

"The project"[1] has been through many stages of development since its conception and has been the subject of one undergraduate, two M.Sc. level and one Ph.D. theses [1, 2, 3, 4] before the present one. The most broad piece of documentation on this project is the Ph.D. thesis of Gavin Teague [4]. In fact Teague's thesis [4] should be considered as a pre-requisite to this thesis, as it covers almost all the background material relevant to topics covered in this text.

[1] "The project", throughout this thesis, refers to reasearch done, specifically towards the goal of monitoring a air-gravel-seawater mixture in an industrial airlift.

1.1 Tomography

"Tomography can be described as the measurement of some characteristic [of a medium] by examining it in cross-section" [5, 6]. The goal of any tomographic system is to, non-intrusively, gain certain information about the interior of some vessel (e.g. human body, pipeline, etc.). Generally speaking, how this information is obtained is by exposing the body being examined to some sort of induced energy field, and simultaneously measuring how this field is affected by the specimen. This is done commonly these days using x-rays in CAT (computer aided tomography) scans, and electrical currents in Electrical Impedance Tomography (EIT).

The information gathering methods of EIT use a number of electrodes - placed around the periphery of the body in question - to apply known currents or voltage fields to certain positions on the body's boundary. The resulting field, as determined by the internal characteristics of the body, is measured on the remaining electrodes. The impedance of the body is what acts on the injected currents or applied voltage field and hence, the kind of information gained about the body is related to its interior impedance distribution. The subject of how we use the collected data to 'reconstruct' the interior impedance distribution of the body being examined, is where most of the complexities of the field lie.

1.2 Impedance tomography at UCT

Two groups currently research electrical tomography at the University of Cape Town (UCT). The goals and project directions of the two groups are quite different, and hence the approach adopted by each group differs somewhat. The first of these groups (in no particular order), headed by Dr. A Wilkinson, concentrates on achieving accurate, real-time imaging - this seems to be the main focus in the electrical tomography community at present. The other research group, headed by Professor J. Tapson and of which the author is a member, is focused on obtaining underlying information about the area under investigation without first creating an image.

The above may seem unavailing, but it is important to understand the differences between the aims and methodologies of this project, and those of conventional EIT techniques - of which Dr. A Wilkinson's group is a world-class example. For this reason a brief summary of each

research group's direction and methods appears in table 1.1.

	Dr. A Wilkinson's Research Group	Prof. J. Tapson's Research Group
data used in reconstruction	resistance data	resistance and capacitance data
type of reconstruction	image	underlying data (e.g. volume fraction)
method of reconstruction	mathematics of inverse problems	neural networks
present focus of research	real-time imaging	real-time mass flow measurement

Table 1.1: A table highlighting the differences between the 2 impedance tomography research groups at UCT.

1.3 Background to the research

Since Teague's results were published in [5], the drive of the project has been to improve and extend certain aspects of his system, so that a functional tomography system may soon be realised in industry. As an example, Giannopoulos improved on Teague's results by using sine-wave excitation instead of the square-wave excitation used previously.

The state of development of the EIT system, when initially handed to the author, was the result of Giannopoulos' research [3] into improving Teague's data collection system. Hardware capable of capturing bimodal[2] data from an 8-electrode electrode ring, as well as the reconstruction software developed by Teague constituted the EIT system as received by the author. The intent was that the hardware should be extended to drive a 16-electrode system (as was the case with Teague's system).

Furthermore, owing to the fact that both Teague and the researchers who followed him had only investigated static (or near static) situations, the logical next step towards the project's ultimate goal would be an investigation into the dynamic implementation of the existing system. A research partnership with the Flow Process Research Centre at the Cape Peninsula University of Technology (CPUT), made available the use of their pumping laboratory. This provided the author's research group with the ability to test a dynamic EIT system, once developed, on high velocity flows similar to those expected in the final application airlift.

[2] Two modes of data i.e. resistance and capacitance data.

1.4 Aims and objectives

As indicated in the previous section, this thesis is involved with the dynamic implementation of a 16-electrode EIT system - a crucial step in the project's industrialisation. More specifically, the aims of the author's research are:

1. Undertake the design and construction of a 16-electrode electrode ring, capable of being fitted in-line of a high velocity pumping loop[3].

2. Upgrade the current hardware from an 8- to 16-electrode system. This involves the addition of four more signal sources, and four more 8-channel demodulators.

3. Identify and perform any additional hardware modifications necessary in implementing a dynamic rig.

4. Contrive and test a new method of training neural networks focused on volume fraction[4] prediction, as opposed to image prediction.

5. Devise some method of training neural networks on dynamic data as opposed to the static training data used previously. This involves some method of absolutely determining the flow characteristics (target values) inside the pipe.

6. If possible, create a link between the conventional image reconstruction methods used throughout the field and the neural computing methods used by the research group.

1.5 Scope and limitations

Although the long-term goal of the research is to develop an instrument capable of the real-time monitoring of industrial flow regimes, this thesis will focus on the instrumentation and signal processing side of the problem. The purpose of the research is to further establish whether

[3] Such a pumping loop is available for use at the Cape Peninsula University of Technology (CPUT). Basically, it consists of a large reservoir fitted with an industrial strength pump. The pump is capable of driving high velocity (up to $5m.s^{-1}$), multi-phase flows through a loop of 57mm PVC piping. The loop is returned to the reservoir to complete circulation.

[4] 'Volume fraction' is that percentage of the 2-dimensional cross-sectional area of the measurement space occupied by a certain phase of the flow.

such a system is feasible and investigate the issues involved with such a task. The research does not aim to achieve all of the functionality required by the final application. Hence, where elements or modules envisioned as part of the final instrument are known to be readily acquired or developed and are not the author's forte (e.g. optimised software development), their functionality is hypothetically assumed (e.g. un-optimal software may be developed) - but with careful consideration and justification.

The limitations of the research - besides the obvious limitations facing every researcher of time and money - are listed below:

1. For the first year of the author's research, the measurement hardware was being used by another member of the group for related research [7]. Thus, any modifications to the hardware could only be made after the completion of this work.

2. The development platform used in data processing was MATLAB - partly due to its superior matrix handling capabilities, and partly because of the author's familiarity with the language. MATLAB is a non-real-time platform, and any software developed in the project would most likely have to be optimised for on-line implementation.

3. The measurement strategy employed in previous projects [4, 3, 7] would dictate the measurement strategy used in this research. This is because the data-collection hardware received by the author is extremely parallel and thus complex in nature, and is the result of more than four years' research; the entire system is thus highly specific and accustomed to the measurement strategy realised by the hardware. It was the desire of the project sponsors that this hardware be upgraded, and research continued using this technique.

1.6 Plan of development

Chapter 2 begins with a brief overview of the basic principles of Electrical Impedance Tomography (EIT). More specifically, it presents issues relevant to the project's history and state of development. Chapter 3 deals with conventional EIT image reconstruction algorithms and investigates the factors affecting resolution using such techniques. Chapter 4 introduces the reader to the computational intelligence approach to EIT reconstruction - the viewpoint

adopted by the author's research group - and highlights the similarities and differences between conventional reconstruction methods and those used throughout the project's history. A similar investigation to that presented in Chapter 3 into the resolution determining factors of such systems is documented. Chapter 5 is involved with the design and implementation of a new electrode ring, as well as the upgrading of the measurement hardware to a 16-electrode system. Chapters 6 and 7 present the results of static and dynamic experiments performed using the new electrode ring and measurement hardware. Conclusions and recommendations for further development appear in the final chapters.

Chapter 2

Principles of EIT and history of the project

This chapter presents a brief overview of the rich field of EIT. It makes no attempt to explain the concepts put forward in-depth. Rather it seeks to familiarise the reader with the concepts relevant to this thesis, and provide a source of references. Where appropriate, justifications are made for the present state of the system. The history of the project is documented in terms of decisions made throughout its various stages.

2.1 A typical EIT system

Figure 2-1 shows the basic sub-systems which constitute a typical EIT system [8]. These systems will be referred to throughout the text.

Measurement Space

This is the area under investigation and can contain a mixture of anything from human tissue to rock and seawater. The literature is rich with methods of extracting data from the measurement space [4, 9, 10] - data which contains useful information about the characteristics of the medium filling the space. The most common of these methods are discussed further in section 2.2.1.

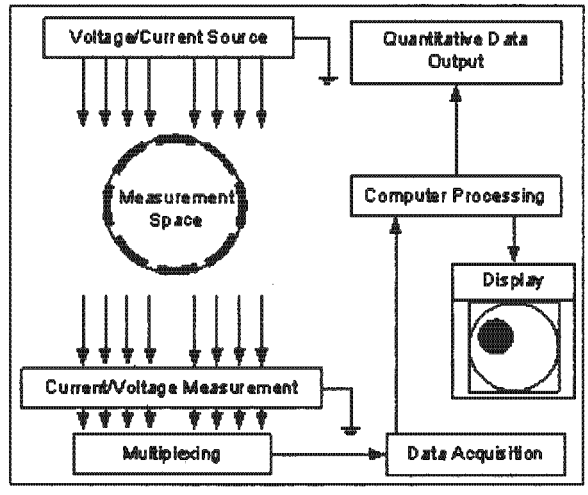


Figure 2-1: A block diagram illustrating the components of a typical EIT system. Blocks here represent subsystems and arrows signal paths. The terminology used here will apply throughout this document.

Voltage or Current Source

Most EIT systems rely on indirectly solving Maxwell's equations to produce some sort of conductivity map of the measurement space. The choice thus exists whether to inject current into the measurement space and measure the resultant voltage field, or to apply voltage and measure the resulting current. The various factors that need to be considered when making such a choice are discussed in section 2.3.

Current or Voltage measurement

If voltages are to be applied to the periphery of the measurement space, then currents must be measured at the periphery and vice versa. This is because, if we are interested in finding the resistivity distribution of the measurement space, then by Ohm's law we need to know the current and voltage distribution within the space.

Multiplexing

To obtain a useful resolution within the measurement space it is necessary to place measurement electrodes at closely spaced intervals around the periphery. In fact, it can be shown [4, 10] that

the number of independent measurements, which is an indication of the maximum obtainable resolution, is directly related to the number of peripheral electrodes (see section 2.2.2).

Thus, if we are to obtain a useful resolution, a large amount of data (over 100 readings for most 16-electrode systems) needs to be captured from the measurement space for each reconstructed 'frame'[1]. It is impractical to read this quantity of data in parallel, meaning that some form of multiplexing will be necessary in order to capture each frame of data.

Data Acquisition

Most modern systems make use of sophisticated data acquisition modules which need to be synchronised with multiplexers in order to capture the large amount of data necessary to reconstruct a single frame.

Computer Processing

Once an entire frame of data has been captured, a significant amount of data manipulation is necessary. Such tasks as data normalisation and calibration may be necessary before information can be extracted from the raw data. Once pre-processing is complete the resulting data can be fed to an imaging algorithm or process controller via an intelligent system.

Display/Quantitative Data Output

In most applications, the most useful and common output is an image displaying the spatial conductivity distribution of the region. This is then further analysed by a human specialist (e.g. a medical doctor in the case of imaging the human thorax) who can make an informed decision. However, especially with the emergence of machine vision and artificial learning algorithms, useful data may be obtained in software without ever producing an image [1].

Most often a resistance or conductance mapping of the vessel is most useful (e.g. air-water mixture). Nonetheless, in systems where different phases of the imaging medium possess similar resistive characteristics (e.g. air-gravel mixture) capacitance mapping may prove more useful [2].

---

[1] 'Frame' here, and throughout the text, can be best explained in terms of image reconstruction. As with a motion picture, each frame is a 'snapshot' in the history of the medium's movement. The data required to reconstruct such a frame is referred to as a 'frame of data.'

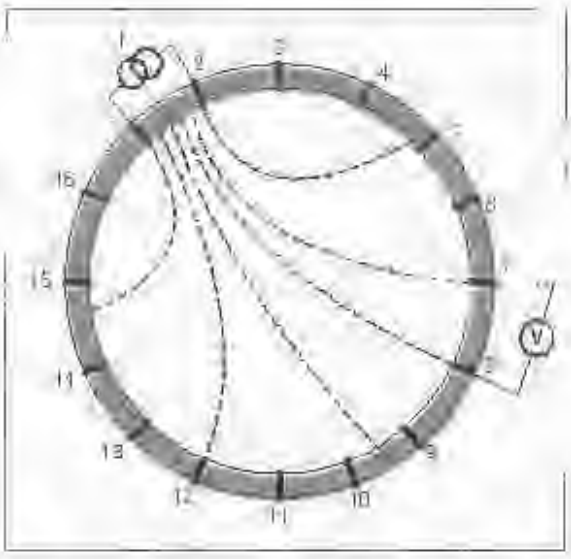


Figure 2-2: A diagram illustrating the concept of the Neighboring Method of data collection. Here, a known current source (I) is applied between an adjacent pair of electrodes (1 and 2), and the potential difference measured between all other adjacent pairs on the boundary (2-3, 3-4, ..., 16-1). The current source is then moved to the next adjacent pair (2 -3) and the voltages on the remaining electrodes measured. This process is repeated until each electrode pair has been excited. This constitutes 1 frame of data. Note that the dotted lines in the measurement space represent equipotentials.

2.2 FDM vs. TDM systems

2.2.1 Time Division Multiplexing (TDM)

Almost all the EIT systems the author has studied, and indeed most documented tomographic systems, have used Time Division Multiplexing (TDM) to inject/measure signals into/from the measurement space. Thus, only one injection signal is present in the medium at any one time. This signal is applied between a number of electrodes (usually two) on the periphery, and demodulated on the others. The signal is then applied between the next pair, and so on. One commonly used technique which illustrates the concept of TDM is the Neighboring Method depicted in figure 2-2.

A wide variety of TDM data collection strategies exist in the field and have been thoroughly tested and documented. Refer to the following sources for more information on their

implementation and characteristics [1, 9, 10, 11, 12].

2.2.2 Frequency Division Multiplexing (FDM)

One of the inherent drawbacks of TDM based data collection techniques arises from the fact that one frame of data requires the sequential switching of the signal source between the electrodes on the periphery. The finite amount of time between switching, small as it may be, implies an assumption that the medium is not moving. The errors introduced by this assumption will be insignificant in static situations, but will become more significant as the speed with which the medium is moving increases. Although fast TDM based systems have been developed [13], the nature of TDM means that it is fundamentally limited by the speed at which signals can be switched between electrodes.

For the reasons outlined above, Frequency Division Multiplexing (FDM) has been the data collection technique used by the author's research group since proposed by Teague and Tapson in 2002 [14]. It has proved to be an effective data collection method, the results of which are documented in [3, 4, 7].

FDM is not completely without disadvantages. As mentioned in section 2.1, the number of linearly independent measurements available, which is directly related to the system resolution, is lower than some TDM variants, e.g. the TDM "4-electrode Adjacent Pair" data collection method produces $\frac{N(N-1)}{2}$ independent measurements [15], while the FDM system developed by Giannopoulos in [3] and used in this project produces $\left(\frac{N}{2}\right)^2$ - where N represents the number of electrodes [4, 16].

This is an important fact to note when designing a measurement rig, as it affects the number of electrodes required. As can be seen from the formulae above, in order to obtain the same system resolution as TDM, FDM methods require more electrodes. It is for this reason that the author's stage of the project will use 16 electrodes, as opposed to the 8 used by Giannopoulos [3] in the FDM system preceding this research.

The concept of FDM is illustrated in figure 2-3. Note that each electrode is a dedicated receiver or transmitter as opposed to performing both roles in TDM.

Further insight into the FDM signal injection method is presented in figure 2-4, which shows an oscilloscope screen capture of the voltage measured by dipping an oscilloscope probe in the

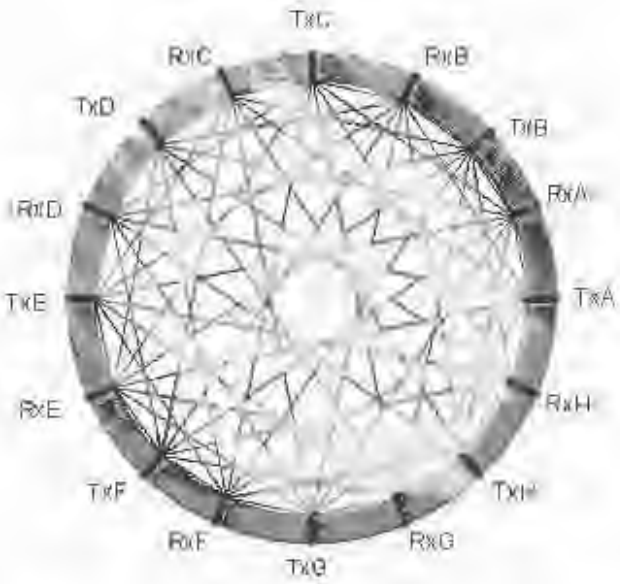


Figure 2-3: A diagram illustrating the concept of EDM. Note that the electrodes are either labeled as transmitters (Tx_) or receivers (Rx_). Each line drawn in the measurement space represents the shortest path between a particular transmitter and receiver; it does not illustrate what the current flow or voltage field looks like.

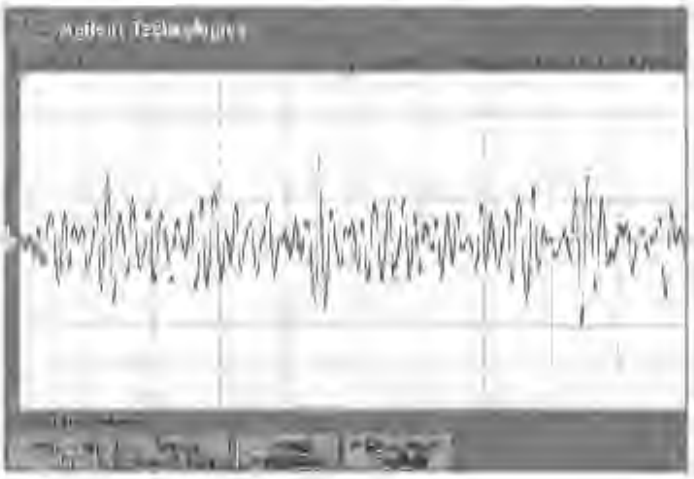


Figure 2-4: An oscilloscope screen capture of the voltage vs. time measured by an oscilloscope probe dipped in the centre of the active rig filled with tapwater. It is clear that a number of frequencies are present here.

centre of the project's electrode ring (filled with tapwater) while all transmitter electrodes are active. Figure 2-5 shows a Fast Fourier Transform[2] (FFT) of the voltage captured in the previous figure. It shows the clear separation of 8 transmitter frequencies in the frequency domain.

2.3 Voltage Driving vs. Current Injection

It was mentioned in section 2.1 that the choice of injecting currents and reading voltages or vice versa needs to be made when designing an EIT system. The single most important design parameter to consider when making such a choice is the output impedance of the driver and input impedance of the receiver. If a driver has a high output impedance (e.g. current source) then the contact impedance (i.e. impedance between electrode and electrolyte) is less significant than that of a low output impedance driver (e.g. voltage source). Likewise, a voltmeter has high input impedance - making the contact impedance of the receiver negligible, while an ammeter has low input impedance - making it more significant.

[2] As executed by the Agilent Technologies 54622D oscilloscope.

Figure 2-5: The FFT of the signal captured in figure 2-4, as performed by the AGILENT TECHNOLOGIES 54622D oscilloscope. 8 distinct frequency spikes can be seen here, illustrating how FDM achieves signal separation.

The other factor influencing the choice between current or voltage driving a system is the ease with which the drivers can be implemented. Current sources are much more difficult to realise than voltage sources. Also, when using FDM, complex demodulation circuitry is required if current sources are to be used[3].

For this reason, when this project was started by Teague [4], voltage drivers and current receivers were decided upon. Further justification for this choice originates from the types of flows expected in the final application's pipeline (air-gravel-seawater). This is relevant because "It has been shown that the contact impedance becomes negligible for the case where only one of the phases is conductive, provided a sufficiently high excitation frequency is used" [17].

2.4 Reconstruction algorithms

It has been put forward, and should be emphasised, that an image need not be the desired output of an EIT system. However in order to determine system performance, it is desirable to have the ability to create images. These images may in fact be useful in the process of

[3] A new demodulation technique has been proposed by Prof. J. Jansen. It is currently being investigated and may be implemented in the next phase of this project.

running computational intelligence to make decisions based on the state of the measurement space. Thus, a large proportion of this thesis focuses on image reconstruction, even though what is required at the end of the line is a *volume fraction*[4] prediction, not an image.

2.4.1 Conventional methods

By far the most popular and widely documented method of reconstructing images in ECT is by using the Modified Newton-Raphson (MNR) algorithm, or similar. The way in which this algorithm achieves image reconstruction is by iteratively guessing what the internal conductivity distribution of the measurement space looks like. This relies on discretising the domain into pixel-like elements. Once an initial guess has been made, numerical methods are used to predict what the boundary measurements *would be* - given this proposed conductivity map and the *known* injection signal magnitudes (boundary conditions). The squared difference between these predicted values and values measured from the rig, is the error function which the MNR algorithm seeks to minimise. These methods are discussed in more detail in Chapter 3.

2.4.2 Neural networks

The imaging methods described above are highly complex, and require a great deal of mathematical understanding of the algorithms and methods used. Also, these methods make a number of simplifying assumptions when performing image reconstruction - artificial neural networks, on the other hand, are trained by example and no such assumptions are made [1, 18]. Neural network techniques also have the potential to perform real-time imaging of fast changing processes, especially due to the fact that their weights and activation functions (which, together with their architecture are entirely responsible for their behavior) can be stored in some form of memory, thus allowing them to be implemented as embedded systems [1, 19, 20, 21].

For these reasons, the author's research group initiated a study into the capacity of intelligent systems to perform image reconstruction on ECT data. This study was based on work done by Nooralahiyan *et al* [21], and is documented in Teague's Ph.D. thesis [4] as well as [3, 7]. In summary, what Teague managed to show is that neural networks (NNs) could be trained,

[4] Recall that 'volume fraction' is that percentage of the cross-sectional area of the measurement space occupied by a certain phase.

using ECT data captured from a physical rig, to successfully predict low resolution images of a multi-phase mixture.

No attempt is made in this text to explain the fundamental concepts of the neurocomputing techniques used by Teague. Rather, a basic understanding of neurocomputing methods is assumed. A more in-depth discussion of the neurocomputing methods adopted from Teague's work [1, 4] and applied to this project is presented in Chapter 4. For more introductory and background information, the reader is referred to [1, 4, 22, 23].

2.5 History of the project

This section documents the history of the research performed by the author's research group. It is presented in a logical chronological sequence.

The research of Q. Smit -2000 [2]

The main topic concerning Smit in his research was to ascertain whether capacitance data is useful in gaining information about the interior of a air-gravel-seawater bearing vessel. The use of Neural Networks for image reconstruction provided promising results.

The research of G. Teague - 2002 [4]

Teague implemented a 16-electrode, bimodal[1], FDM data acquisition system to extract data from an air-gravel-seawater filled vessel. Tests were performed mainly on purely static situations, but also on simulated bubble movement. This was done by pulling pockets of gravel and polystyrene through the rig using servo motors. Cross-correlation techniques enabled Teague to measure these simulated flows.

Teague realised novel image reconstruction software using neural networks. Using the training procedure outlined in later chapters, he showed that neural networks could be trained to predict low resolution images of the measurement space. Interestingly, and significant to the requirements of the final airlift application, *volume fractions could be predicted directly*[2].

[1] 'Bimodal' refers to the use of both resistive and capacitive data in reconstruction.

[2] This is significant because the final application of the research requires volume fraction predictions as opposed to images.

The research of A Giannopoulos - 2003 [3]

Where Teague used square-wave excitation, Giannopoulos used sinusoidal excitation - proving Teague's predictions correct i.e. performance would be improved using the more 'harmonically clean' sine-waves. He used the image reconstruction software developed by Teague [1, 4] to obtain results. Giannopoulos however, only implemented an 8-electrode bimodal data acquisition system [3]. One of the topics of the present phase of the project is to extend Giannopoulos' hardware to a 16-electrode system.

The research of S. Herholdt - 2004 [9]

Herholdt successfully implemented a measurement system and image reconstruction software used to measure a settling process. Unlike researchers in the group before him, he used conventional image reconstruction techniques. He provided valuable insight into such mathematical methods as the Finite Element Method - discussed further in section 3.3.

The research of V. Capindissa - 2005 [7]

Capindissa investigated using the hardware developed by Giannopoulos and the image reconstruction software developed by Teague to monitor an industrial hydrocyclone. Not surprisingly, similar results to that of Giannopoulos were obtained justifying the continuation of this branch of the research.

2.6 A note on the author's phase of the project

It is the aim of the author's research to modify the data collection hardware developed previously [3], to collect *dynamic* data and predict the volume fractions (percentage of the cross-sectional area of the measurement space occupied by a certain phase) of a multi-phase flow. Besides the design of an open ended[7] electrode ring, this involves re-evaluating the methods previously used by the group. Some pertinent issues need to be considered when crossing between a static and dynamic environment.

[7] 'Open-ended' here points to the fact that the ends of the pipe section which electrodes are to be fitted to (electrode ring) are physically open, allowing flow to pass through. Previous projects, because they were involved with static situations, used a blocked off section of pipe.

Almost all the research performed by the group prior to the author's stage of the project, has dealt with near static situations[8]. Specifically, the performance measure used (mean squared error) was acquired in a static scenario. This is due to the same reason that the research problem itself exists - i.e. there is no convenient, cost-effective, non-intrusive and accurate way of knowing exactly what real flow - moving at a significant velocity - looks like in cross-section.

It seems highly unlikely that neural networks trained using ideal bubble shapes and the visual verification procedure used previously[9], will generalise to real flows. This is because real flows are very much more complex and irregular than one could hope to emulate in each of the 'snap-shot' instances used by Teague, Giannopoulos and Papalissa in network training. It is a well known fact that the performance of neural networks is directly and strongly related to how representative their training data set is, of the data they are to interpolate from.

Another factor that needs to be considered when training networks in this way, is the immense time and effort required to populate a training database extensive enough to capture the underlying complexities of real-world turbulent flow. This costliness limits us to obtaining a small number of data points in a super dimensional space, which is a known pitfall of neural networks.

Even if we are to determine how a statically trained system performs on dynamic flows, some way of absolutely determining the flow characteristics, needs to be addressed. If we don't know this, then how can we evaluate system performance other than through visual estimation and guesswork?

[8] In fact, Teague *did* test his system on ideal bubble shapes being pulled through the pipe by servomotors, but this is far from the velocity and bubble distributions expected in the final application. These have previously considered to be static in this thesis.

[9] Briefly, the visual verification procedure used previously involves looking at the static measurement space from above to determine the distribution within the space. See section 4.2.1 for a more in-depth description.

Chapter 3

Conventional EIT Imaging Systems

3.1 Introduction

Once data has been collected from a vessel using the methods described Chapter 2, it is the goal of any EIT system to produce some sort of useful output which is representative of some aspect of the interior of the vessel. Almost all EIT systems aim to produce an image - this is because an image is what we as humans are comfortable dealing with. In the case of Electrical Resistance Tomography (ERT), the image would take the form of a spatial conductivity map of the vessel's interior. If capacitance data were extracted from the vessel, a dielectric mapping would be more appropriate.

It was stated in section 2.4 that by far the most popular method of obtaining images in EIT is using the Modified

Newton-Raphson (MNR) algorithm. Hua and Woo describe it as "one of the most theoretically sound algorithms" and state that "it is known to be the most sophisticated" of all the algorithms investigated [24]. For this reason it will be the focus of this chapter.

Note that a wide collection of literature contributed to the concepts and equations presented in what follows. In the following explanations, where a particular author (or collection of authors) has specifically been drawn from, it is cited in the normal fashion; elsewhere all of the following are cited: [9, 12, 23, 24, 25, 26, 27].

3.2 The Forward vs. Inverse Problem

When voltages or currents are applied to the boundary of a medium, the current flow or electric field in that region is described by Poisson's Equation:

$$\nabla \cdot \rho^{-1} \nabla V = f \tag{3.1}$$

with boundary conditions:

$$V = V_0 \text{ on } \partial\Omega \tag{3.2}$$

$$\rho^{-1} \partial V / \partial n = J_0 \text{ on } \partial\Omega \tag{3.3}$$

where ρ = *resistivity*, V = *voltage* and f = *impressed current source* distributions within the region of interest. ∇ is the Poisson operator. V_0 and J_0 are the voltage and current density respectively, at the boundary.

The Poisson operator (∇) used above produces the partial derivatives of a function in orthogonal dimensions. Thus, it is intuitive that equation 3.1 can be applied to the 2-dimensional (2-d) and 3-dimensional (3-d) case. For the purposes of this thesis, only 2-d situations are considered. The literature however, contains an abundance of 3-d applications[28, 12].

In EIT, there generally exist no significant current sources within the region of interest, hence $f = 0$ and equation 3.1 becomes: [24]

$$\nabla \cdot \rho^{-1} \nabla V = 0 \tag{3.4}$$

This equation is known as the *governing equation* of EIT and it gives rise to the 'forward' and 'inverse' problems. The 'forward problem' in words is: "Given ρ, find V and J (the internal voltages and current densities) and hence V_0 and J_0 on the boundary." The 'inverse problem' is: "Find ρ^{-1} given V_0 and J_0" [24]. "Generally speaking, the numerical solution of an inverse problem requires iterative solutions of a forward problem" [24].

3.3 Solving the Forward Problem

It turns out that in most cases Poisson's equation is impossible to solve on a continuous domain so numerical methods such as the Finite Difference Method (FDM) or the Finite Element Method (FEM) must be employed. For reasons given by Hua and Woo [24], the FEM is superior in the type of problems generally encountered in EIT. It will thus be the only forward problem solver discussed here.

3.3.1 The Finite Element Method (FEM)

- The FEM is a well documented method, commonly used for solving partial differential equations (PDEs). The derivation of the equations used to implement the method are complex and unimportant with regards to this thesis.
- In essence, what the FEM achieves is changing the calculus problem of equation 3.4 into a linear system of algebraic equations i.e. $\nabla \cdot \rho^{-1} \nabla V = 0$ is transformed into the more friendly $Y v = I$[1]. This is achieved by dividing the domain on which the solution exists into a finite number of elements.
- It can be shown that as the size of elements approaches zero, so too does the FEM solution approach the real solution [24].

3.3.2 Discretisation of the domain (meshing)

- The first step in the FEM is to discretise the domain. This is done by systematically defining a number of points, or nodes, in the region for which the field variable (voltage) of the governing equation 3.4 will be solved for exactly. The nodes should be numbered for indexing reasons, and are characterised by a set of cartesian coordinates.
- The next step in a meshing algorithm is to 'join the dots' in such a way as to create a collection of non-overlapping elements. As with the nodes, these elements must be numbered for indexing reasons. When considering the final image (or conductivity map), these

[1] $Yv = I$ is analogous to Ohm's law; $v \Longrightarrow$ *voltage*, $I \Longrightarrow$ *current*, $Y \Longrightarrow$ *admittance*

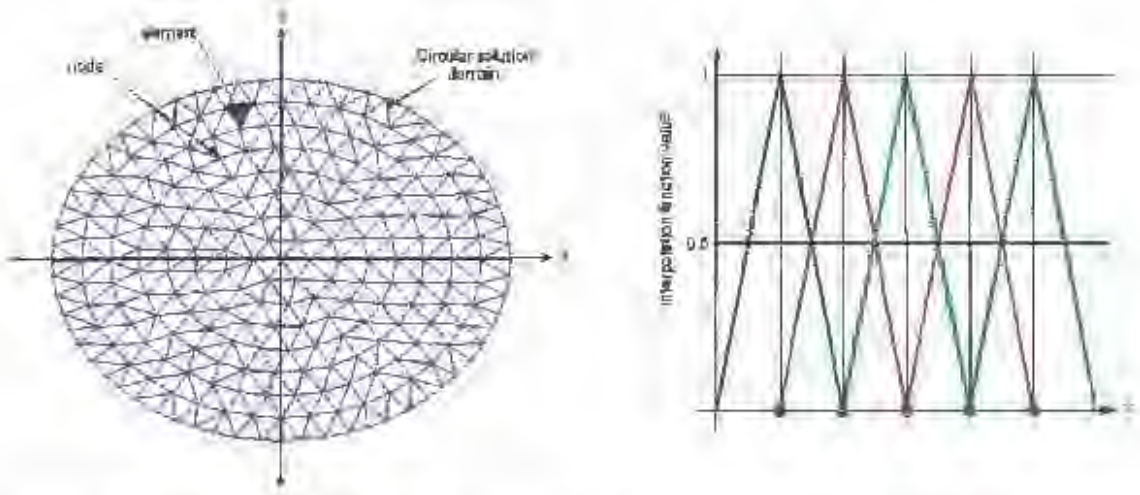


Figure 3-1: A diagram illustrating the concepts of a finite element mesh. On the left is a circular geometry (such as the cross-section of a pipe) which has been discretised. In the FEM, each element is assumed to have the same conductivity when solving the governing equation. The solution to the governing equation is solved with zero error at each node in the mesh; between nodes the solution is approximated by multiplying the known node values by interpolation functions. On the right, a simple 1-dimensional case is used to illustrate how the interpolation functions might achieve this - note that each node's interpolation function is unity at the node in question, and zero at all other nodes.

elements can be considered pixels. Obviously then, increasing the number of elements increases image resolution.

- Due to the discretisation of the problem, the exact value of the field variable (voltage) is unresolved between these nodes, and must be approximated using interpolation functions. These functions assume the resistivity of each element to be constant inside the element, and constitute the abovementioned linear system of algebraic equations.

- The choice of such functions is problem-specific; higher order functions will produce a better approximation for more complex cases. In most EIT systems simple linear interpolation functions are used - the advantage being that these are less computationally intensive.

A diagram illustrating the concepts of a finite element mesh is presented in figure 3-1.

3.3.3 Node numbering and matrix assembly

- If simple linear interpolation functions are used to approximate the voltage change in each element, they can be written as:

$$\phi_j(x, y) = p_j + a_j x + \omega_j y \tag{3.6}$$

where p_j, γ_j and ω_j are polynomial coefficients.

- The function $\phi_j(x, y)$ is unity at the jth node of an element and zero at the other. Using this property, and after considerable algebraic manipulation it can be shown that the 'element matrix' (sometimes called the 'stiffness matrix') of a triangular element is:

$$[k] = \frac{\sigma}{4\Delta}\begin{pmatrix} b_1^2 + c_1^2 & b_1 b_2 + c_1 c_2 & b_1 b_3 + c_1 c_3 \\ b_1 b_2 + c_1 c_2 & b_2^2 + c_2^2 & b_2 b_3 + c_2 c_3 \\ b_1 b_3 + c_1 c_3 & b_2 b_3 + c_2 c_3 & b_3^2 + c_3^2 \end{pmatrix} \tag{3.10}$$

where σ = *element conductivity*, Δ = *area of element*. The scalar values $b_1, b_2 \ldots c_3$ are related to the lengths of the different sides of each triangular element – see [24] for exact definitions.

- This element matrix describes the relation between current and voltage at each node of the element. If we are to obtain a solution over the entire region we must assemble the 'master matrix', Y. "The basis of the assembly procedure is that the values of the field variable are the same at the nodes where elements are interconnected" [24].

- Each element's nodes are numbered locally i.e. if an element has n nodes associated with it, then locally (to the element in question), these nodes will be numbered 1 *to* n. Local node numbering considers each element as being disjointed and independent. When it comes to assembling the master matrix Y, it is necessary to re-number nodes in a global sense. This is because different elements will share certain nodes. These concepts are illustrated in figure 3-2.

- When global node numbers are assigned, the meshing algorithm must take into account the

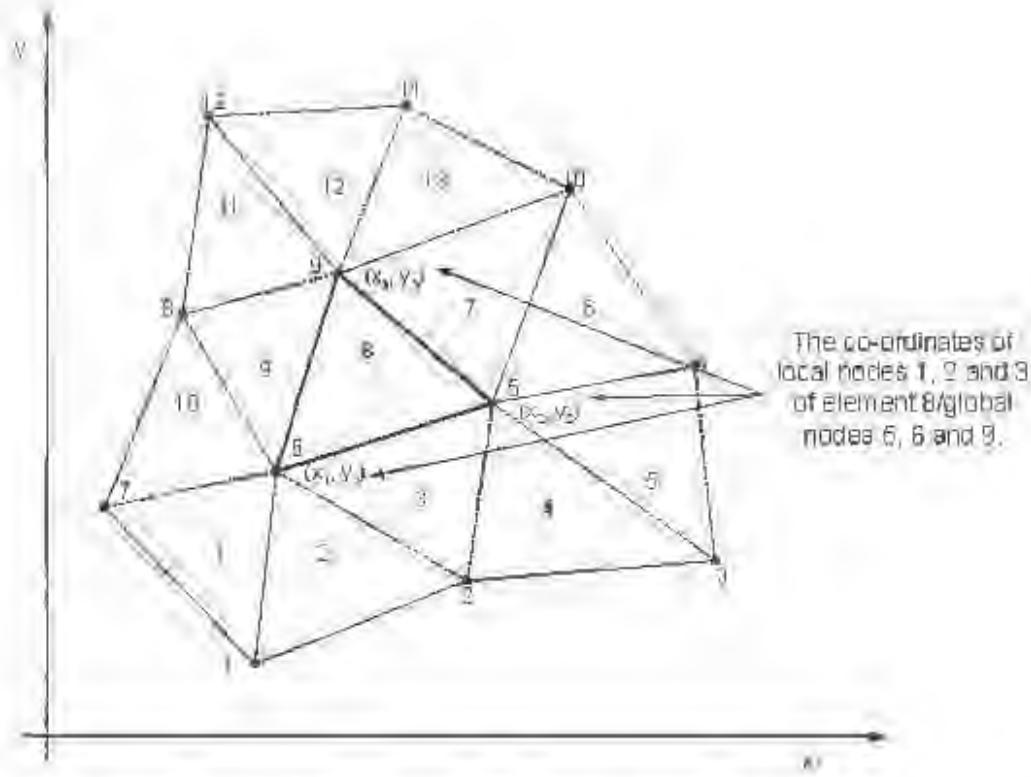


Figure 3-2: A 18 element, 12 node mesh. Numbers in the middle of elements are element numbers; numbers close to nodes are global node numbers. Note that each element has 3 local nodes, which may also be local nodes of adjacent elements. The local nodes of each element are used to create each specific element's stiffness matrix. The inter-element sharing of nodes necessitates the formation of the global master matrix which conforms to the global node numbering scheme.

concept of 'mesh bandwidth'. The mesh "bandwidth relates to the maximum difference in global node numbers within an element" [2]. When solving the inverse problem (discussed further in section 3.4) the bandwidth will greatly affect the memory space used for storing the sparse matrices typical to inverse problems. This in turn affects convergence time of the algorithm.

3.3.4 Solving the FEM equations

- The most common measurement strategy in EIT - the Neighboring Method (described in section 2.3) - results in known currents being injected into a medium and unknown voltages being captured at the boundary. Thus, in simulating this measurement protocol, we would solve for the unknown boundary voltages given the known injection currents by performing the following operation:

$$v = Y^{-1}I \tag{3.7}$$

  here v and I are the voltage and current vectors respectively. v is a N *by* 1 array containing the voltages of each of the N nodes in the mesh. I is a 1 *by* N array containing each node's net current flow.

- In the case of the measurement strategy employed in this project (known voltages applied and unknown currents measured at the boundary), a different approach is necessary. The calculation which needs to be performed is:

$$I = Yv \tag{3.8}$$

- To solve this equation requires the enforcement of the problem's boundary conditions. Consider the voltage and current vectors (v and I). Certain nodes in the mesh have boundary conditions prescribed by the application of voltages on the boundary. Others are 'free to vary' [9, 26]. If we rearrange the voltages in v such that the prescribed voltages are separated from the 'free to vary' voltages, and then rearrange the I and Y

matrices accordingly, we end up with the following:

$$\begin{pmatrix} Y_{ff} & Y_{fp} \\ Y_{pf} & Y_{pp} \end{pmatrix} \begin{pmatrix} v_f \\ v_p \end{pmatrix} = \begin{pmatrix} I_f \\ I_p \end{pmatrix} \tag{4.6}$$

where the subscripts $_p$ and $_f$ refer to 'prescribed' and 'free to vary' respectively.

- The rearranging of I will follow suit of v's rearrangement as those nodes which have a prescribed voltage are forced to have a net inward or outward current flow. The remaining nodes' currents (I_f) will all be zero due to the assumption of an equivalent current sources being present within the mesh.
- Using standard matrix algebra, the unknown boundary currents (I_p) can be solved for. See [26, 9] for details.

3.4 Solving the inverse problem

Given the limited number of known boundary conditions captured from an EIT electrode ring, it is the task of an imaging algorithm to reconstruct, within some error threshold, the conductivity or permittivity distribution[2] within the imaging space. Generally speaking, conventional reconstruction algorithms seek to find the inverse of a 'Sensitivity matrix' (or 'Jacobian matrix' in ERT) which describes how the changes in boundary voltages or currents are related to changes in each pixel's resistivity or permittivity [15, 29].

Many methods of obtaining the Sensitivity matrix in EIT exist – e.g. Linear Back-Projection Method, Perturbation Method, Double Constraint Method etc. - the most common of which are well documented in the literature (Hua and Woo [24] provide a good overview of these). All authors agree however, that solving the inverse problem is a highly complex and nonlinear problem.

As stated in section 2.4.1, the most widely used and superior algorithm for reconstructing images in EIT is the Modified Newton-Raphson (MNR) method. For this reason it is the only conventional technique discussed here.

[2] In the case of ERT, a conductance distribution is appropriate, whereas in Electrical Capacitance Tomography (ECT), a permittivity distribution would be more apt.

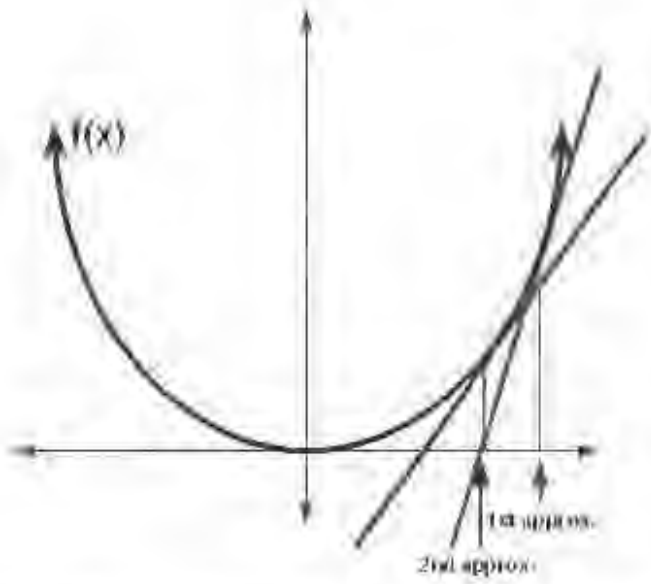


Figure 3-3: A diagram illustrating Newton's method. We seek to minimise the function $f(x)$, start by guessing an arbitrary value of x, evaluate the function and calculate the equation of the tangent line at that point. The next guess for x is the y-intercept of this tangent. Repeat until convergence.

3.4.1 The Newton-Raphson Method

"The Newton-Raphson algorithm is an iterative reconstruction algorithm specifically developed for nonlinear problems" [24]. The algorithm seeks to minimise the error between measured boundary voltages or currents and those predicted by a forward problem solver (usually the FEM). It is comparable in principle to Newton's Method, where an arbitrary guess to the root of a function serves as an initial trial solution. The function is evaluated at this guessed value and the tangent (or derivative) of the function at that point found. The guess is then updated in the direction of decreasing gradient. This process is repeated until the error criterion has been reached [9]. A diagram adapted from [9], illustrating this concept appears in figure 3-3

The Jacobian Matrix and the Update Equation

The Newton-Raphson algorithm defines the error function in EIT as the mean squared error between the measured and computed boundary values:

$$\Phi(\sigma) = \frac{1}{2}(f(\sigma) - f_0)^T (f(\sigma) \quad f_0) \tag{3.10}$$

where $f(\sigma)$ is the computed current or voltage response for a given conductivity distribution σ and f_0 is the measured response to the real (unknown) conductivity distribution.

In order to find σ which minimises Φ, we set the derivative Φ' to zero, i.e.,

$$\Phi'(\sigma) = [J'(\sigma)]^T (f(\sigma) - f_0) = 0 \tag{3.11}$$

where $J'(\sigma)$ is called the 'Jacobian matrix'.

After taking a Taylor series expansion of $\Phi'(\sigma)$ and cancelling insignificant terms, the update equation (i.e. the equation which tells us in which direction of σ to update our guess, so as to head in the direction of decreasing gradient of the error function) is:

$$\Delta\sigma^k = -[J'(\sigma^k)^T J'(\sigma^k)]^{-1} [J'(\sigma^k)^T] f(\sigma^k) - f_0] \tag{3.12}$$

where ^{k} is the conductivity element number [0, 4, 24].

A flow chart adapted from [24] of the algorithm is depicted in figure 3-4.

Ill-conditioning and the need for regularisation

The nature of tomography itself produces system matrices[2] which are said to be 'ill-posed' or 'ill-conditioned'. A matrix is said to be ill-conditioned when the ratio of its maximum to minimum singular values is significantly large [23, 24]. The singular values of a matrix are proportional to the lengths of the axes delineating the smallest ellipse (in 2-d) or ellipsoid (in higher dimensions) enclosing the data points represented by the rows of the matrix. Thus,

[2] System matrices referred to here are matrices which describe the mapping between known characteristics and boundary values of a given tomography system.

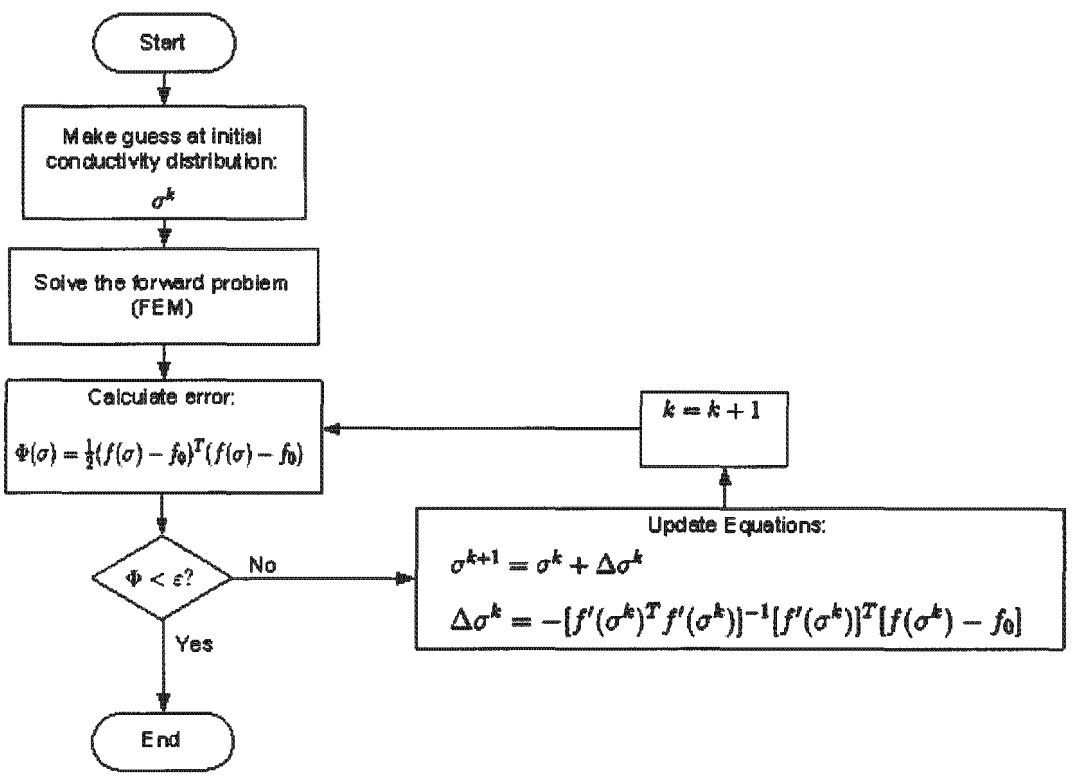


Figure 3-4: A flow chart of the Modified Newton-Raphson algorithm adapted from [24].

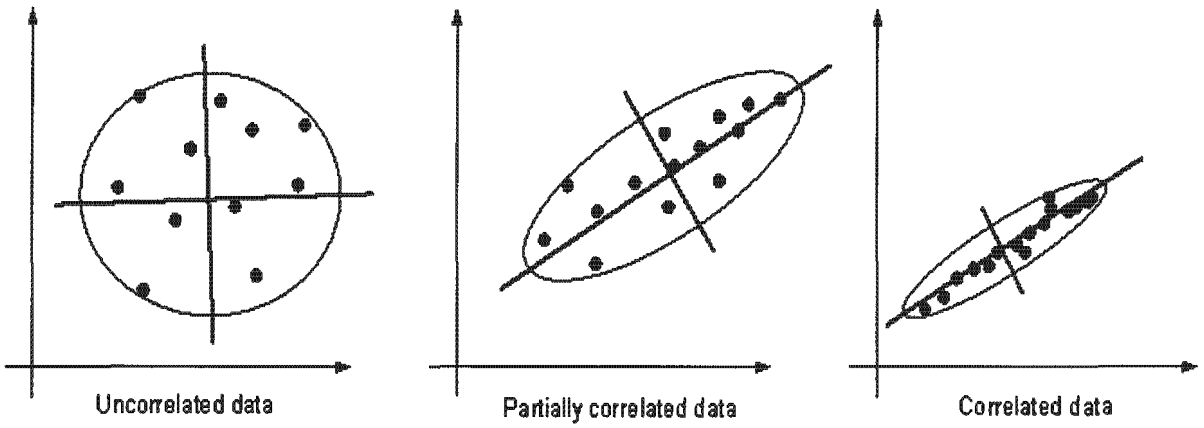


Figure 3-5: A diagram of 3 datasets: the first dataset is uncorrelated data, the second and third possess different degrees of correlation. Each dataset's encapsulating ellipse is drawn with major and minor axes in red and blue respectively. The greater the ratio of the variance of the data in these directions, the more ill-posed the data.

more ill-conditioned data can be contained in a more elongated ellipse/ellipsoid than less ill-conditioned data. A diagram illustrating these concepts appears in figure 3-5.

In ERT this can be explained as large changes in conductivity near the centre of the region resulting in small changes in the captured boundary measurements. It is thus intuitive to see that if such a system is ill-conditioned, then these small changes on the boundary will easily be swamped by noise. Thus, such a system will become insensitive to changes near the centre of the region.

This undesired phenomenon can be worsened by the measurement strategy employed. If a measurement strategy which promotes current to flow mainly near the boundary is adopted, then it is plain to see that little current can pass through the centre and hence little information about this region is contained in the boundary measurements. In reconstruction, ill-conditioning leads to conductivity values near the centre being spurious and irregular, as they have little effect on the objective function of equation 3.10. Thus, the need for some sort of "smoothing" arises.

Regularisation and Ridge Regression

The concepts, graphics and equations presented in this section are based on explanations put forward in [23]:

The task of updating the conductivity values of a FEM mesh so as to move in the direction of decreasing error, carried out by the MNR algorithm, is comparable to the problem of fitting a plane to a set of highly dimensional points in a least squared error sense; in matrix notation:

$$\mathbf{X}_{(p,n)} \times \mathbf{w}_{(n,1)} = \mathbf{t}_{(p,1)} \tag{3.13}$$

where $\mathbf{X}$ is a p *by* n matrix (p = *number of points* and n = *dimensionality of the input space*[4]), $\mathbf{w}$ is a vector of coefficients and $\mathbf{t}$ is a vector of target values. Target values being the desired values of the function at the corresponding data points in $\mathbf{X}$. Equation 3.13 is the general equation of a plane in $n+1$ dimensional space which passes through the origin. The shape of this plane is determined by the coefficients of $\mathbf{X}$ (i.e. $\mathbf{w}$), and this is what needs to be solved so as to determine the plane. If $\mathbf{X}$ were invertible, then the solution would be:

$$\mathbf{w} = \mathbf{X}^{-1} \cdot \mathbf{t} \tag{3.14}$$

$\mathbf{X}$ however, is often non-square (more equations than unknowns) and un-invertible. The solution is to form the 'Moore-Penrose pseudo-inverse' of $\mathbf{X}$. "It is easy to show by differentiation (and is a standard result in statistics) that the loss function [error function between fitted plane and desired target values] is minimised by:

$$\mathbf{w} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1} \cdot \mathbf{X}^{\mathbf{T}} \cdot \mathbf{t} \tag{3.15}$$

"[23]. Here, $(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1} \cdot \mathbf{X}^{\mathbf{T}}$ is the Moore-Penrose pseudo-inverse (or just pseudo-inverse) of $\mathbf{X}$*[5]. Equation 3.15 represents the plane which fits the data in a least squared error sense. Note too, that it is identical in format to equation 3.12 - the update equation of the MNR

[4] 'Input space' is the n-dimensional space in which one of the points described by the n rows of $\mathbf{X}$ lives.

[5] It is easy to show that $(\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1} \cdot \mathbf{X}^{\mathbf{T}}$ represents an inverse since $((\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1} \cdot \mathbf{X}^{\mathbf{T}}) \cdot \mathbf{X} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1} \cdot \mathbf{X}^{\mathbf{T}}\mathbf{X} = \mathbf{X}^{-1}(\mathbf{X}^{\mathbf{T}})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{X} = \mathbf{X}^{-1}\mathbf{I}\mathbf{X} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$

algorithm repeated below for ease of comparison:

$$\Delta\sigma^k = -[f'(\sigma^k)^T f'(\sigma^k)]^{-1}[f'(\sigma^k)]^T[f(\sigma^k) - f_0]$$

where

$$\begin{aligned} \mathbf{X} &\dashrightarrow f'(\sigma^k) && \text{(Jacobian matrix)} \\ \mathbf{w} &\dashrightarrow \Delta\sigma^k && \text{(conductivity update vector)} \\ \mathbf{t} &\dashrightarrow f(\sigma^k) - f_0 && \text{(error)} \end{aligned}$$

Consider the simple 2-d datasets of figure 3-5 on page 30. Each data set of figure 3-5 can be represented by the *p by n* matrix of equation 3.13 ($\mathbf{X}_{(p,n)}$). The first data set will produce an $\mathbf{X}$ matrix of full rank (i.e. all columns/rows are linearly independent), meaning that an exact solution/plane can be found. The third data set, which is a correlated data set (i.e. 2 or more columns of $\mathbf{X}$ are linearly dependent), will produce an un-invertible correlation matrix ($\mathbf{XX^T}$), meaning that infinitely many solutions/planes exist. In the partially correlated case of the second data set, the correlation matrix (although technically of full rank) is ill-conditioned, meaning that there are many equally well fitting planes[6] and the solution is thus 'unstable'.

Regularisation is used to address this problem of multiple unstable solutions; a technique known as 'Tikhonov Regularisation' or 'Ridge Regression' (this thesis adopts the latter nomenclature) is used in most implementations of the MNR algorithm. It involves adding to the existing ill-conditioned $\mathbf{X}$ matrix, a set of virtual data points close to the origin - one on each axis. This has the effect of reducing the slope (smoothing) of the plane in all directions, except in those in which it is well defined. Just how close to the origin these points are placed - which directly effects the degree of smoothing - is determined by a scalar factor known as the regularisation parameter[7]. The addition of these virtual data points in matrix notation is:

$$\mathbf{X}_{regularised} = \bar{\mathbf{X}} + \gamma\mathbf{I} \tag{3.16}$$

[6] These many equally well fitting planes are local minima of the error function in the MNR algorithm.
[7] Typically, for normalised data, the regularisation parameter (γ) is in the order of 10^{-6}.

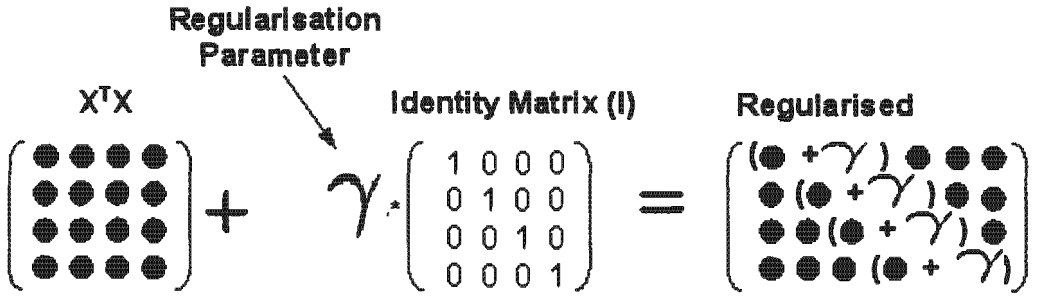


Figure 3-6: A matrix manipulation diagram showing how regularisation of a square correlation matrix ($\mathbf{X^TX}$) is achieved through 'Ridge Regression'. A small number (gamma) is added to the matrix's diagonal - in vector space this is equivalent to introducing a 'virtual data point', at a distance of $\sqrt{gamma}$ from the origin, on each axis of the input space.

where $\gamma = regularisation\ parameter$, $\mathbf{I} = the\ identity\ matrix$ and $\bar{\mathbf{X}}$ is the correlation matrix ($\mathbf{X^TX}$). This is depicted in the matrix manipulation diagram of figure 3-6.

Thus, when performing the MNR algorithm with regularisation, the update equation (equation 3.12) becomes:

$$\Delta\sigma^k = -[f'(\sigma^k)^T f'(\sigma^k) + 2\gamma\mathbf{I}]^{-1}[f'(\sigma^k)]^T[f(\sigma^k) - f_0] \tag{3.17}$$

where γ and $\mathbf{I}$ have the same meanings as in equation 3.16. Note that the expression $f'(\sigma^k)^T f'(\sigma^k)$ is analogous to the correlation matrix discussed above.

The 2-d datasets of figure 3-5 are redrawn in figure 3-7 with target values represented by the z-axis. The possible solutions/planes for each data set without regularisation are depicted to illustrate the need for regularisation.

3.4.2 A note on image resolution in conventional EIT systems

"The literature concerning measurement of spatial resolution in EIT is vague" [30]. Different groups often quantify image resolution by their own means, thus making it difficult to compare resolution between systems using a generalised measure. The task of quantifying image resolution in EIT is further complicated by its spatially variant nature - meaning that a single parameter is not adequate, as is the case with most imaging techniques. This spatial variance

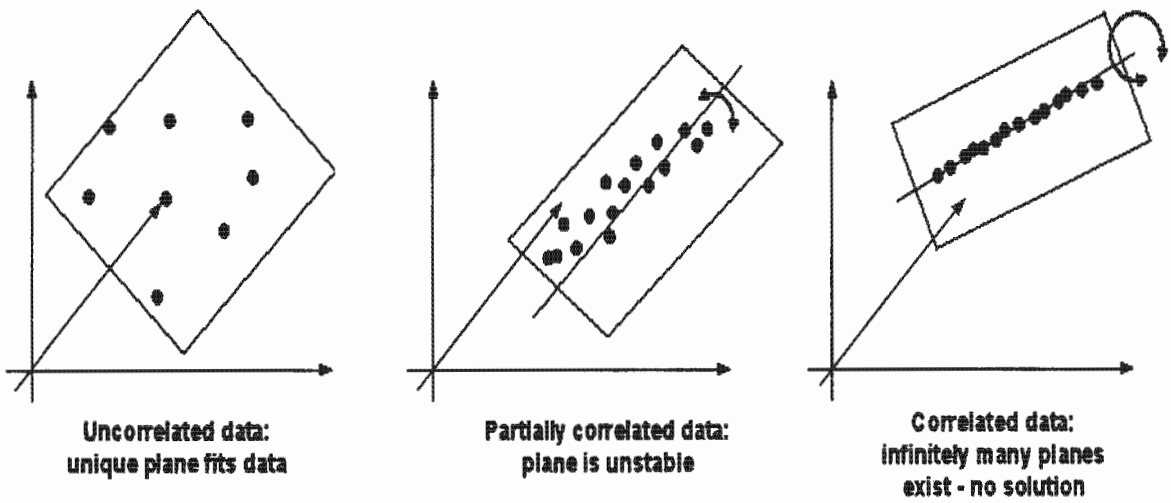


Figure 3-7: The datasets of figure 3-5 are redrawn with target values represented by the z-axis. The dataset on the left has a unique plane that fits the data, the dataset on the right has infinitely many such planes. The middle dataset (ill-conditioned) produces an unstable set of planes - local minima. Regularisation is the process of stabilising this plane by adding virtual points to the dataset.

in EIT is due to the non-uniform flow of current through the imaging medium, and is thus different for dissimilar current injection patterns [30]. In fact, the spatial variance in EIT is directly linked to - and can be explained in terms of - the ill-conditioning of the problem, where large changes in impedance near the centre of the region cause small changes in voltage at the boundary. Thus, it is possible to obtain much higher resolution near the edges of the measurement space (where the current density is high) than near the centre. In this discussion, resolution will be explained in terms of the number of elements for which the MNR algorithm is guaranteed to converge.

In EIT, the maximum number of unknowns (conductivity values) which the MNR algorithm is able to resolve, is equal to the number of linearly independent measurements available from the measurement space [9, 16]. This is because the heart of the MNR algorithm - the update equation (3.12) - relies on computing the inverse of the so called 'Hessian matrix' $(f'(\sigma^k)^T f'(\sigma^k))$; recall that $f'(\sigma^k)$ is the Jacobian matrix which holds information about the effects that each conductivity element has on the boundary measurements. This means that,

in order for convergence, the Hessian should be of full rank i.e. its rows and columns should be linearly independent [31].

As the number of elements in the finite element mesh increases, so the effect that each element's conductivity has on the boundary measurements decreases. It is thus intuitive that in a real system, as the resolution of a particular mesh increases, so the entries in the Jacobian and hence Hessian matrix will shrink - decreasing the signal to noise ratio and conditioning of the matrix. Note that this noise is present in real systems owing to such factors as measurement noise and modelling error. It would be naive to think that no noise is present in a computer simulation of an EIT problem; even the tiny error introduced by computer rounding is sufficient to limit the resolution of ill-conditioned problems.

It should be mentioned that 'cunning' methods of increasing the resolution of such images are known - especially when some *a priori* information about the measurement space is available. It has already been shown that regularisation is one way of improving the conditioning of the Hessian matrix, and therefore maximum resolution of an image (at the expense of image smoothing). Mesh-grouping is a technique where certain elements of the mesh are grouped together by forcing them to have the same conductivity [32]. This allows the mesh to contain a higher number of elements than independent measurements available. The algorithm will still converge if, in increasing the number of elements, the number of unknown conductivities has not changed. Obviously these methods significantly increase the computation time of such an algorithm.

In summary, the main limiting factor on resolution in EIT is the number of independent measurements available of the measurement space. If resolution is pushed beyond this, then the MNR algorithm has more unknowns than equations and will not converge. Regularisation makes convergence possible at higher resolution, but at the cost of image smoothing. The second important resolution-limiting factor in EIT is measurement noise, which heightens the ill-conditioning of system matrices.

Chapter 4

Neural Networks and Computational Intelligence for EIT Reconstruction

This chapter seeks to provide the reader with an overview of the reconstruction methods used by the research group in the past. This is because some of the techniques used previously are carried over to this project. Also, it will be important to have some understanding of the methods used previously when they are discarded or altered. Furthermore, background information on kernel methods is presented so as to prepare the reader for the results obtained in later chapters.

4.1 Introduction

Since the explosion of commercially available computing power, artificial learning techniques have been finding an ever increasing ensemble of real world applications. Simple algorithms can be used to teach intelligent systems, by example, to simulate the mapping performed by complex systems. Because such techniques are known to produce good approximators of highly complex, nonlinear functions, they may be useful in approximating the functional mapping between the internal conductivity or permittivity distribution within the measurement space in EIT, and the boundary measurements obtained from a typical tomography system. In other words, artificial learning methods have potential to solve the inverse problem discussed in the previous chapter.

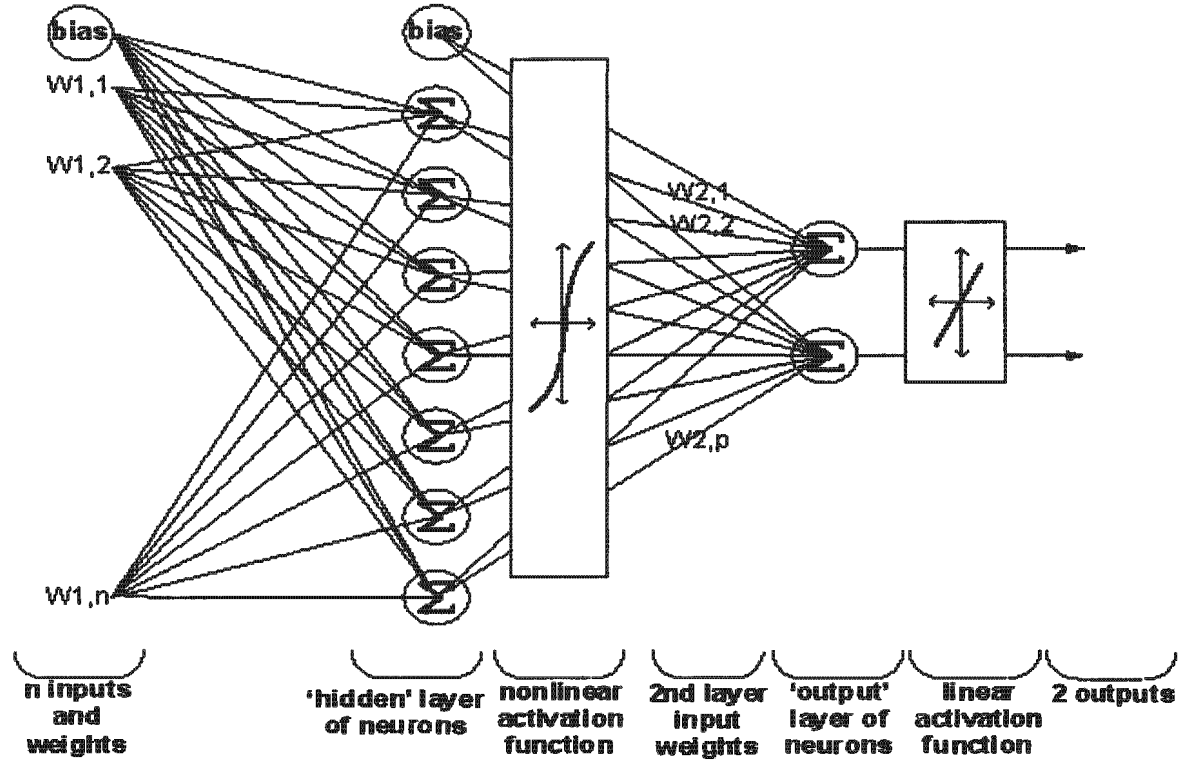


Figure 4-1: A connection diagram representing a common Artificial Neural Network architecture used in EIT. The specific nomenclature and symbols used are similar to that used by Teague [1, 4]. Note that each 'neuron' has such an activation function as represented by the singular blocks.

4.2 Neural networks for EIT

"Neurocomputing, parallel distributed processing, connectionist networks, artificial neural networks or simply neural nets, whatever the terminology, are all models of large numbers of simple but highly parallel and distributed processing elements with a high degree of connectivity between them "[21]. Throughout this document, the term 'Neural Network' (NN hereafter) refers to the classical architecture of a number of layered neurons or 'summers'. Each neuron performs the weighted summation of its inputs, and typically feeds the result to a non-linear activation function. An example of such a network appears in figure 4-1.

Indeed NN's have found a wide variety of uses in complex medical imaging tasks such

as feature classification, image restoration [33] and EEG[1] spike detection [34]. Research on applying NNs to the task of image reconstruction is however, less common - perhaps due to the fact that conventional methods (e.g. the MNR algorithm) have been the industry standard since long before NNs became widely accepted engineering tools.

In the available documented research on using NNs to solve the inverse problem in EIT, simulated impedance measurements are most often used in network training [19, 20, 21, 35, 36, 18]. In these, boundary measurements are simulated using a finite element modelling of systematically created impedance distributions. It was found in [18] that when testing the performance of these networks on a physical system, acceptable results were only attained if additional simulated noise - similar to the physical system's measurement noise - was added to the training data.

Hence, research into using "a new practical approach whereby the networks are trained and tested using real data from an existing impedance tomography system" [5] was initiated by the research group. The results and methods of the research are thoroughly documented in Teague's Ph.D. thesis [4] and summarised in [5]. The reconstruction technique developed has been applied to a number of research projects within the group since then [3, 7], and to the author's knowledge constitutes a unique reconstruction method for EIT applications. A brief description of how reconstruction is achieved follows.

4.2.1 Teague's reconstruction methods

This section presents the key issues required to gain a general understanding of how Teague (and the researchers who followed him) used NNs to perform reconstructions of the measurement space. Where appropriate, extracts and graphics are taken from Teague's Ph.D. thesis [4] and paper [14].

Training database population

Training data was obtained by recreating what a cross-section of a flow might look like in different 'snap-shot' scenarios. Rounded polystyrene cylinders and pockets of gravel submerged in seawater were used to reproduce instances in time inside an air-gravel-seawater flow. These

[1] A graphical record of electrical activity of the brain; produced by an electroencephalograph.

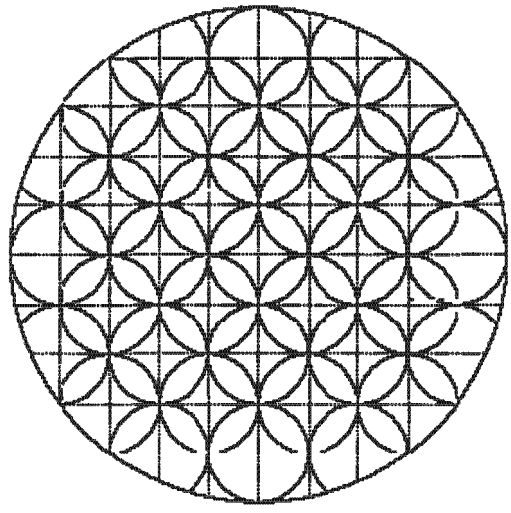


Figure 4-2: A diagram illustrating the individual training bubble positions within the imaging vessel. Circles represent a bubble position, lines form a 10 by 10 matrix of pixels which are used to set the desired network output (target).

contiguous masses of polystyrene or gravel are referred to as 'bubbles' in [4, 5] and throughout this document. Boundary measurements would be captured for different bubble positions - "Essentially, the smallest identifiable element of one phase, or bubble, is placed in a specific location in the second phase [seawater] and the readings are taken. The bubble is then moved to a different position and the readings are taken again. This process is repeated until the superposition of all the individual bubble positions effectively covers the cross-section of the vessel." [5]. A diagram illustrating the individual bubble positions taken from [5] appears in figure 4-2.

Due to the static nature of the training database (i.e. the fact that boundary measurements are taken while the medium is stationary), the networks' desired output or target values were manually recorded by looking at the scenario from above. The 'trainer' would, via a graphical user interface (GUI), record the entries of an 88 pixel image as being either air, gravel or seawater. A screen capture taken from [4] of the GUI used for training appears in figure 4-3.

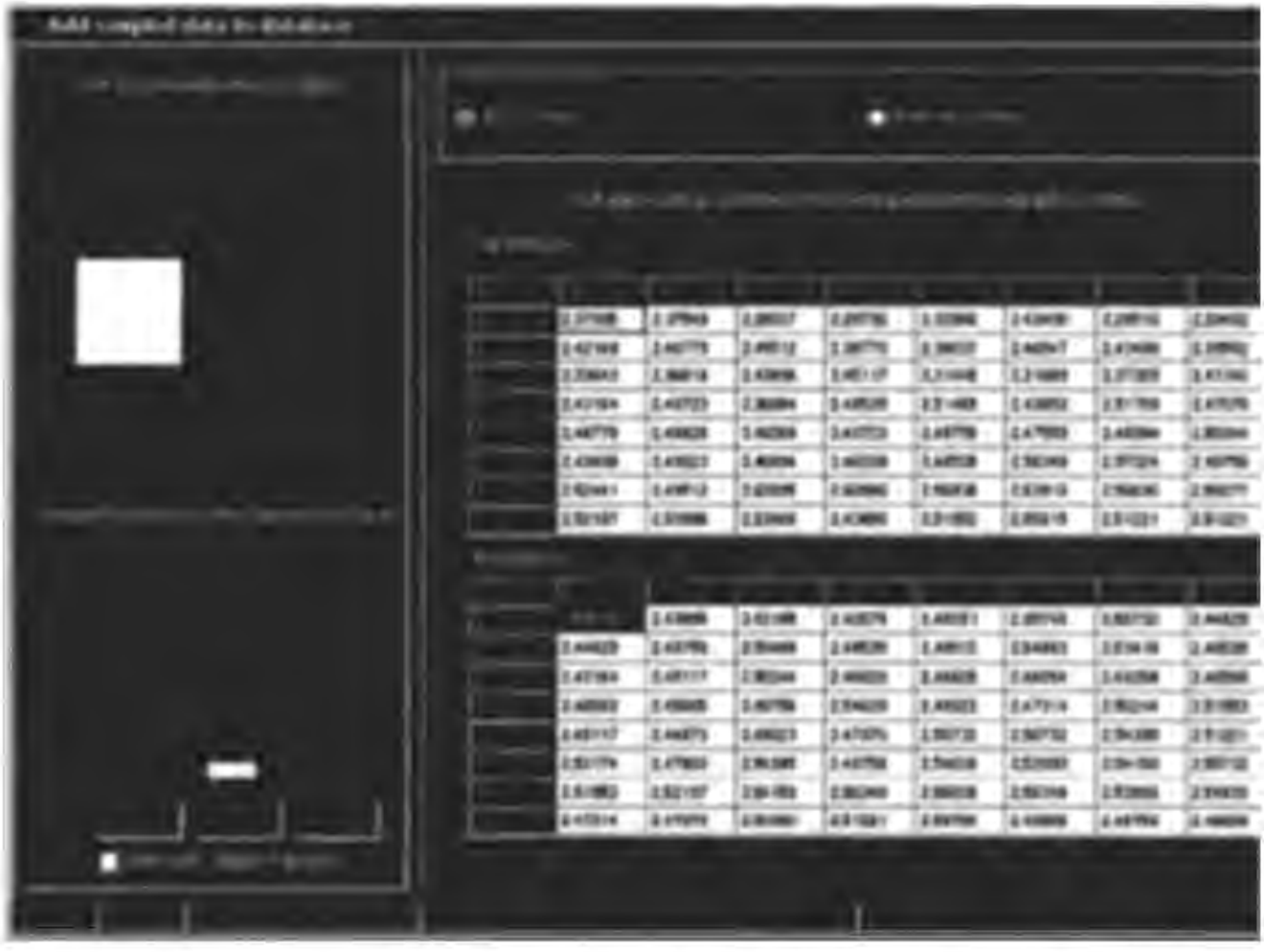


Figure 4-3: A screen capture of Teague's GUI used for static network training database formation. This same interface was used by Giannopoulos and Capindissa in their research.

A note on the performance measure used to train neural networks

Since the research group started investigations into neural networks for EIT, the performance measures used to determine the generalisation ability of such networks have been called threshold error and volume fraction error. An extract from [4] describes how these quantities are obtained:

" 1. Threshold error, which is the percentage of pixels that are not predicted correctly. For example, if the network predicts that a particular pixel is gravel when it is actually air, then this represents an error. These errors are summed over the entire database and are divided by the total number of pixels in the database. This performance measure applies only to the image reconstruction neural network and gives an indication of the positional correlation between the desired network output and the network prediction.

2. Volume fraction error, which is the percentage error of the volume fraction predictions for each of the three phases. For example, if the image reconstruction neural network predicts that there are 42 pixels of seawater when there are actually 44, then this represents an error of two pixels. These errors in the seawater prediction are summed over the entire database and are divided by the total number of pixels in the database to give the seawater volume fraction error. For the volume fraction predictor neural networks, the absolute values of the differences between the desired and the predicted volume fractions are summed over the entire database and then divided by the total number of frames in the database. It therefore represents the mean absolute error of the volume fraction predictor neural network for a specific phase. This measure is independent of pixel position and operates simply in terms of volume. It gives an indication of the ability of the network to predict volume fractions accurately."

Neural network image reconstruction

Teague researched the use of Multi-Layer Perceptron (MLP) NNs as well as Radial Basis Function (RBF) NNs to predict the desired output image. In both cases, network architecture was such that 88 outputs were available - each output representing a pixel in the image. Each pixel would be predicted by the networks to be either air, gravel or seawater.

RBFs were found to be inferior to MLPs and were thus discarded early in his research. Teague's best results for static, 3-phase image reconstruction were obtained using a single-layer

feed-forward neural network. These results (documented in [4, pp.75]) are tabulated in table 4.1.

Performance Measure	Single-Layer Feed-Forward Neural Network
Threshold error (%)	2.69
Air volume fraction error (%)	1.22
Gravel volume fraction error (%)	1.49
Water volume fraction error (%)	0.80
Sum volume fraction error (%)	3.50

Table 4.1: Teague's results for volume fraction predictions using a single-layer feed-forward neural network. The training algorithm used was the Gradient Descent method. The entries of the table are the mean absolute errors between desired and predicted volume fractions of a test database (reported as a percentage of the pipe's cross-sectional area).

Interestingly, networks trained using only 2-phase data (i.e. either bubbles of air *or* gravel submerged in seawater) could successfully generalise to 3-phase bubble distributions (i.e. bubbles of air *and* gravel submerged in seawater).

Neural network volume fraction prediction

Research preceding [4] had shown that greater volume fraction prediction accuracy could be obtained by training neural networks to predict the volume fractions directly [1]. Furthermore, this type of network can achieve results faster than if performing an image reconstruction first [1, 4, 5]. This is because fewer output neurons were necessary (2 as opposed to 88 for image prediction), and hence far fewer input weights to this final layer needed to be found. It was found that the best results could be obtained using a double-layer feed-forward neural network, the hidden layer of which contained 25 neurons. These results appear in the following table 4.2 [4, pp.77].

Performance Measure	Double-Layer Feed-Forward Neural Network
Air volume fraction error (%)	1.15
Gravel volume fraction error (%)	1.49
Water volume fraction error (%)	0.86
Sum volume fraction error (%)	3.50

Table 4.2: Teague's results for volume fraction predictions using a double-layer feed-forward neural network (Multi-Layer Perceptron). The single hidden layer consisted of 25 neurons, each having a sigmoid activation function. The training algorithm used was the Resilient Back-Propogation method. The entries of the table represent the mean absolute errors between desired and predicted volume fractions of a test database (reported as a percentage of the pipe's cross-sectional area).

4.3 Kernel methods

Briefly, kernel methods (as they are called in the computational intelligence community) refer to those techniques of function approximation or classification, where kernel functions[2] are used to map the training data into a higher dimensional feature space.

Although Teague investigated and ruled out the use of RBF NNs (which are a kernel method), re-directed project goals and a new training method proposed by the author have inspired a re-opening of the investigation into the ability of kernel methods to perform volume fraction prediction in EIT.

4.3.1 Justification for further investigation of kernel methods

The reason given by Teague for the poor performance of RBFs (and hence kernel methods) is that they suffer from the 'curse of dimensionality' "which specifies that the number of training data points required to 'fill' the input space grows exponentially with the dimensions of the input space" [23, 37]. In fact, using the database population techniques described in section

[2]The exact definition of a kernel function is complex and highly mathematical. For the purposes of this thesis it is unnecessary to define such a function, and is sufficient to realise that the Gaussian/Normal distribution is such a function.

4.2.1, the time taken to suitably fill the input space would be in the order of months! This is due to the fact that Teague's GUI (figure 4-3) and training software was focused on image reconstruction i.e. the position of training bubbles in the measurement space was important, and played a significant role in network training. Thus, the time taken to set up each individual bubble configuration, sample the boundary voltages, record the target values of each of the 88 pixels and reposition the bubbles, makes training a laborious and tedious task.

Because of the improved results achieved by Teague [1, 4] when predicting volume fractions as opposed to image predictions, the focus of the research shifted to predicting volume fractions directly. Although this was the case, researchers following Teague (Giannopoulos [3] and Capindissa [7]) continued to use Teague's database population method designed for image reconstructing networks. This meant that results were still slow in coming forth, and the curse of dimensionality remained a limiting factor in the types of intelligent computing techniques investigated.

4.3.2 Kernel methods

Section 3.4.1 likened the task carried out by the MNR algorithm to fitting a plane to a number of data points using simple linear regression techniques. Even when regularising EIT data however, it will be impossible to find a plane which fits such data suitably using linear regression. This is because the mapping between the data (measured boundary voltages) and target values (desired volume fractions) is highly non-linear, deeming linear regression useless; unless some non-linear transformation is applied to the data first.

Kernel methods are capable of achieving highly irregular, nonlinear function mappings by placing on each data point (in the training database) a kernel function. These kernels are functions of the euclidean distance[3] between all points in the training data. The most common and versatile kernel is the local Gaussian function[4], which is characterised by a 'width' or 'spread' factor [23]. The kernel width parameter (usually called σ) is a parameter which needs to be tuned by trial and error. Different datasets will have different optimum σ values which

[3] Euclidean distance between two vectors (x and y) is defined as: $D = \sqrt{\sum(x_k - y_k)^2}$. In 2-d, Pythagoras' theorum is a special case.

[4] Although kernels are by no means limited to this case, they are the only kernels used in this thesis.

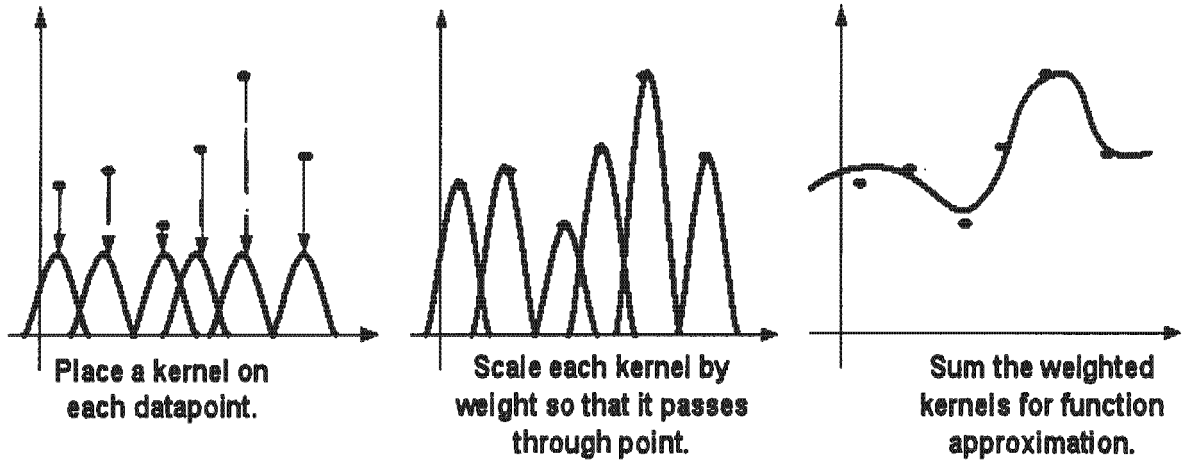


Figure 4-4: A series of diagrams showing the basic principles of kernel methods. The diagram on the left shows how Gaussian bumps are 'placed' on each data point in the training data - note that all bumps have the same width factor (σ). The middle sketch shows how each bump is scaled according to the target value corresponding to the input data point. The diagram on the right shows how the summation of the smooth, scaled Gaussian bumps results in a function which can be used to generalise to new data.

give rise to the best classification or regression error. Most datasets however, produce similarly shaped σ vs. error*[5] curves, with a sharp notch in error occurring around a certain value of σ.

Each of these kernels is then scaled by a weight, or α coefficient, and the linear sum of all kernels is formed. This weighted sum of kernels becomes the classification function - in classification problems, and the regression function - in regression problems. In both cases, the α weights applied to each kernel are adjusted so that the kernel has approximately the target value (+1 or -1 in classification problems, or a real number between 0 and 1 in regression problems) at each data point. The resulting function, as a superposition of smooth basis functions, is itself smooth; hence it can generalise or interpolate to new instances [23]. These concepts are illustrated in figure 4-4 extracted from [23].

The way in which these kernels are mathematically 'placed on each data point', is illustrated in the matrix multiplication diagram of figure 4-5.

[5] 'Error' here is generalisation error: the error between the predicted values and the real target values in a test database.

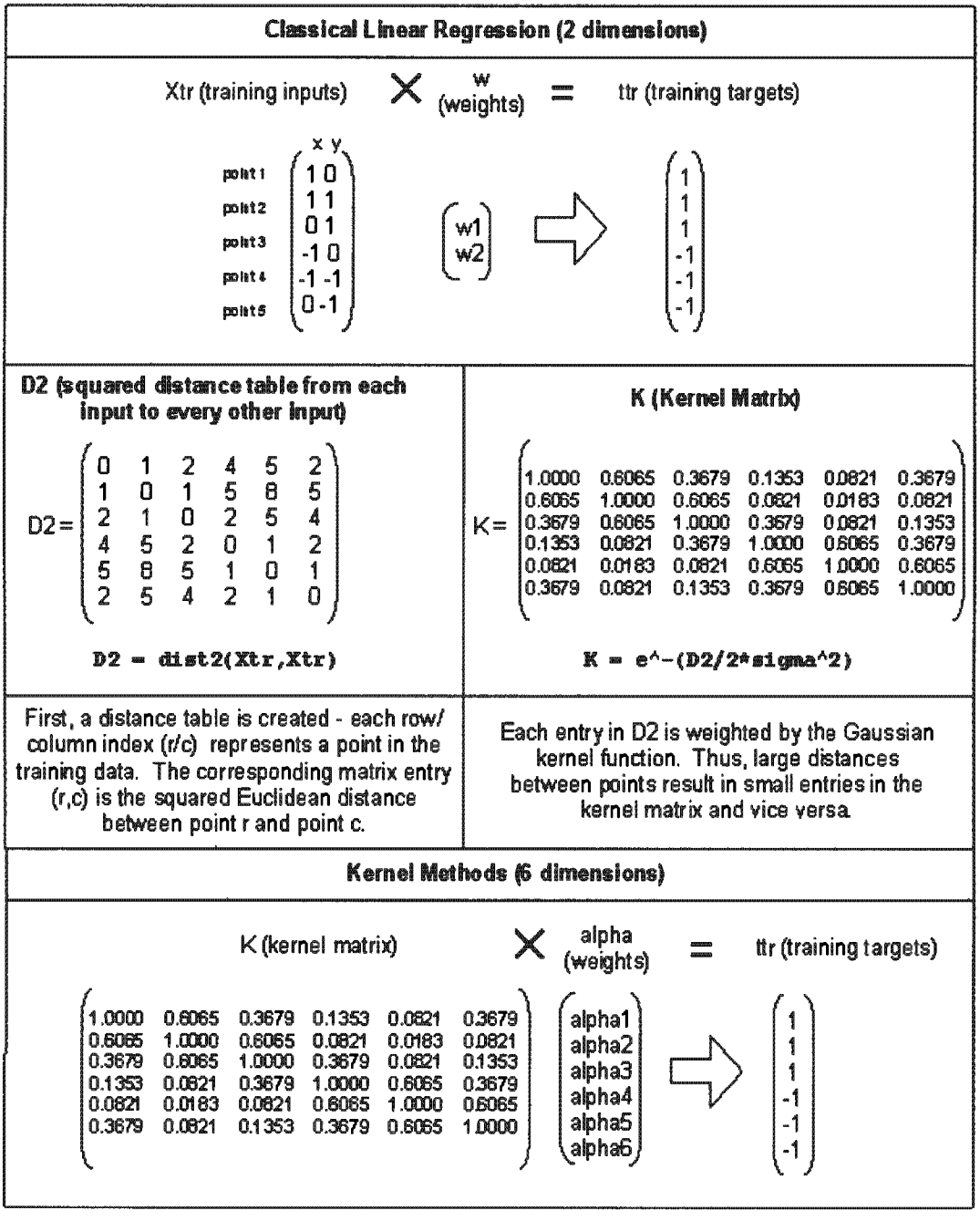


Figure 4-5: A matrix manipulation diagram illustrating how a simple 2-dimensional linear regression problem is transformed by the kernel function into a higher dimensional problem. The way in which kernels are 'placed' on each data point is demonstrated in the middle section of the figure.

4.3.3 Kernel Ridge Regression

The process of adjusting the weights, or α coefficients, of each kernel function can be done using a variety of methods. Various different kernel classification and regression 'machines' exist, and are distinguished by the different methods used to adjust these weights. Support Vector Machines (SVMs) for instance, iteratively adjust the weights so that the function passes as closely as possible through the training points. In the process many weights become zero, and only the 'support vectors' remain. SVMs are viewed as state-of-the-art in the computational intelligence field, yet their performance is virtually indistinguishable from a technique known as Kernel Ridge Regression (KRR) [23].

Simply, how the weights (α values) are found in KRR, is by using the methods of classical linear regression i.e. matrix pseudo-inversion (described in section 3.4.1). This process is however ill-conditioned - tending to overfit data - hence, regularisation is added.

In MATLAB, which is the author's KRR development platform of choice, pseudo-inversion of matrices is performed very robustly when using the backslash operator:

$$\mathbf{w} = \mathbf{X}\backslash\mathbf{y}, \text{ or in the case of KRR: } \alpha = \mathbf{K}\backslash\mathbf{ttr} \tag{4.1}$$

MATLAB'S backslash operator (\) is an 'overloaded' one, which performs many different functions. If the $\mathbf{X}$ or $\mathbf{K}$ matrix is un-invertible or ill-conditioned (as in the EIT case), the above notation will achieve a minimum-norm, least squared error solution to the equations. This is done (very robustly) using 'Singular Value Decomposition' [23], which is described further in section 7.1.2. Thus, no need of hardcoding the Moore-Penrose pseudo-inverse will be necessary when finding the α-weights, although it will be necessary to perform regularisation (Ridge Regression) on the kernel matrix before pseudo-inversion; hence the name Kernel Ridge Regression.

Once the α-weights which best map the transformed training data to its corresponding target values have been found, then predictions can be performed on the unseen 'test data'. Firstly, the same process as followed in mapping the training data into a higher dimensional space needs to be applied to the test data (i.e. a distance table made between each testing point and each training point, followed by application of the kernel function). Once the test

data has been mapped into this higher dimensional space, the α-weights (found previously) can be multiplied by the transformed test data to arrive at predicted target values.

Below are a few lines of MATLAB pseudo-code for implementing KRR. The reader should not let the simplicity of the code detract from its power.

```
function yte = krr(Xtr, ttr, sigma, gamma, Xte)
%
% usage:  yte = krr(Xtr, ttr, sigma, gamma, Xte)
% Xtr is the training data matrix - rows are instances, columns features.
% ttr is a vector of the associated target values.
% sigma is the kernel width, gamma the regularisation parameter.
% Xte is the test data (same format as Xtr).
% the predictions from Xte are returned in yte.
%
D2=dist2(Xtr,Xtr);
% creates a distance table of all the inter-training point squared
% euclidean distances.
K=e^(-D2/(2*sigma^2));
% the kernel matrix using the Gaussian kernel is a function of the
% distance matrix.   This is what 'places' a kernel on each data point.
K=K + gamma*IDENTITY_MX;
% kernel matrix with ridge regression (regularisation).
alpha=K\ttr;
% finds kernel weights using MATLAB's robust pseudo-inversion operator (\).
D2=dist2(Xte,Xtr);
% distance table of test-training point squared euclidean distances.
K=e^(-D2/(2*sigma^2));
% kernel activations at test points.
yte=K*alpha;
% function value at test points.
```

4.4 A note on resolution in neural network-type EIT systems

A discussion of image resolution was included in section 3.4.2. The topic was discussed in terms of the maximum number of unknowns which would result in an invertible Hessian matrix - the requirement for convergence of the MNR algorithm. Since intelligent computing techniques are function approximators, and are not directly involved with equation solving, it is difficult to predict at what resolution their methods will collapse. This is the main drawback of such techniques, i.e. they do not provide the outside world with information about the function they approximate, and hence it is difficult to analyse their mechanisms.

Because the intelligent computing techniques used in EIT (such as the ones developed by Teague [1, 4]) are merely another method of solving the same inverse problem as that solved by conventional techniques (i.e. image reconstruction using the MNR algorithm), the same inherent sources of error contribute to the bounds on reconstruction resolution. Just as the MNR algorithm fails to converge for ill-conditioned Hessian matrices, so too does computational intelligence find it difficult to find a unique surface which fits the training data if it is ill-conditioned. Therefore, the first and foremost factor influencing resolution in EIT is the number of independent measurements available, which is determined by the number of electrodes and measurement protocol employed.

Given an EIT system with a certain number of electrodes, it is thus safe to say that, in an ideal noise-free measurement environment[6], the resolution of any EIT reconstruction method is determined by the data-acquisition hardware's ability to measure small changes in the boundary values, and the computational device's ability to store (and perform calculations with) these small changes. Of course, in the presence of measurement noise, the system resolution will deteriorate as these small changes become swamped in noise. Furthermore, once measurements have been digitised for computer manipulation, rounding errors will further encroach on resolution. Thus, when sampling boundary measurements from an EIT rig, the analogue-to-digital conversion resolution and number storage format should be chosen with the expected amplitude of measurement noise in mind. It is relatively easy to obtain high resolution

[6] Measurement noise in EIT, besides the effects of electromagnetic interference, is mainly due to unknown contact impedances between electrodes and electrolyte.

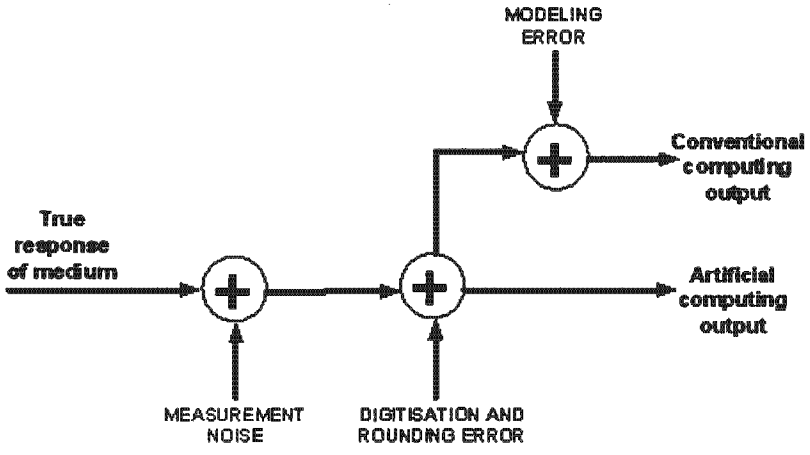


Figure 4-6: A diagram showing the sources of error introduced during the reconstruction process in EIT. The largest source of error is that introduced by measurement noise, which is mainly due to unknown contact impedances between the electrodes and the medium. Note that an additional source of error is introduced in conventional reconstruction techniques.

analogue-to-digital conversions and number storage formats; the main factor influencing system resolution is therefore measurement noise.

A diagram illustrating these concepts is presented in figure 4-6. The diagram illustrates a definite advantage of using an intelligent computing approach in EIT; a further source of error introduced by conventional techniques, but not by neurocomputing methods, is that of modelling error in the forward problem.

Chapter 5

Design and Implementation of a Dynamic EIT Measurement System

All the topics covered in this text so far have been theoretical in nature. Their purpose was to introduce the reader to the concepts of EIT relevant to the project - specifically the author's stage of it. Background information has been abundant in earlier chapters (and in Appendix B) so as to familiarise the reader with the state of the system and aims of the research - as received by the author. All concepts presented have been based on the work of previous researchers on the project or in the literature. On the contrary, all that follows is work done, by the author, using the tools gained through studying the literature and investigating the constructs presented so far.

This chapter is concerned with the practical issues involved with the design and implementation of a dynamic EIT measurement system. More specifically, the practicalities involved with undertaking such a task (given the present state of the system) are: the design and implementation of an open-ended electrode ring, the implementation of a new data capture system and the upgrading of the current hardware.

5.1 New direction of the project

At a meeting with the project sponsors (25 August 2004), it was decided that the project needed to become more focused on the practicalities involved with building a new electrode

ring; one which could be fitted into the Flow Process Research Centre's pumping loop at the Cape Peninsula University of Technology (CPUT). The main decisions made at that meeting, concerning the direction of the project appear below:

1. An electrode ring capable of handling real flows must be designed and built as soon as possible.

2. Since improved results had recently been achieved through Herholdt [9] extending his point electrodes to line electrodes[1], the effects of performing a similar alteration to the design of a new electrode ring should be investigated.

3. Due to the issues raised in section 2.6 on page 17 about the training of neural networks with dynamic data, a purely intelligent computing (or black box) type approach may not be the best route to follow. The project needs to become more inclusive of previously avoided, conventional image reconstruction techniques.

4. Emphasis must be placed on designing a system to output volume fraction data (as opposed to image data). It is however understood that image data is a useful tool in research of this nature, and should not be sidelined.

5. For the sake of simplicity the dynamic measurement system should be a single-plane, unimodal (resistance data only) system. The decision to use only resistance data is further justified by the fact that capacitance data is mainly useful in distinguishing between phases of different permittivity (i.e. rock and air); resistance data has proved sufficient for distinguishing between phases of different conductivity (e.g. water and air/rock). Because the final application seeks a volumetric flow of gravel[2] rather than air, the electrode ring can be placed below the air injection point, and measure a purely gravel water mixture. The reconstruction of such a mixture should be possible with only resistance data.

[1] This is discussed in detail in section 5.4.1.

[2] This gravel is the whole reason the airlift exists - it is what DeBeers are after - and hence why the flow rate of *gravel* needs to be monitored. Also, the amount of air being injected is a known in any case.

5.2 Before design, some careful considerations

When designing a new rig, many design choices must be made which affect the future of the project. Specifically, hardware design choices will play a significant role in the successful exchange of the project from the author to his successor. This has been learned through the difficulties experienced in reviving the dormant hardware of Teague, and analysing Giannopoulos' upgraded version.

In fact, Giannopoulos *did* make provisions for his work to be built upon and extended. It is in this spirit, that whenever a design choice is made, careful consideration of the project's future is taken. Thus, in what follows, all decisions and design choices are justified in terms of their extendability - not necessarily to the ultimate goal, but to the next stage of the project.

5.3 Data acquisition

In previous research, both Teague and Giannopoulos had used the EAGLE PC30G data acquisition card to acquire their data. Teague had the following to say about the card: "This card is intended for an ISA slot and is soon to be obsolete since it is increasingly difficult to purchase computer motherboards that support the ISA standard" [4]. The research group recently obtained the much more user friendly DATA TRANSLATION DT9802 card. It is a USB data acquisition module with high functionality. The module is supplied with the usual software development kit (SDK), which contains numerous libraries with hosts of functions which can be called to control the card's sub-systems. More attractive to the end user is a popular development tool called DT MEASURE FOUNDRY. At first this seemed attractive because of its user friendly GUI with which no hard coding is necessary, and procedures such as exporting data to Matlab are performed with mere touches of buttons. However, a little experimentation revealed that the complexities of the sampling task at hand (namely: the high sampling rate, and need for sampling to be perfectly synchronised with external hardware) required the module to be hard coded in C.

5.3.1 Outside help

Unfortunately, with a purely electrical background, the author had never coded in C. So it was decided, with the aid of Prof. J. Tapson[3] that an outside source should be hired to produce some skeleton code on which the data acquisition software of the project could be based. An electrical/computer engineer (Tim Long) who recently completed his B.Sc. thesis in EIT and is currently doing his M.Sc. in EIT, was given the job. The author supplied Long with a list of requirements which the skeleton program should meet. All the requirements were based on specs obtained from Teague's Ph.D. thesis [4] i.e. number of samples to capture per frame, required frame rate etc. The specific requirements appear below:

1. 256 analogue voltages need to be sampled per frame.
2. The final application may require a *reconstruction* rate of up to 200 frames per second, therefore the data should be captured at least as fast as this.
3. Some way of controlling the address lines of 8 16-channel multiplexers needs to be realised. Preferably, the card should use one of its onboard counters to drive these address lines in parallel. Alternatively, as Teague and Giannopoulos did, an output port of the module could be used to communicate with an external sample controller board.
4. The code should be able to average the data of a number of frames e.g. capture 10 frames of data, and then save (as one frame) the average of these 10 frames. This would be useful to reduce noise effects.
5. The code should accept (as an input parameter) a mask, which tells it which samples are significant i.e. which samples to use in reconstruction.

Note that although it was decided at the August meeting that, initially only one plane of resistance data would be used (this corresponds to 64 samples), the requirements above categorically state that 256 samples are required per frame (which is the number of samples required for a 2-plane 16-electrode FDM system). The requirement for the code to accept a masking file as an input parameter takes into account that only one plane, or only one mode

[3] Project supervisor.

(resistance or capacitance) of one plane of data, might be needed by the user; in this case the user would simply enter a mask file corresponding to the desired samples. Keeping extendability in mind, this same code can be used for sampling any sequence of channels up to a 2-plane 16-electrode bimodal system.

5.3.2 Functionality of initial code

The completed skeleton code met all the requirements and ended up controlling the multiplexer address lines via an external sample controller board - a board which still needed to be designed and built. This was done as opposed to the requested use of onboard counters due to their serial nature (i.e. the counter values are output serially, on one pin) which would be ill-suited to the high speed requirements of the sampling task at hand. The code appears in Appendix C.

5.3.3 Sample controller board

Long's skeleton code output a clocking signal on one of the external pins of the DT9802 device. This was to be fed to the device's ADC 'input trigger scan' pin, which triggers the internal ADC to sample all 16 analogue input channels on its rising edge. This clocking signal could then be simultaneously fed to the clock input of an external counter, which in turn would drive the address lines of the multiplexers on the demodulation boards. A block diagram of the sample controller board (designed by the author) and how it interacts with the measurement system appears in figure 5-1. Its circuit diagram and photograph appear in Appendix B.

The principle of operation of the sample controller board is similar to that used by Teague [4] and Giannopoulos [3], yet slightly simpler - with less chance of loss of synchronisation occurring. This is significant because Teague stated in his recommendations that: "there is an occasional loss of synchronisation between the sample controller board and the data acquisition card. This loss of synchronisation is the result of false triggering of the data acquisition card by noise on the external trigger line." His recommendation was to address the issue in future research, and although Giannopoulos used screened cable on his external trigger line, the cause of the problem (i.e. separate counter and trigger clock source) had still not been eliminated. In the present system, the sampling trigger scan and counter clock come from the same source

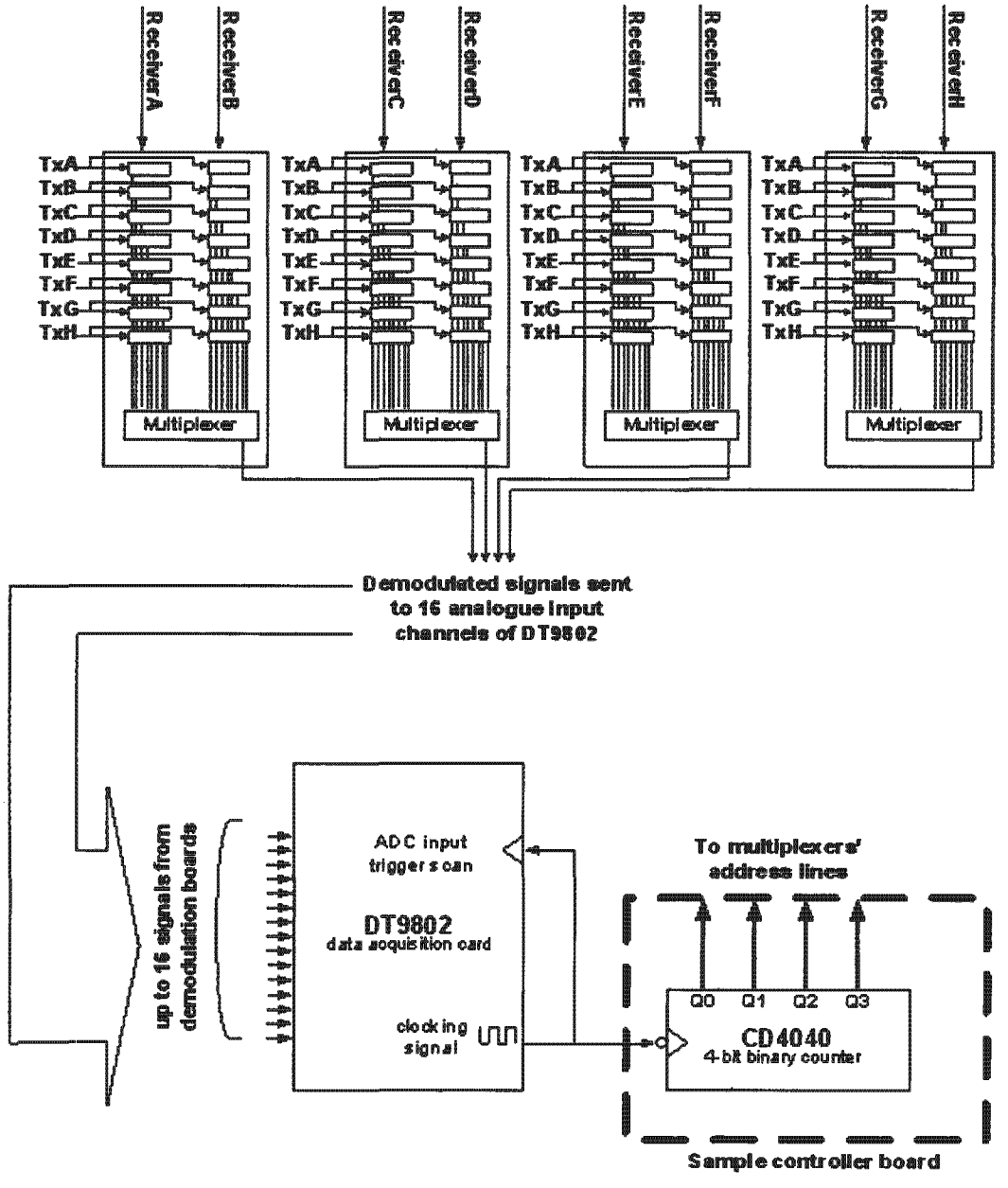


Figure 5-1: A block diagram showing the interactions between the counter on the sample controller board, the multiplexers' address lines and the analogue inputs of the data acquisition card. Note that Long's application (coded in the DT9802 flash memory) sampled the analogue input channels on the rising edge of an internally generated clocking signal. On the falling edge, the multiplexers would be incremented to switch through the next set of measurements on the demodulation boards. Note also, that only 4 demodulation boards are used in the present system; the data acquisition board however, has been wired to accept up to16 analogue inputs, i.e. the number of demodulation boards required for a dual-plane, bimodal, 16-electrode FDM EIT system.

(DT9802 output trigger), as opposed to an external clock source (implemented using a 555 timer in Teague's and Giannopoulos's sample controller). Thus, even if noise occurs on the trigger line, both counter *and* the trigger scan are effected, meaning that no loss of synchronisation occurs, only a bad data point.

5.3.4 Modification of code for training

The initial skeleton code was designed to capture - and average if requested - a certain number of frames and save the result in a .csv (comma separated variables) file. The application would then close. This would be adequate for real-time testing, however it was not adequate for training NNs using the methods used by Teague (and the researchers who followed him), where time would be needed between each frame in order to change bubble positions.

It turns out that the two tasks (real-time testing and training data capture) required to be performed by the card are quite different; they require totally different use of internal buffers and buffer interrupt routines. Thus, an appreciable portion of time was spent gaining an understanding of how the DT9802 performed its buffering operations.

There exists (now), a separate application suitable for training NNs in Teague's fashion using the DT9802 (see Appendix C). In fact, as the system stands, a different and more elegant method of clocking the external counter has been implemented. This uses a 'clever' function of the DT9802 which Long overlooked. It is called the 'digital dynamic output' capability and solves problems encountered when dealing with situations which arose when modifying the code to be used for training. The functionality is basically the same and is not discussed here; however, a figure showing a block diagram of the subsystems' interaction appears in figure 5-2, and an oscilloscope screen capture of the digital dynamic output pin and the counter output pins is presented in figure 5-3.

Note that all the commented data capture code developed for this project appears in the appendix where its functionality is briefly explained.

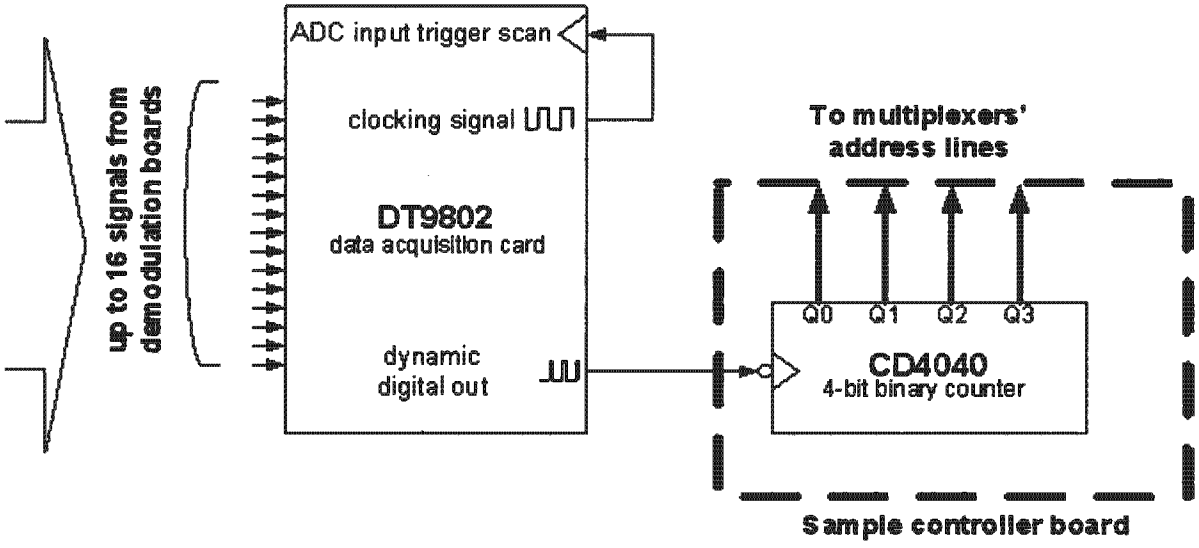


Figure 5-2: A block diagram showing how the dynamic digital output pin of the DT9802 is used to clock the external counter. The dynamic digital output pin outputs a narrow pulse each time the last channel is sampled, which happens at the end of every trigger scan.

5.4 Designing the electrode ring

5.4.1 Point vs. line electrodes

An undoubted improvement in EIT image reconstruction is achieved when using axially[4] extended point electrodes (line electrodes) in conjunction with a 2-d finite element model of the imaging plane [9]. This is because a simple 2-d model takes no account of fringing. Using line electrodes is a well known way of eliminating this fringing, or at least reducing it. The concept of how fringing is minimised when using line electrodes is illustrated in figure 5-4.

At design time, the question still remained as to whether the use of line electrodes would improve the predictions made by NNs in EIT. Thus, a practical investigation into the effects of training NNs using data collected from both line and point electrodes was initiated and is documented below.

[4]The axis being referred to here, and in what follows, is that of the centre line pipe being imaged.

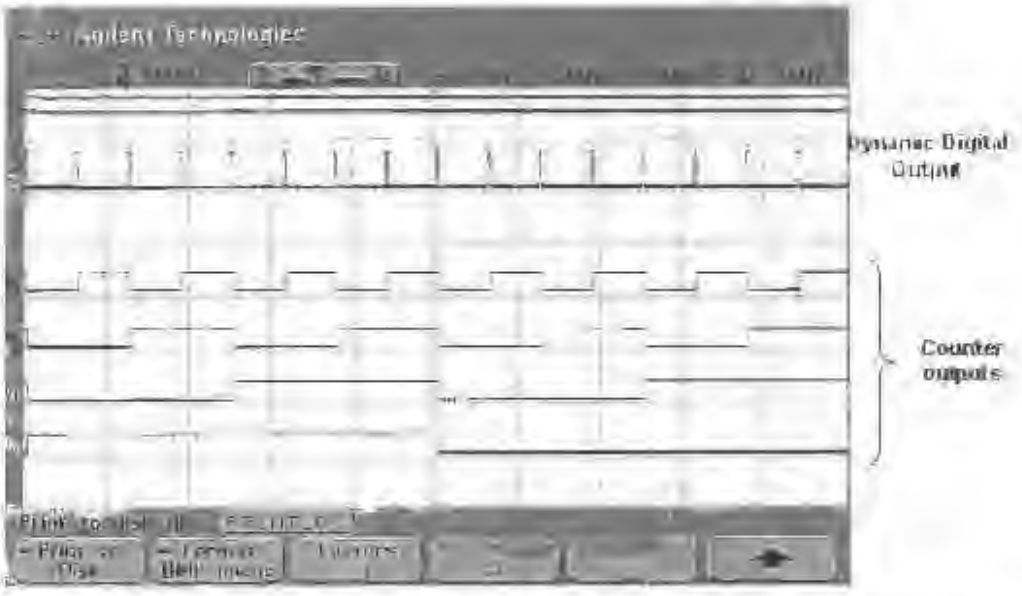


Figure 5-3: An oscilloscope screen capture showing the timing relationship between the data acquisition card and the sample controller board (which drives the multiplexer's address lines). The top trace is the dynamic digital output signal of the card, which outputs a pulse each time the 16th channel is sampled. The counter is incremented on the falling edge of this pulse.

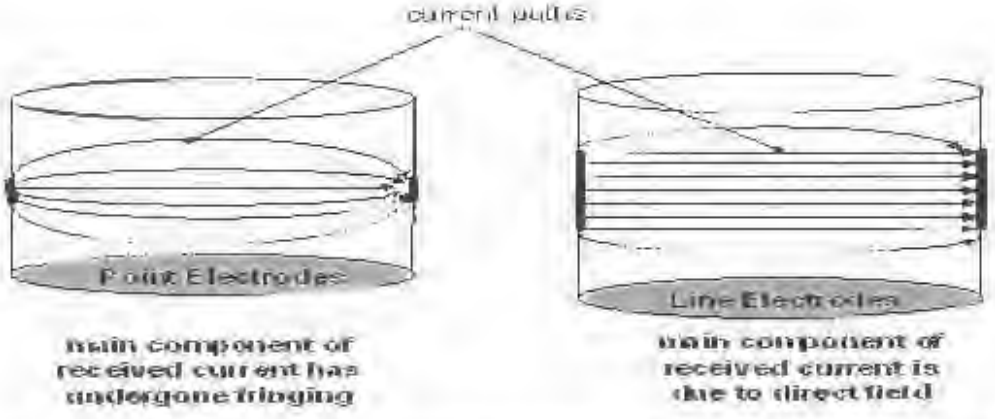


Figure 5-4: A diagram illustrating how fringing is reduced by using line electrodes. Note that on the left (point electrodes), a major component of the received field is due to fringing. On the right (line electrodes), the fringing component is significantly reduced relative to the main field.

Apparatus

- Ideally, tests should have been performed using Giannopoulos' improved version of Teague's electronics, as this is what was to be used in the dynamic rig. Unfortunately, the author's colleague (Vladimir Cupidlova) was performing tests of his own using this hardware at the time[5]. The principle of operation of Teague's (square-wave excitation) hardware is however identical to the sinusoidal excitation used by Giannopoulos, except that sinusoidal excitation reduces ripple at the demodulator outputs. Any improvement gained through line electrodes using square-wave excitation should intuitively carry through to pure sinusoidal excitation. Thus, Teague's sensing electronics were used.

- The electrode ring used was that of Teague's, as constructed for his Ph.D. thesis [4]. Only one plane was used, and only resistance receivers were connected (as decided upon in August meeting).

- Aluminium tape and copper wire were used to extend the point electrodes. A photograph of the extended electrodes appears in figure 5-5.

Method

- The same method of simulating bubble distributions as used in the forerunning projects [1, 2, 3, 4, 7] was utilised, i.e. rounded polystyrene cylinders representing air bubbles[6]. A photograph of some of the polystyrene bubbles used appears in figure 5-6.

- In both Teague's B.Sc. and Ph.D. thesis [4, 4], best results in predicting the volume fractions of a multi-phase, static mixture were obtained using a Multi-Layer Perceptron (MLP) architecture NN with one hidden layer consisting of 25 neurons. Resilient Back Propagation (RPROP) was the training algorithm used to achieve these results. All networks in this investigation were thus trained in MATLAB's neural network toolbox [22] using these specifications.

[5] It was mentioned in Chapter 1 that one of the limitations of the author's research was this inconvenient situation.

[6] No gravel bubbles were used as this would require manual (resistance and capacitance) data.

Figure 5-5: A photograph of the line electrodes used in the experiment. The line electrodes were realised by extending the point electrodes of Teague's rig with aluminium tape.

Figure 5-6: A photograph of the various polystyrene bubbles used in the experiment.

- In order to test the effect of the suggested change in electrode shape, it was decided to capture two comparable sets of training data - one using point electrodes and one using line electrodes. Because the ultimate goal is volume (as opposed to image) reconstruction, bubble positions are unimportant in training - only their size is significant. However, it was decided to create both training databases with exact matching bubble positions. That way errors introduced by one data set having more representative data than the other would be eliminated.

- Each time a certain volume fraction database was simulated, calibration data frames were captured before and after. This allowed factors such as temperature drift and impurities added to the water to be compensated for using the following formula:

$$V_{jk} = V_{jk} - \left(V_{1k} + \frac{j}{N} \left(V_{Nk} - V_{1k} \right) \right) \tag{5.1}$$

$$\begin{aligned} where\ j &= frame\ number \\ k &= sample\ number \\ N &= number\ of\ frames\ in\ database \end{aligned}$$

- Once both line and point databases had been populated, each database was used to train 10 independent MLP's - all with the same architecture. The reason for training 10 MLP's is due to the fact that no trained MLP will have identical weights (and hence the same generalisation error), because of the random nature of the initial weights.

- Average the results of each set of 10 MLP's and report results.

Results

The following table displays the results obtained when testing the trained networks on unseen test cases. The performance measure used is the mean absolute error between predicted volume fractions and desired ones.

Training Data	MLP1	MLP2	MLP3	MLP4	MLP5	MLP6	MLP7	MLP8	MLP9	MLP10	Average
Point electrode	3.16	4.74	2.37	4.34	12.01	1.36	5.61	1.82	6.18	5.15	4.66%
Line electrodes	2.04	1.91	3.20	8.03	7.22	9.66	8.59	8.03	5.00	11.33	6.58%

Table 5.1: A table showing the average generalisation errors of 20 MLPs - 10 trained with point electrode data and 10 with line electrode data. The average of each 10 is taken as the performance measure of the electrode type. Note that all entries in the table refer to the mean absolute errors in volume fraction prediction, and are reported as a percentage of the pipe's cross-sectional area.

Conclusions on using line electrodes in terms of NN reconstruction results

1. *No significant change in generalisation error of MLPs is noticeable between those trained using point electrode data, and those trained using line electrode data.* However, it should be noted that the data captured for the line electrodes was significantly more noisy than that of point electrodes. This was because Teague's rig was not optimised for the use of line electrodes, which result in higher currents being injected and received. This results in saturation occurring in some stages of the receiver chain, in turn resulting to higher ripple amplitudes on the demodulators' outputs. This ripple can easily be eliminated in the new rig, by ensuring no saturation occurs in the receiver amplification chain.

2. *Both line and point electrodes should be implemented on the new rig.* Because the results of this experiment were somewhat inconclusive due to the noise present in the line electrodes data set, a thorough investigation should be performed with robust electrodes using measurement hardware suited to the currents demanded by line electrodes. Once completed, more sound conclusions could be drawn.

   It was still strongly believed by the author that the inclusion of line electrodes would benefit the research, as definite improvements in FEM simulations of boundary measurements could be expected. FEM simulations are the heart of the conventional MNR algorithm and may be useful in training NNs, thus they are an important factor to consider when designing an electrode ring.

5.4.2 Electrode ring construction

- An electrode ring needed to be implemented conforming to the 57mm (inner-diameter), 5mm thick, PVC pipe used in the pumping loop at CPUT, where the system's dynamic tests were intended to be performed.

- The fabrication of the electrode ring was realised with the help of Mr. A. Sutherland[6]. It's construction was realised by skilled artisans in the workshop's of CPUT, while the machining of electrodes was performed by a commercial precision engineering company. The result: a geometrically precise electrode ring, capable of being conveniently fitted into a high velocity pumping loop.

- It was decided to implement 16 line electrodes, as well as 16 'guard electrodes' - positioned on either side (axially) of each line electrode. The reasons for the inclusion of these guard electrodes are twofold: firstly, the use of guard electrodes is known to reduce the effects of fringing – it has been shown [38] that the use of guard electrodes, while offering no special advantage in physiological EIT definitely does no harm. A diagram illustrating the principle of how guard electrodes minimise fringing, and how they are shaped and positioned in the electrode ring, appears in figure 5-7.

- The second reason for implementing guard electrodes on the new ring, is because the effects of using line electrodes in dynamic EIT reconstruction are not conclusive[7]. Should their use prove inferior to point electrodes, then *the originally intended guard electrodes (which are shaped to more closely resemble point electrodes) can be used for measurement.* Thus, by including these additional multi-purpose electrodes, one electrode ring can be used to more accurately investigate the consequences of using line electrodes (with and without guarding) as opposed to point electrodes. The inclusion of both types of electrodes in the ring has the obvious advantage of saving the time and money involved in fabricating separate electrode rings for each electrode type.

[6] Mr. Andrew Sutherland is a senior researcher at the Flow Process Research Centre at CPUT.

[7] Although tests were performed on Teague's old system to determine the effects of using line electrodes, as was mentioned in the previous section, the results were not conclusive as the hardware used in the investigation was ill-suited to the use of line electrodes.

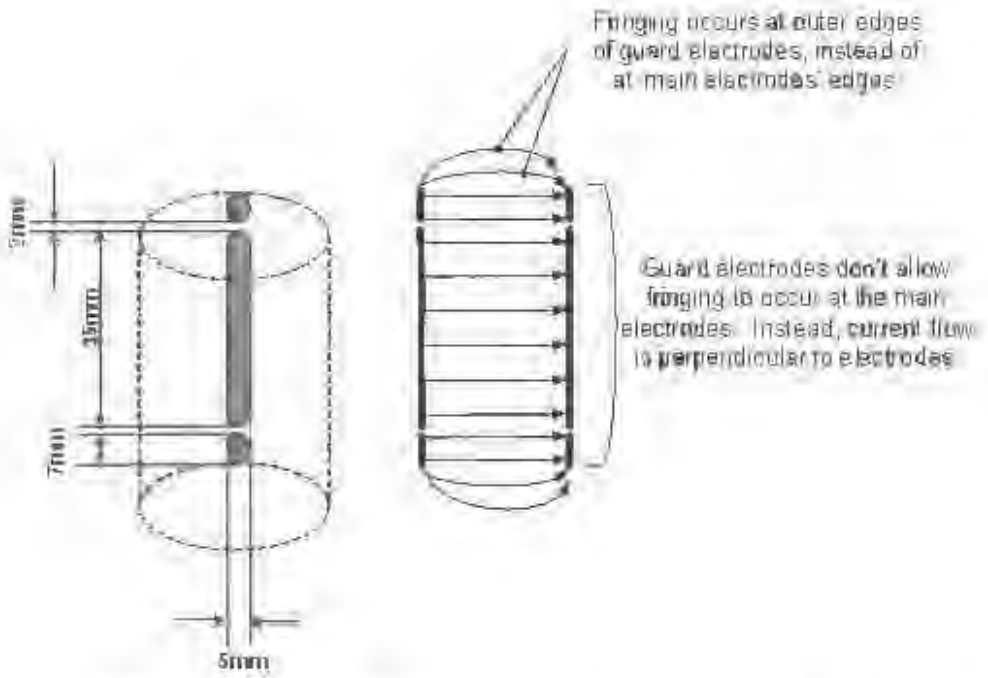


Figure 5-7: A diagram showing the dimensions of one of the 16 electrode sets, as well as how guard electrodes reduce fringing. In the sketch on the right, current field lines between two sets of electrodes on opposite sides of the pipe are drawn in 2-d. Notice that because the guard electrodes are so close to the main electrodes, and at the same potential, fringing is reduced at the edges of the main electrodes (which are used for measurement) - fringing still occurs, but mainly at the guard electrodes.

- The length of the line electrodes is an important design parameter, and should take into account such considerations as required frame rate and maximum expected flow velocity. The desired length of the electrodes were calculated using the specifications of the pipeline on which dynamic testing of the system was intended, as opposed to the specifications of the final application. These electrodes would have to be redesigned when testing on pipes with different characteristics. The specifications as provided by Mr. A Sutherland are listed below:

  - *Pipe inner diameter* $= 57mm$
  - *Pipe outer diameter* $= 63mm$
  - *Maximum flow rate* $= 5m.s^{-1}$

- Given a maximum flow rate of $5m.s^{-1}$ and frame rate v $frames.s^{-1}$, a tagged disturbance in the flow will travel $d = 5m.s^{-1} * v^{-1}$ $s.frame^{-1}$. In order for the disturbance to be within the uniform 2-d field (i.e. directly under the line electrode), the electrode length should be at least d *meters* long.

- Teague stated that "It can be shown that the frame-rate [v - of a dual plane EIT system intended for mass flow measurement] must meet the following requirement:

$$v = \left(\frac{1}{\frac{2L}{V_{MAX}} \frac{\Delta v}{V_{MAX}}} \right) \tag{5.3}$$

  where L is the separation between the two planes, V_{MAX} is the maximum expected velocity and $\frac{\Delta v}{V_{MAX}}$ is the desired velocity discrimination"[19]

- Teague also stated that an optimum plane separation[1] of three to four pipe diameters is required to minimise cross-talk between planes.

- Using values of 3 pipe diameters plane separation for L, 10% velocity discrimination for $\frac{\Delta v}{V_{MAX}}$ and $5m.s^{-1}$ for V_{MAX}, a required frame rate value of 188 $frames.s^{-1}$ is calculated.

[19] Teague took this from [39].

[1] Because the final application is a flow meter, 2 planes will be necessary for velocity calculation using cross-correlation techniques. Thus, plane separation is the distance between these planes.

If we take an over estimate of this value ($160\ frames.s^{-1}$), the required length of each electrode - rounded to the nearest 5mm - is **35mm**.

- The width of the electrodes will be 5mm, which is a compromise between thin electrodes required for receivers (to avoid averaging of equipotential contours) and broad electrodes required for transmitters (to create a uniform field). This decision was based on educated guesswork as well as the dimensions used by Dr. A Wilkinson and Mr. B. Randall who have successfully implemented a dynamic EIT rig at CPUT.

- Note that although cross correlation measurement of flow is not part of this stage of the project, factors such as plane separation and frame rate have been taken into account for extendability reasons.

5.5 Upgrading Giannopoulos' hardware

Although Giannopoulos' hardware was not perfect, it was decided that it would be used so as to save time. Redesigning and fabricating entirely new PCBs would extend the wait for real results even further. Giannopoulos had made provisions for extension from 8-electrode to 16-electrode demodulation boards, so all that was required was to populate the boards with components[12].

5.5.1 Transmitters

Giannopoulos used the same transmitter boards in [3] as Quinton Smit used for his capacitance tomography research [2]. These boards were designed for use on large pipes ($diameter > 200mm$) containing seawater. For the purposes of this research, fresh water will be used to eliminate corrosion, and a much smaller pipe diameter will be realised. Thus, the currents expected are far lower and the use of a regular opamp to drive the transmitter electrodes will be implemented.

[12] Giannopoulos made many undocumented alterations to the frequency generation boards and demodulation boards. It took a while for the author to determine what modifications were made and redraw the modified circuits. This was one of the difficulties mentioned earlier with regard to the handing over of the project, and is discussed further in appendix B.

Other reasons for changing the driving amplifier were: firstly the LM1875T used previously is obsolete and secondly, the bandwidth of this power opamp is around 70kHz. This is below the maximum frequency found by Giannopoulos' PBIL code, which seeks the optimum transmitter frequencies resulting in the least ripple present on the output of the synchronous detectors.

The chosen replacement opamp was the AD797 which offers ultra-low distortion, wide bandwidth and high current output (see Appendix D for data sheet). The redesigned circuit details are much the same as those used by Giannopoulos [3], except different gains are applied to obtain the desired current injection. The circuit details appear in figure 5-8 and are further detailed in Appendix A.

In terms of the extendability of these transmitters to higher current applications (e.g. large pipe, seawater), a search has been done for power opamps with larger bandwidth, however they are extremely costly and have significant delivery times. Thus, in an effort to accelerate the process as much as possible, the abovementioned standard opamp was used.

5.5.2 Receivers

The receiver circuit details remain similar to before except that, with the magnitude of received currents being far lower than in Giannopoulos larger rig, no buffer will be required in the feedback path of the transimpedance amplifier. Also, an improved amplifier was used (LF411 - see Appendix D for datasheet). A circuit diagram of one receiver amplifier circuit appears in figure 5-9; for more detailed circuit information refer to Appendix A.

5.5.3 A note on the implementation of the transmitter and receiver circuit boards

The transmitters and receivers were built by the author on VEROBOARD. This was due to their being prototypes. Also, the cost and time taken to develop entirely new PCB's was a further factor to consider.

An important difference to note about the new boards is that up to 6 channels are implemented on one section of VEROBOARD. This is in contrast to each channel on the boards designed by Smit [2] and used by Giannopoulos [3] being built on a separate board. The reason why each channel was previously on its own board, was to reduce the effects of stray

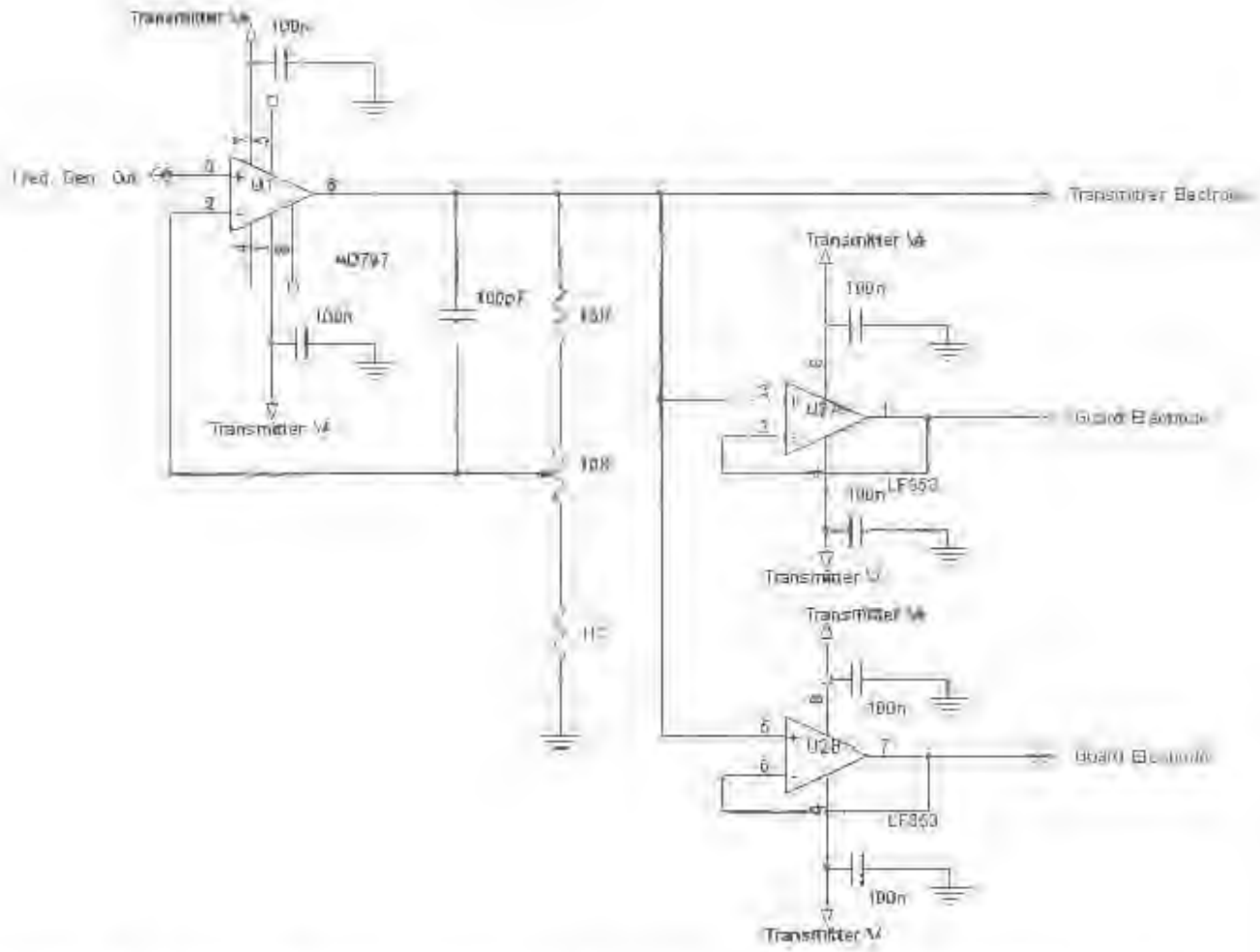


Figure 5-8: A circuit diagram of the transmitter amplifier. The signal generated on the frequency generation board is sent to the AD797, which is configured in the non-inverting mode. The amplified signal is then sent to the main electrode and buffered twice - each buffered signal can be used to drive a guard electrode if desired. Note the inclusion of a small capacitance in the AD797s feedback loop; this was done for stability as recommended in the data sheet.

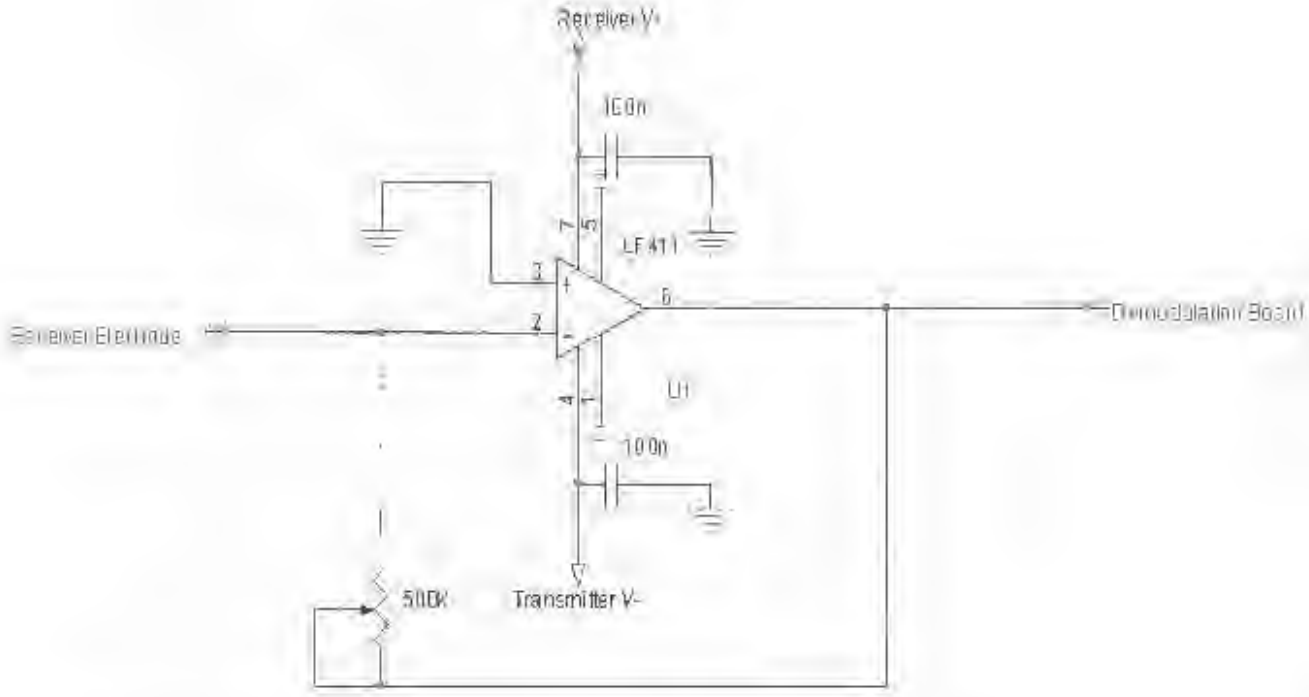


Figure 5-9: A circuit diagram of one receiver amplifier. The current out of the receiver electrode in question is amplified by the transimpedance to produce a voltage signal which is then fed to the demodulators.

capacitance which become significant when measuring capacitive data (which this project is not concerned with). If this project is to be extended to measure capacitive data, then new transmitter and receiver boards need to be realised on separate substrates. The actual circuit details can remain the same.

One more thing to note about the differences between the old and new transceiver boards is that previously, banks of decoupling capacitors were present for each driver and receiver. Because the driver and receiver amplifiers are now on a common substrate, only one bank of decoupling capacitors were used for all transmitters and another bank for all receivers. The voltage regulators[13] present on the previous transmitter boards were however, carried through for extendability reasons (i.e. to reduce the ripple at each driver should the medium become more conductive - e.g. seawater). Regulation of the receiver board's voltage rails was performed at the power supply board.

5.5.4 Frequency Generation

Each of Giannopoulos' frequency generation boards outputs 4 discrete frequencies - 2 transmitter frequencies and 2 90° phase shifted versions of these frequencies. The phase shifted versions are used only as reference signals in demodulation of capacitance data.

With the extension of the project to 10 electrodes, 4 more frequencies will be required to drive the extra transmitters. Originally it was decided to realise these frequencies using the existing boards. This could be done due to the fact that only resistance data is of interest at the moment. This means that new frequencies could theoretically be implemented using the micro-controllers' ports previously dedicated to the phase shifted signals. Once again, to allow for extendability, it was decided to fabricate totally new frequency generation boards, based on those designed previously.

Exact copies of Giannopoulos' frequency generation PCBs were made and populated[14]. The changes he made to the original boards, which were printed from the schematics appearing in [3] (and in Appendix A), were carried through to the new boards. An attempt to document

[13] Previously, and on the new boards, 2 voltage regulators were used per channel - one for the positive, and one for the negative rail.

[14] As mentioned, Giannopoulos made a number of undocumented changes to the boards. Such changes as cutting tracks and re-wiring signals made populating the boards difficult.

the differences between these schematics and how the boards were received by the author appear in Appendix B.

5.5.5 Demodulation boards

Due to the difficulties experienced in mirroring Giannopoulos' frequency generation boards, and the excessive design time required to design, test, print and populate entirely new boards, it was decided not to order new demodulation boards. Undocumented changes had been carried out on these boards too, and carrying this over to new boards which would have to be redesigned soon anyway was deemed wasteful.

Instead it was decided to populate the empty half of each demodulation board, and use what was originally designed as a demodulation board for a single receiver in a 16-electrode bimodal system as a demodulation board for 2 receivers in a 16-electrode unimodal (resistance data only) system. This is depicted in figure 5-10 and described in more detail in Appendix B.

The way in which this is accomplished is by using demodulation channels previously reserved for capacitance demodulation, as demodulators for another resistance channel. This is done by re-routing the source of the synchronous detectors' reference signals.

5.5.6 A note on the guarding of signal carrying cables

It was mentioned in section 5.5.3 that previously, transmitter and receiver boards were implemented on separate substrates, this meant that transmitter and receiver substrates could be placed close to their corresponding electrodes, thus minimising the effects of stray capacitance associated with signal carrying cables. Giannopoulos went to further lengths to minimise the effects of stray capacitance; by buffering the signals of all signal carrying conductors and applying this buffered signal to the conductor shields, 'bootstrapping' of the signal was achieved. 'Bootstrapping' by ensuring the potential difference between conductor and shield is always zero, reduces the capacitance of this cable to nearly zero – thus negating any voltage dividing effects stray capacitance otherwise would've caused. [40] provides another reason for driving the shields of signal carrying conductors in EIT; by reducing the stray capacitance of cables between the current source and electrode, current source performance is improved – this is because an ideal current source should have infinite output impedance which would be degraded

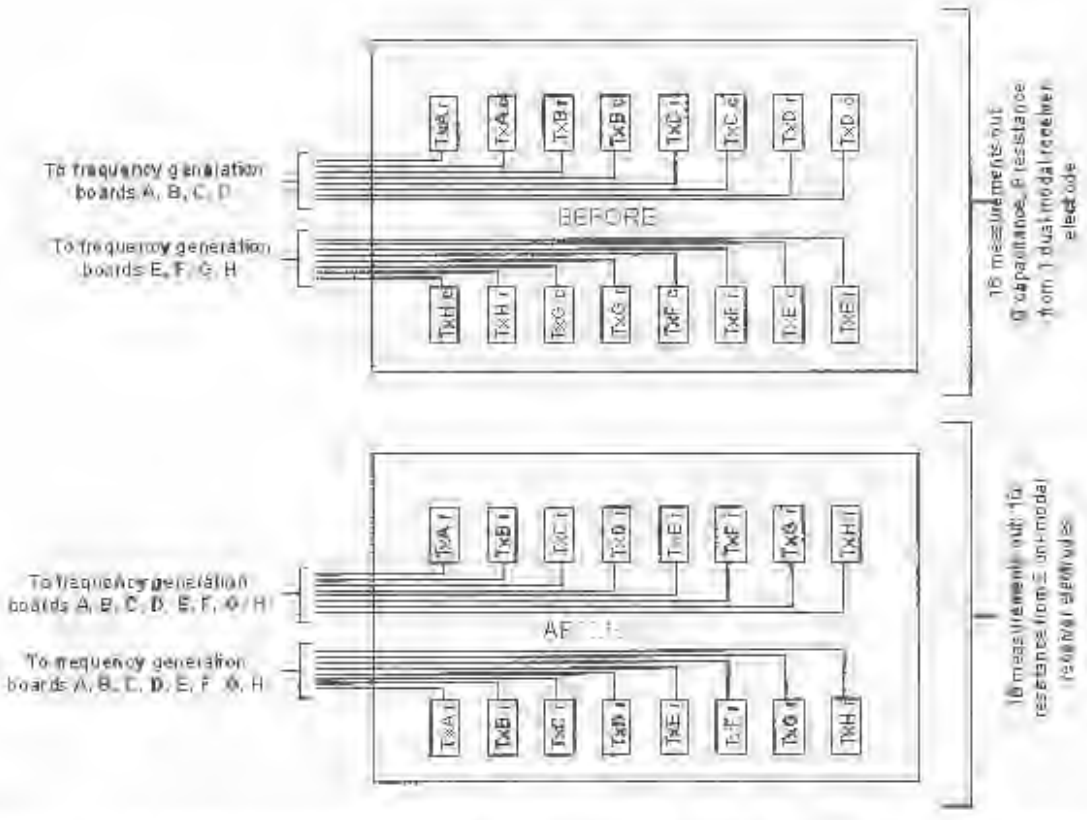


Figure 5-10: A block diagram illustrating the differences between Gamma-pensilos' demodulation boards and the new (altered) boards. The large blocks represent an actual PCB demodulation board as it exists in the system. The small blocks contained therein are synchronous detectors, demodulating the incoming signal with respect to their reference signal. Each synchronous detector is labelled so as to describe its task e.g. "TxC c" means that this synchronous detector is demodulating the capacitive component of the signal transmitted by transmitter C, and received by this demodulation board's receiver electrode. Note that the layout of the boards has not changed, merely the routing of synchronous detector's reference signals has

by any shunt capacitance seen by it.

Because the author's research is involved with resistance data and voltage generators are used as signal sources, no shielding of cables is necessary. Thus, where buffers intended for shield driving occur in Giannopoulos' hardware, they are bypassed - the shields are merely grounded. These concepts are discussed further in Appendix B.

Chapter 6

Preliminary Experiments

Once the upgrading of the hardware and fabrication of the electrode ring had been completed a number of preliminary experiments were conducted. The main objective of the tests was to determine the *static* performance characteristics of the new system. This was necessary in order to create a benchmark with which the system's dynamic performance could be compared. Furthermore, the system's performance would need to be compared with previous work by the group - all of which was done in static environments.

Another reason for performing static tests before moving to a dynamic environment, was to investigate the use of conventional image reconstruction software as developed by Herhold[6] for the group in 2004 [9]. The developed software was not optimised for the monitoring of fast changing processes and thus it was decided to perform static tests and draw conclusions from there.

In what follows, the results of experimenting with Herhold's software are presented first, together with the justification for their investigation and modifications required for the system at hand. Following that, the results of using Multi-Layer Perceptrons (as used by Teague [1, 4], Giannopoulos [3] and Capindissa [7]) and Kernel Ridge Regression to predict volume fractions of static EIT data are documented. The methods used in these experiments are applied in the remaining chapters of this document – they are therefore explained in detail here, so as to prepare the reader for later investigations. Finally, to close the chapter, conclusions are drawn about the tests performed.

Note that throughout this chapter and the remainder of the document, various MATLAB

script files and functions - written by the author - were used to achieve certain tasks (e.g. data normalisation, training database collection, etc.). All significant code appears in Appendix C where its workings are explained through thorough commenting. Where appropriate certain code fragments will be referenced should the reader wish to gain a further understanding of the techniques applied.

6.1 Experimenting with conventional image reconstruction techniques

For the reasons outlined in section 2.6, it is important that tried and tested image reconstruction techniques be investigated by the research group. The modified Newton-Raphson (MNR) algorithm is the most widely used reconstruction technique for such tasks. In fact, free, downloadable and optimised software is widely available[1] for such purposes and should be made use of by the research group in future. Unfortunately, the data collection method used by the research group at the moment (i.e. voltage excitation and current measurement) is hardly used in the impedance tomography community. This is due to the significance of the errors introduced by unknown contact impedances (between electrode and electrolyte) when evaluating the forward problem - required to be solved in each iteration of the MNR algorithm. Hence, no suitable software which could be applied to the present system was found.

The research of Siegfried Herboldt [9] was the first in the Dan Tech impedance tomography group to implement the MNR algorithm. Although his application was for imaging a settling process using TDM, the principles are easily applied to a circular geometry using FDM. Herboldt's data collection method had also used a voltage driven strategy. Thus, it was decided to modify his code, unoptimised as it was, to deal with the problem at hand.

6.1.1 Forward problem solver: Finite Element Method

In order to use Herboldt's code to perform the forward problem on the new rig, it was necessary to make a few alterations. A new, more complex meshing program was required to create a

[1] An example of such software is available from the EIDORS website [28], which is a dedicated site concerned with the research of tomography reconstruction algorithms. Also, D. J. Wilkinson of UCT has developed a high speed, high resolution imaging system for EIT.

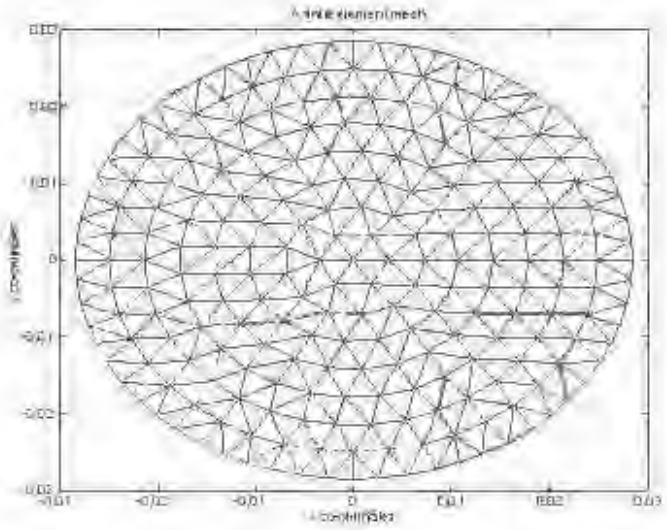


Figure 6-1. An example of a mesh created using the new meshing algorithm. The mesh is specifically structured to conform to the size of the electrodes on the new rig (i.e. 5mm wide)

circular mesh structure which conformed to the exact electrode spacing of the new electrode ring. Also, matrix assembly had to be altered so as to reflect the FDM measured data capture

Meshing algorithm

Due the differences in the geometry between Herhold's rig (rectangular) and the new dynamic rig, a new meshing algorithm would have to be implemented from scratch. This was done in MATLAB and conformed to the conventions used by Herhold (see Appendix C: "mesh16.m"). The method used was based on that documented by Dong in [25] - work done for the ERT research group of Dr. A Wilkinson at UCT. In brief, the principle of how the mesh is created is by dividing the circular region into a specified number of concentric rings. Next, starting at the centre and moving outwards, each ring is divided into triangles - as equilateral as possible. Thus, the lengths of the element sides is roughly equal to the ring separation. The reason for the 'equilateralness' of elements is to maintain a good 'quality factor' throughout the mesh (ie elements should have no angles close to 180° [24, 25]. An example of a mesh created using this algorithm appears in figure 6-1.

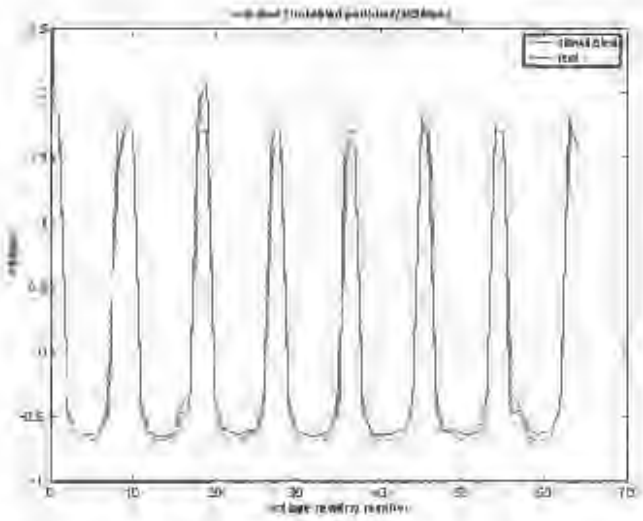


Figure 6-2. A plot of the measured and calculated values. The x-axis represents the transmitter-receiver combination number (i.e. TxA/RxA=1, TxA/RxB=2, ..., TxH/RxH=64), the y-axis is proportional to the resistance between the pair. As with Herhold's experiments, the calculated periphery voltages are very similar to the measured ones.

Results

The results of using Herhold's modified code (see Appendix C: "femm", "plotprop.m", "summodes.m") to evaluate the forward problem look promising. A plot of these calculated results appears in figure 6-2. On the same set of axes is the plot of data measured from the physical rig. Both the calculated and measured values were those resulting from a homogenous conductivity distribution. Note that the values have been normalised (i.e. subtract the mean and divide by the standard deviation of the 64 values).

6.1.2 Inverse problem solver: MNR algorithm

Herhold successfully applied the MNR algorithm to his problem. Results however were not optimal. The resolution of his system was limited to 10 levels, which was deemed adequate for the problem of imaging a settling process. The reason for this limited resolution was due to the measurement strategy employed - very similar to the strategy applied to this project. A figure extracted from Herhold's M.Sc. thesis appears in figure 6-3: it shows the results he

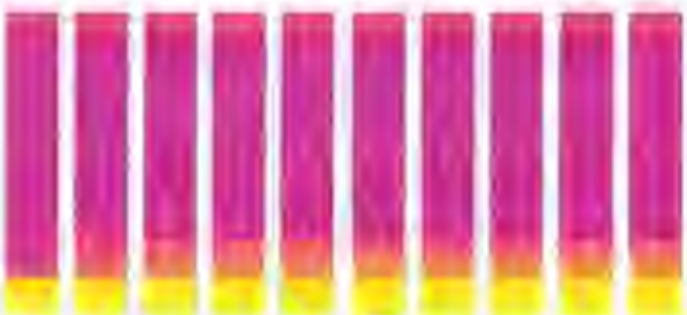


Figure 6-3: A figure extracted from Herholdt's M.Sc. thesis [9]. Note that only 16 levels of resolution were obtained - this was deemed adequate for the task at hand.

obtained using the MNR algorithm to reconstruct the interior conductivity distribution of a settling tank.

Tests performed using Herholdt's MNR algorithm and the new electrode ring revealed that similar restrictions in resolution are present in the new rig, clearly, 16 levels of resolution (*i.e.* pixels) is severely inadequate for the task of predicting volume fractions of as little as 1% of the vessel's cross-sectional area, and the results are not shown here. As mentioned above it is strongly believed that this restriction in resolution is due to the measurement strategy used, which allows unknown contact impedances to significantly degrade the signal to noise ratio of the measurements obtained.

6.1.3 Investigation of kernel methods

Before an in-depth investigation into the use of kernel methods for EIT could be initiated, it was necessary to determine whether their performance was indeed comparable to the methods used previously in the research (classical NN architectures). Luckily, owing to the manner in which kernel methods generalise to unseen data (by summing superimposed kernel functions), it is relatively easy to ascertain whether their techniques hold potential for the classification of a specific data set. Generally speaking, if a kernel has such potential, then when testing for generalisation ability on a given data set (by scanning over various values of kernel width - σ),

Figure 6-5: A photograph of the 'bubbles' used in training. Polyethylene rods were shaped into bubble shaped masses and mounted on dowel-sticks for ease of positioning. Note that there are ½ of each bubble diameter so as to introduce a wider variety of volume fractions when combined with other bubbles – thus providing a wider variation in target values used in training.

critical and indicative parameter.

Methods

The methodologies presented in this section are important because they are used in later chapters in the investigation of dynamic situations. Most of the methods used in the remainder of this document are presented, justified and explained here - it will be assumed in later chapters that these concepts are understood.

Training database construction A variety of bubble sizes were created, modelled in a similar fashion to Teague's - cylindrical shapes as long as the main electrodes (35mm) with rounded ends. They were manufactured from poly-ethylene, which has a similar conductivity to air, and attached to thin dowel-sticks for easy positioning in the measurement plane. A photograph of these 'bubbles' appears in figure 6-5. Each bubble or combination of bubbles represents a specific target value.

For each target value, 1000 frames were captured while each bubble combination was '*moved*'

around the imaging plane so that the whole plane was filled multiple times. The electrode configuration used was that which was expected, but not yet proven, to produce the best results. Note that bubbles were allowed to come into contact with electrodes (unlike Tengku's training database collection techniques where only bubbles near the centre were considered), which was done so as to simulate a more realistic dynamic flow situation. On-the-fly averaging was performed so that each captured frame was the result of averaging 3 actual frames of data. This helps increase the signal to noise ratio of the data, improving NN generalisation ability[9].

After data normalisation, the measurements captured were stored in a database, with target values appended to each set of measurements. Target values were obtained by calculating what volume fraction each bubble combination created - an easy task since the inner diameter of the electrode ring and diameter of bubbles was known *a priori*. The number of bubble combinations used resulted in 14 different volume fractions (and hence target values) being present, thus producing an available training database of 14 000 instances.

A note on the 'stirring' training method Note that the method used here for filling the training database with measurements obtained from various bubble positions is somewhat different to that used throughout the project's history. It is certainly a lot simpler as well as thousands of times faster. The difference is owing to the fact that bubble positions are no longer an important part of the target data; only the percentage of the vessel's cross-sectional area which the bubbles occupy (volume fraction) is important. Thus it is possible to continuously capture data at high frame rates, while 'stirring' each bubble combination around the measurement space. All frames captured for each bubble combination will have the same target values. This is in contrast to the different, spatially concerned target values associated with each training instance used previously.

A note on data normalisation "Usually, in classification and regression it is advisable to normalise X [the training data matrix] (and y if values are real). Normalisation, here, means making the mean of each column [feature] 0 and the standard deviation 1" [23]. Indeed, all the classification and regression problems the author has encountered have undertaken some form of

[9] This method of averaging 3 frames of data follows suit of Tengku's training database collection techniques.

data normalisation, the reconstruction techniques developed by Teague (and used in the project since then) employed data normalisation - the process used is documented in [1]. It should be noted that data normalisation is essential when implementing kernel methods, where kernels are functions of the euclidean distance between data points. In this case, normalisation ensures that certain features of the input data are not allowed to dominate system performance. In terms of Multi-Layer Perceptrons, normalisation improves the numerical conditioning of the optimisation task involved in training [4, 23]. A further, hardware related, advantage achieved through normalisation is that the effects of hardware measurement drift are eliminate by removing each feature's de-bias (mean). See Appendix C: "normal_svdcm" for the normalisation code.

Volume fraction reconstruction (MLP's vs. KRR) In both of the following cases (MLPs and KRR) the exact same data was used for training and testing, and because the number of training instances was large (3000 training instances), the training database is seen to be a true representation of ECT data - the same is true of the test data (1000 instances). When training MLP's a further set of 1000 instances were used as validation data, so as to implement 'early stopping'[4] as used by Teague.

Imperative to note is that whenever generalisation error is reported for a given prediction method, the test data used to calculate this measure took absolutely no part in training. Therefore the comparison of the results produced in each case should be a reliable indication of performance. In order for the results obtained to be comparable to those of Teague and the researchers who followed him, generalisation error is calculated as the mean absolute error between a test data set's desired target values and those predicted by the machine.

MLP NNs

- Firstly, 10 MLP (Multi-Layer Perceptron) NNs were trained using MATLAB's Neural Network Toolbox [22]. The exact same NN training methods and architecture as those used by Teague in his best performing volume fraction predictor were used, namely:

[4] 'Early stopping' is the process of stopping the adjustment of MLP weights during the training process, after the generalisation ability of the MLP starts to deteriorate on a separate data set - called the validation data set. The technique ensures that the MLP does not overfit the specific training data split. For more in depth discussion of its implementation see [23].

Resilient Back-Propogation training algorithm with early stopping, and a single hidden layer of 25 neurons per MLP.

- The results of using these 10 MLPs to predict the volume fractions of the test data set would then be averaged[2], and the result reported. This would serve as a comparison to previous research, allowing certain conclusions to be drawn. Furthermore, a benchmark would be created with which the system's dynamic performance, and the results of using KRR (Kernel Ridge Regression) on the data, could be compared.

KRR

- Next, the use of KRR was investigated - the process of investigation involved performing a grid search for the optimal kernel width (σ) and regularisation parameter (γ) - see Appendix C: "econsg10fold.m" 23. This process was done by:

  1. Selecting the available training data, and using tenfold cross-validation (discussed in the following section) to determine the generalisation error for each σ, γ combination.

  2. Using the value for γ which resulted in lowest error, a refined search for σ is performed between the values $0.1 \times best\ \sigma$ to $10 \times best\ \sigma$ - where $best\ \sigma$ is the σ-value which resulted in least error. The reason for performing a refined search for σ but not γ, is due to the fact that γ is fairly insensitive above a certain value which would have been surpassed in the grid search.

- The best σ, γ combination was then applied to training data (see Appendix C: "krr.m") and predictions made on the test data - these results are then reported.

A note on tenfold cross-validation When seeking the optimum σ and γ tuning parameters for a kernel machine, one needs to perform multiple runs of training and testing to see what the generalisation errors resulting from various combinations of these parameters are. If for each combination of σ and γ, the same data split (between training and test data) is used, then

[2] The reason for averaging the results of 10 MLPs is that all of them were trained starting from a random weight initialisation. Thus no MLPs weights will end up exactly the same.

these parameters would be specifically tuned for this particular data split. Thus, these values would not be a reliable indication of the optimum values relating to the data set as a whole. The solution is to perform multiple data splits for each combination of tuning parameters. The generalisation error for each combination is then taken as the average error for each data split. This ensures that the parameters have not been tuned around a specific split in the data.

Tenfold cross-validation is the most common validation procedure the author has experienced and observed in the literature. More generally speaking, it is sometimes called v-fold cross-validation – v referring to the number of data splits performed for each run. A diagram illustrating the process of tenfold cross-validation appears in figure 6-6 - the code for performing the procedure appears in Appendix C: "tenfoldkrr.m" [23].

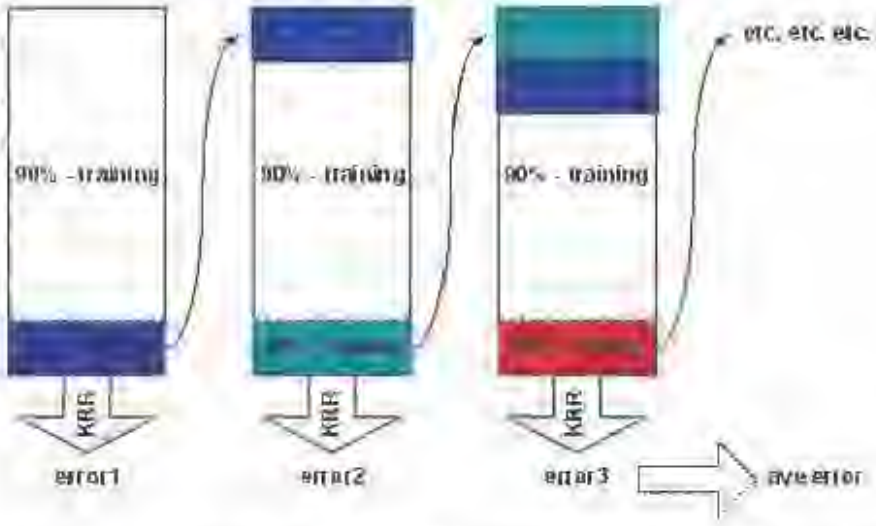


Figure 6-6: A diagram illustrating the process of tenfold cross-validation. First the data is split into 90% training, 10% testing – KRR is performed and the error recorded. Next, the previous testing data is affixed to the top of the data matrix while the rest of the data is shifted down. Another data split is made and KRR performed again. This process is repeated until all data has been used for testing and training (10 times) and the results averaged and reported.

Results

This section presents a series of graphs and tables which show the results obtained when performing tests in the manner described above. First, the results of training MLPs are presented

in figure 6.7 and table 6.1. These results will form the benchmark performance measure for the rest of this thesis since they were obtained using the same methods as used by Teague [1, 4] and Giannopoulos [3].

Next, the results of performing a grid search for the optimal σ, γ combination values used in KRR is shown in figure 6-8 by a surface plot of the mean error resulting from tenfold cross-validation. Following that, a summary of a further refined σ search and best σ and γ values appear in figure 6-9, together with the desired vs. predicted volume fractions of KRR using the optimum σ and γ. Note that the way in which the results are presented and depicted in this section will constitute the standard used in later chapters in order to produce comparable outcomes.

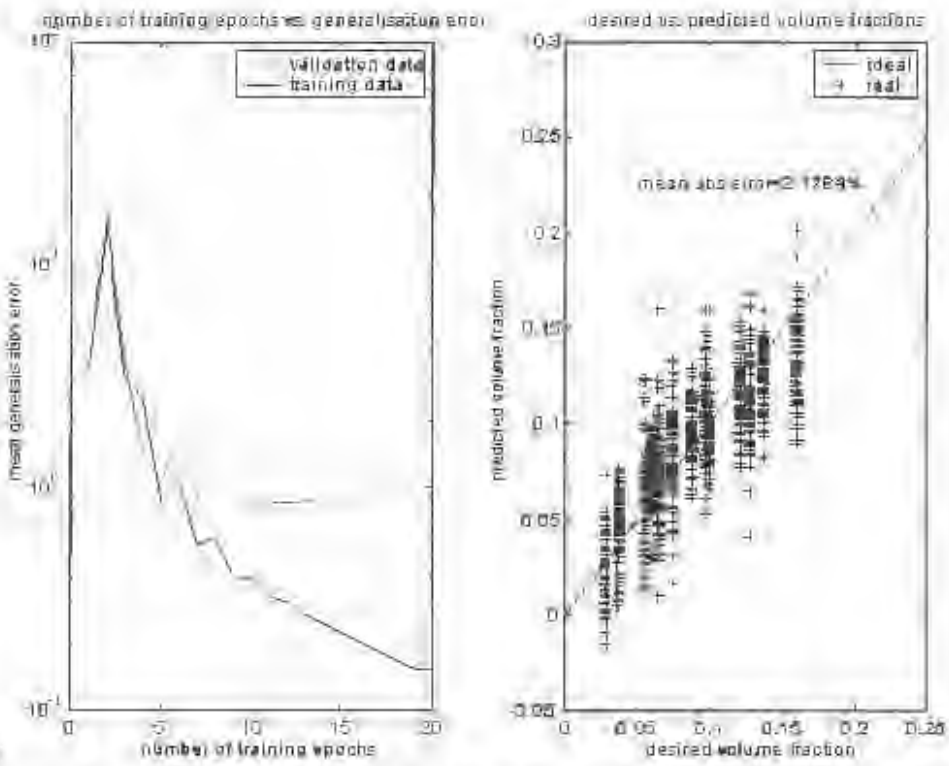


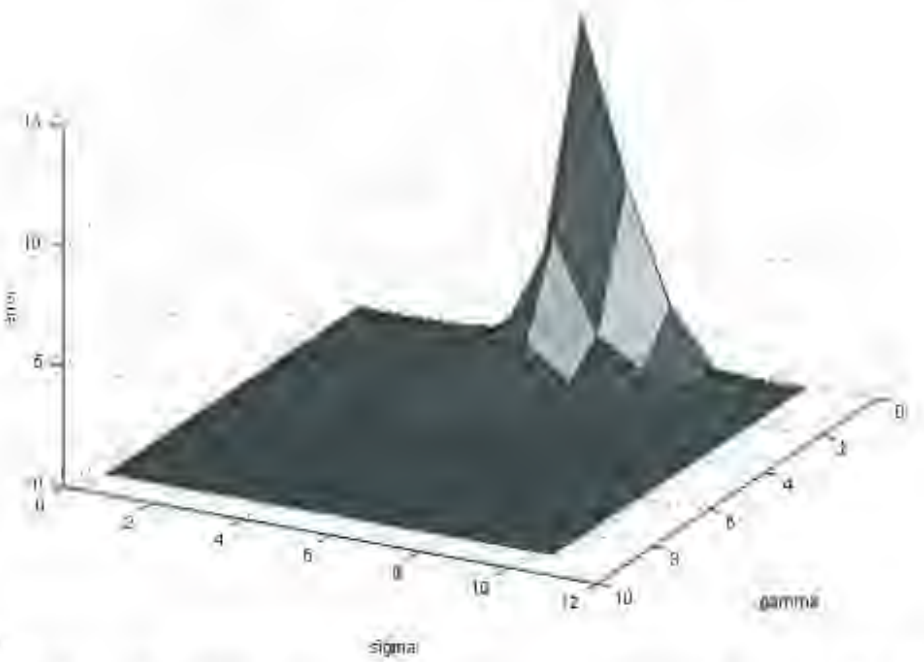
Figure 6-7: A figure showing 2 subplots; on the left, a graph showing the generalisation error for each epoch of weight adjustment during the resilient Back-Propagation training algorithm. The darker line shows training dataset error, the lighter line, validation dataset error of one of the 10 MLPs trained. The subplot on the right shows a plot of the desired *vs.* predicted volume fractions of an average performer of the 10 MLPs trained. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error of all 10 MLPs appears written (as a percentage of the pipe's cross-sectional area) on the plot too.

MLP number	MLP 0	MLP 1	MLP 2	MLP 3	MLP 4	MLP 5	MLP 6	MLP 7	MLP 8	MLP 9	Mean generalisation error
Mean absolute error	1.98	1.99	2.02	3.91	1.86	2.28	2.08	1.93	1.79	1.93	2.189%

Table 6.1: A table displaying the results of training 10 separate MLPs. The error values which appear as entries in the table are defined as the mean absolute error between the desired and predicted volume fractions (as a percentage of the pipe's cross-sectional area). The average of these 10 error values forms the benchmark performance measure.

Fig 6.8

Error surface of a grid search for optimum sigma and gamma values for the Quantum Force Dataset



sig6.pdf

Figure 6-8: A surface plot of the mean generalisation error obtained using tenfold cross-validation and KRR for each σ, γ combination. The x-y plane represents σ and γ, the z-axis represents error. Note that the characteristic dip in error around a certain σ, γ combination is not visible in this surface plot. This is because the dip is relatively small with respect to the huge peak in error occurring in this figure. The dip bleuds into its surrounding plateau and is highlighted in later plots.

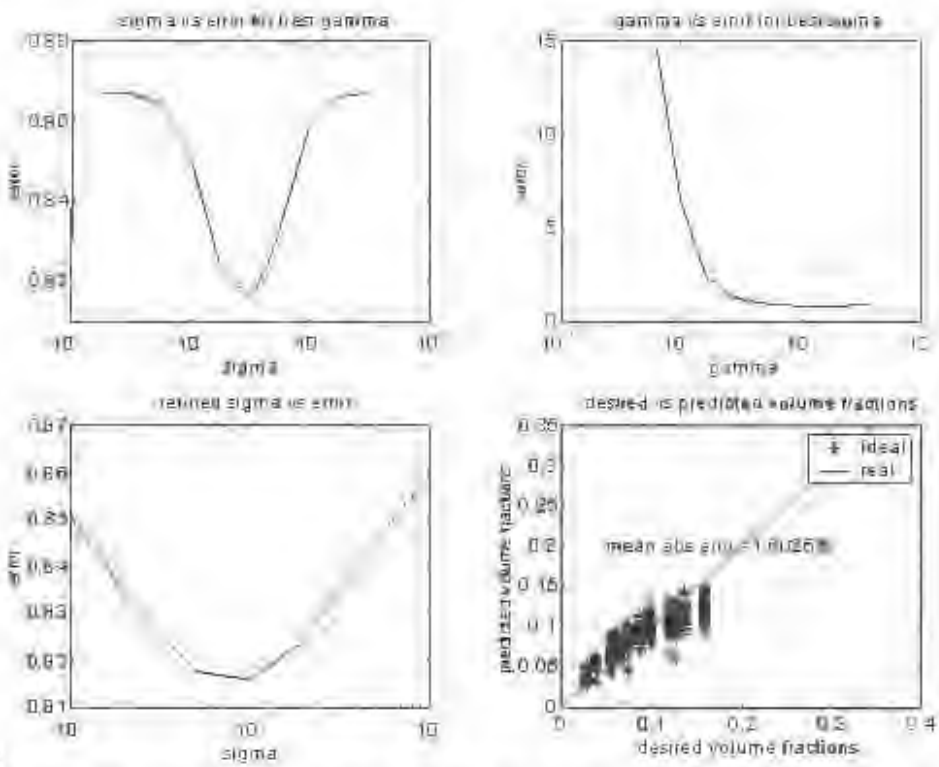


Figure 6-9: A figure showing four subplots; the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by $+$s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipeline cross-sectional area) on the graph too.

Summary, discussion and conclusions based on preliminary experiments using conventional image reconstruction techniques

- It has been emphasised in this text that there is a need for the research group to have the ability to construct images using conventional, optimised reconstruction techniques. Even if we are developing other methods with other applications than image reconstruction it is important that we have the ability to reconstruct images.
- Herbuldt's FEM code has been altered and promising results have been obtained. Although these results cannot be used to produce high resolution images, it goes a long way to confirming that the measurement hardware is operational and that the assumptions made when implementing the FDM measurement strategy were sound.
- The present voltage driven measurement strategy used, does not allow for high resolution images to be obtained using the MNR algorithm. This is due to the errors introduced by unknown contact impedances and the limited number of independent measurements available from the rig - thus a new measurement strategy should be investigated[5].
- Although the FEM code developed is of little use in image reconstruction, it may be useful in NN training, where an intelligent system (e.g. NNs or KRR) might be capable of extracting information regardless of the errors between the simulated periphery voltages and those measured from the vessel.

Summary, discussion and conclusions based on preliminary experiments using computational experiments

- MLPs have been trained using the best architecture and techniques as reported by Teague [1, 4]. Their overall volume fraction prediction error, for a static, 2-phase mixture using 16 conductance electrodes and FDM, is **2.18%**. This, compared to Teague's reported **1.15%**. The slight degradation of performance is expected since fewer measurements are available from the present system than from Teague's setup - where both resistance and capacitance data was measured.

[5] A new strategy using current excitation and voltage measurement has been proposed by Prof. J. Tapson. This would greatly improve image resolution.

- A grid search for the kernel width (σ) and regularisation parameter (γ) used in KRR have been performed on the data. The results look promising since the characteristic shape of a well suited kernel has been achieved for σ vs. generalisation error. The regularisation parameter is found to be uncritical above a certain value, which is also characteristic in such problems.

- KRR has been performed on the same test data as used to calculate the MLP error above. The prediction error for a static, 2-phase mixture using 16 conductance electrodes and FDM, is **1.60%**.

- KRR is thus deemed a valuable volume fraction prediction technique, and further research should be performed on its use in ECT. Specifically, the effects of performing a singular value decomposition on the training and test data should be investigated, as this is known to improve performance of such methods.

Chapter 7

Experimental Results and Discussion

This chapter is divided into 2 parts; the first part is involves static tests while the second part involves dynamic tests. The reason for further investigating the system's static performance is to determine whether the extension of point electrodes to line electrodes does in fact, improve system performance. A similar question concerning the use of guard electrodes needs to be answered; the reason why these experiments need to be done in a static environment, is that we have no way of absolutely determining instantaneous volume fractions of real flows. This makes it extremely difficult to evaluate the system's dynamic performance on a frame-by-frame basis.

In what follows, a brief description of the procedures adopted in each experiment is given, followed by the presentation of results. Most of the methodologies used here, have been detailed in the preceding chapter; where unexampled techniques are applied, they are discussed. Following the presentation of results, a summary and discussion of them is put forward.

7.1 Static situation results for a 16-electrode unimodal FDM EIT system

It was mentioned in section 5.4.2 that the guard electrodes implemented on the new electrode ring , which more closely represent point electrodes in shape and size, would be useful in measuring the performance difference between NNs using point electrodes and those using line electrodes. Although a similar experiment had been performed using Teague's dormant rig

(documented in section 5.4.1), conditions were not conducive to drawing meaningful conclusions (e.g. saturation in the receiver chain was present due to increased currents being received by line electrodes). Thus, the opportunity to more accurately determine this performance difference was utilised.

Moreover, the effects of using guard electrodes with line electrodes needed to be investigated. The results presented in this section are therefore based on 3 sets of measurement data.

1. Point electrodes (where one ring of guard electrodes are used as point electrodes for measurement – the remaining electrodes are disconnected).
2. Line electrodes (where only the main line electrodes are used for measurement).
3. Line & guard electrodes (Where the main electrodes are used for measurement while guard electrodes are active).

7.1.1 Method

The 3 sets of training data mentioned above were collected using the techniques described in the previous chapter (ie. using the 'stirring' method to capture 1000 frames of data for each target value). After normalisation and random shuffling[1] of training instances, tests were performed to determine the performance of MLPs and KRR in predicting static volume fractions. For MLPs, the methods used were identical to those of the previous chapter. Likewise for KRR except that, based on the conclusion of the last chapter that "the effects of performing a singular value decomposition on the training and test data should be investigated", a singular value decomposition of the data was performed and the results compared to those obtained from the raw data.

7.1.2 A note on Singular Value Decomposition

"In statistical pattern recognition and computational intelligence we often work with square symmetrical, non-negative matrices – for example the covariance matrix $\mathbf{X}^T\mathbf{X}$, the squared

[1] The random shuffling of training instances is performed so that when a data split is made, an equal distribution of the various target values is used in training.

measure matrix, the kernel matrix $\mathbf{K}$ etc.. All are compact summaries of the way the data is distributed in the vector space.

"Such matrices can always be decomposed or factorised as $\mathbf{M} = \mathbf{P} \times \mathbf{D} \times \mathbf{P}^{\mathbf{T}}$ where $\mathbf{P}$ is a matrix in which each column is an eigenvector and $\mathbf{D}$ is a diagonal matrix of associated (non-negative) eigenvalues. The eigenvectors are all mutually orthogonal" [23].

Because the columns of a matrix can be arbitrarily re-ordered, we may re-order the columns of $\mathbf{D}$ (and $\mathbf{P}$) such that the eigenvalues appear in decreasing order from left to right (this is done automatically in a singular value decomposition which is based on eigen decomposition). If some of the eigenvalues are zero, then the original matrix is of less than full rank (i.e. linear dependence between columns of original matrix). Eigenvalues close to zero indicate near linear dependence or ill-conditioning and that the data does not fully occupy the vector space, "but lives in a subspace or manifold of the vector space. Omitting the eigenvectors with near-zero associated eigenvalues projects the data into this subspace, often eliminating 'noise' and improving the performance of subsequent pattern classification processes" [23].

If our data matrix ($\mathbf{X}_{(n,m)}$) is non-square, as in EIT, then a 'Singular Value Decomposition' (SVD) can be performed where $\mathbf{X} = \mathbf{U} \times \mathbf{S} \times \mathbf{V}^{\mathbf{T}}$. In terms of our eigen decomposition of $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ these matrices have the following meanings:

$$\mathbf{P} = \mathbf{V} \tag{7.1}$$

$$\mathbf{D} = \frac{\mathbf{S}^2}{(n-1)} \tag{7.2}$$

We can then proceed to zero all singular values (diagonal elements of $\mathbf{S}$) which are below a certain threshold - leaving k columns in $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$. The resulting matrix $\mathbf{X}_{SVD} = \mathbf{U}_k \times \mathbf{S}_k \times \mathbf{V}_k^T$ "is the best rank-k least-squared error approximation to $\mathbf{X}$" [23], and is often a more useful representation of the data in many classification/regression problems.

7.1.3 Results for the 'point electrodes' data set

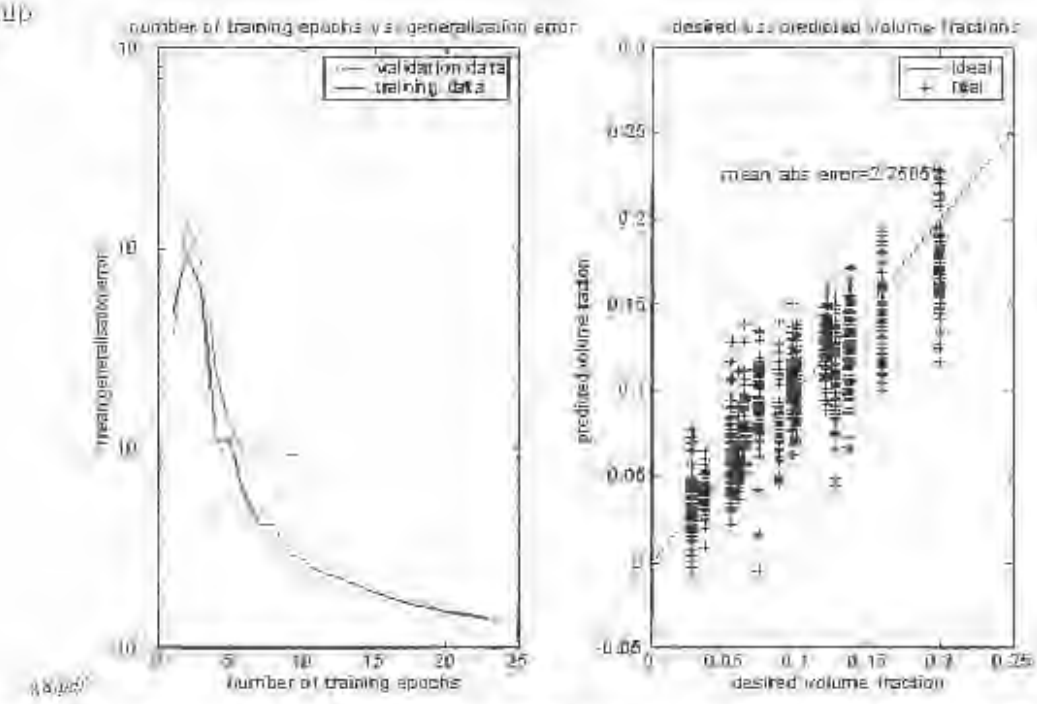


Figure 7.1: The results of using MLPs to make volume fraction predictions on the **point electrodes** dataset. The figure shows two subplots: on the left, a graph showing the generalisation error for each epoch of weight adjustment during the resilient back-propagation training algorithm. The blue line shows training dataset error, the green line validation dataset error of one of the 10 MLPs trained. The subplot on the right shows a plot of the desired vs. predicted volume fractions of an average performer of the 10 MLPs. Predictions made on the test dataset are represented by $+$'s, while the line shows the ideal zero error response (i.e. the line $y = x$). The mean absolute error of all 10 MLPs appears written (as a percentage of the pipe's cross-sectional area) on the plot too.

MLP number	MLP1	MLP2	MLP3	MLP4	MLP5	MLP6	MLP7	MLP8	MLP9	MLP10	Mean overall error
Mean absolute error	2.02	5.01	4.53	1.86	4.25	1.80	3.10	1.72	2.28	1.85	2.75%

Table 7.1: A table displaying the results of training 10 separate MLPs using the point electrodes dataset. Each entry in the table represents the mean absolute error between the real and predicted volume fractions of a test dataset. The average of these values appears in the last column.

point krr norm

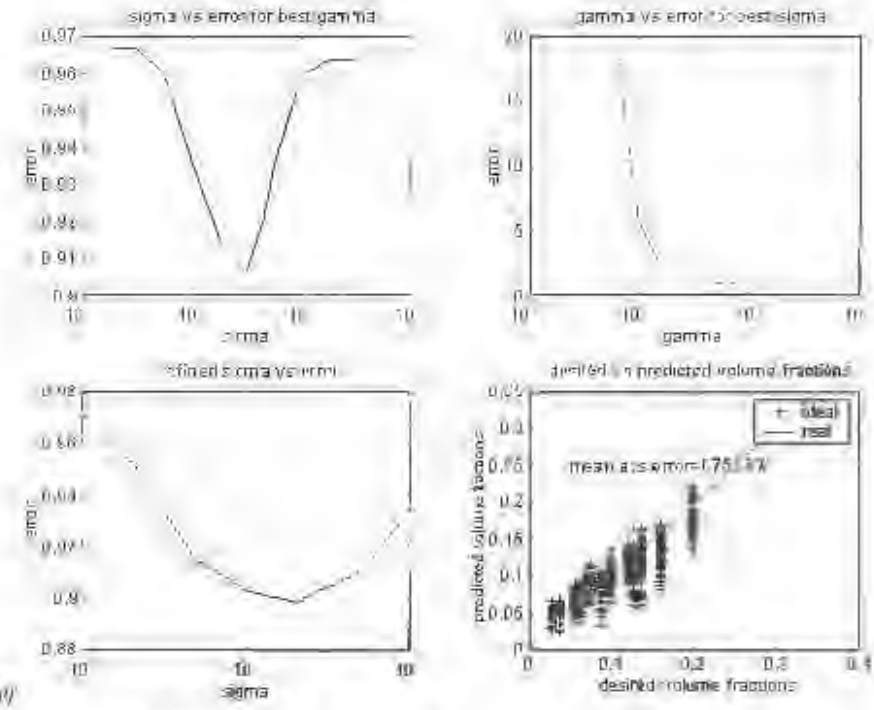


Figure 7-2: The results of using **KRR** to make volume fraction predictions based on the **point electrodes** dataset. The figure shows four subplots; the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by $+$'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's cross-sectional area) on the graph too.

point krr svd

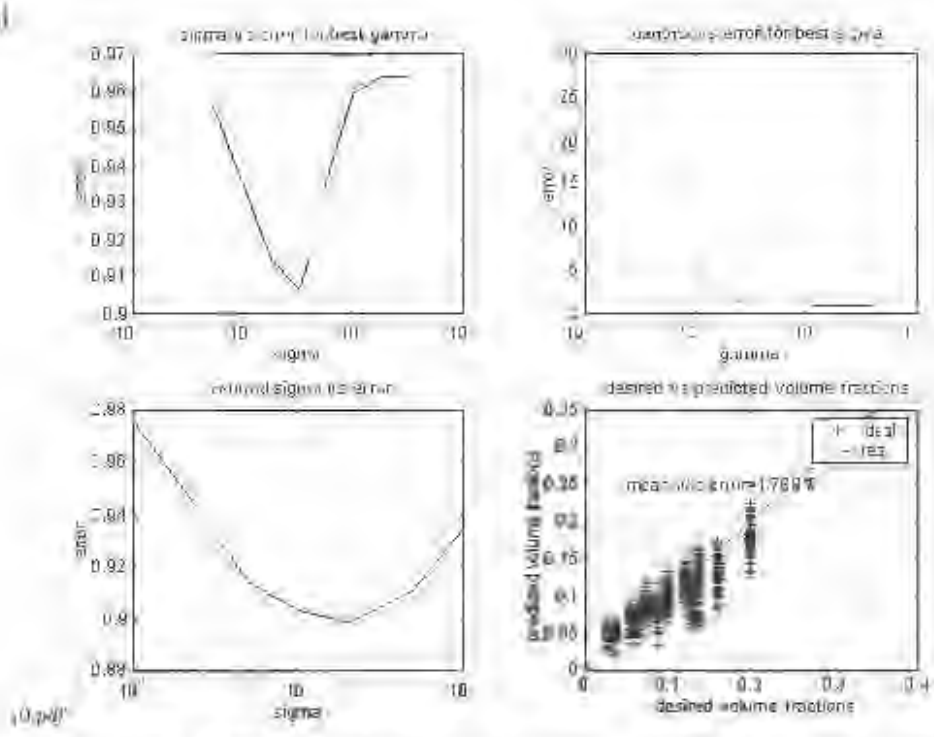


Figure 7-3: The results of using **KRR** to make volume fraction predictions based on the **point electrodes** dataset. A singular value decomposition (**SVD**) was performed on the data - zeroing any singular values contributing less than 1% of the sum of all singular values. The figure shows four subplots; the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by $+$'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's cross-sectional area) on the graph too.

7.1.4 Results for line electrodes

line.mlp

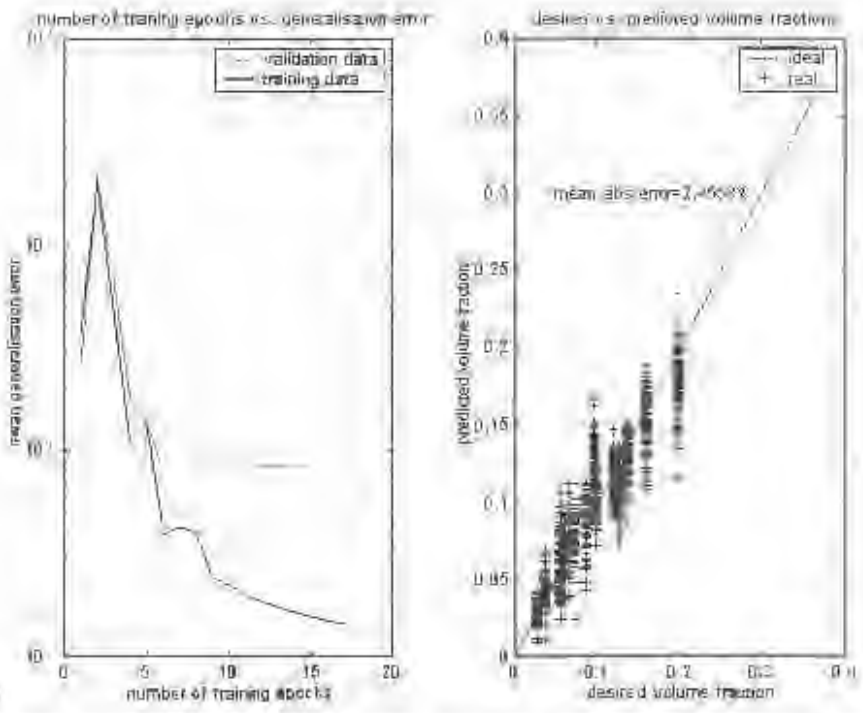


Figure 7-4: The results of training MLPs to make volume fraction predictions on the **line electrodes** dataset. The figure shows two subplots: on the left, a graph showing the generalisation error for each epoch of weight adjustment during the resilient back-propagation training algorithm. The blue line shows training dataset error, the green line, validation dataset error of one of the 10 MLPs trained. The subplot on the right shows a plot of the desired vs. predicted volume fractions of an average performer of the 10 MLPs. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error of all 10 MLPs appears written (as a percentage of the pipe's cross-sectional area) on the plot too.

MLP number	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Mean absolute error	3.41	4.37	1.02	2.01	3.76	1.08	1.41	2.40	2.25	2.46%

Table 7.2: A table displaying the results of training 10 separate MLPs on the line electrodes dataset. The error is defined as the mean absolute error between real and predicted volume fractions. The average of these 10 error values appears in the last column.

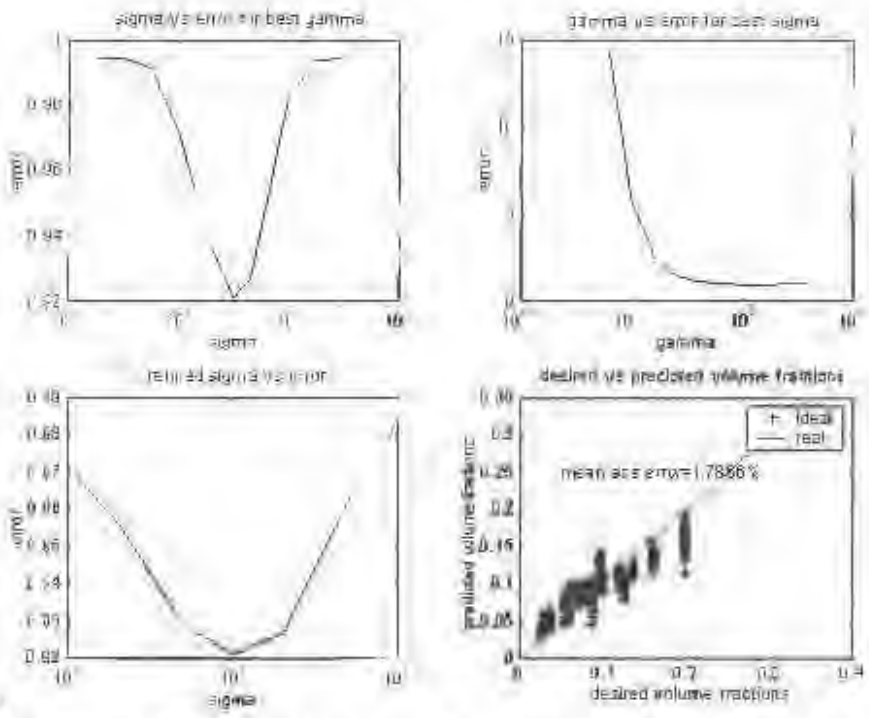
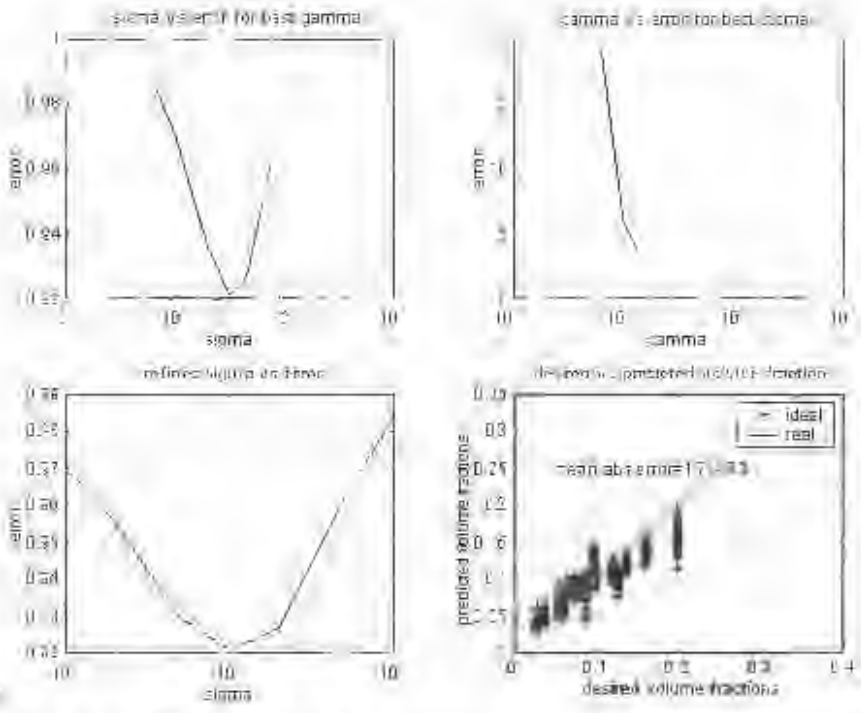


Figure 7-5: The results of using **KRR** to make volume fraction predictions based on the **line electrodes** dataset. The figure shows four subplots; the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (i.e. the line $\hat{y} = y$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's cross-sectional area) on the graph too.

line-lrr-svd



(log)

Figure 7-6: The results of using **KRR** to make volume fraction predictions based on the **line electrodes** dataset. A singular value decomposition (**SVD**) was performed on the data zeroing any singular values contributing less than 1% of the sum of all singular values. The figure shows four subplots; the top left and right graphs show a plot of error *vs.* σ (for best γ), and error *vs.* γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired *vs.* predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (*i.e.* the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's cross-sectional area) on the graph too.

7.1.5 Results for line & guard electrodes

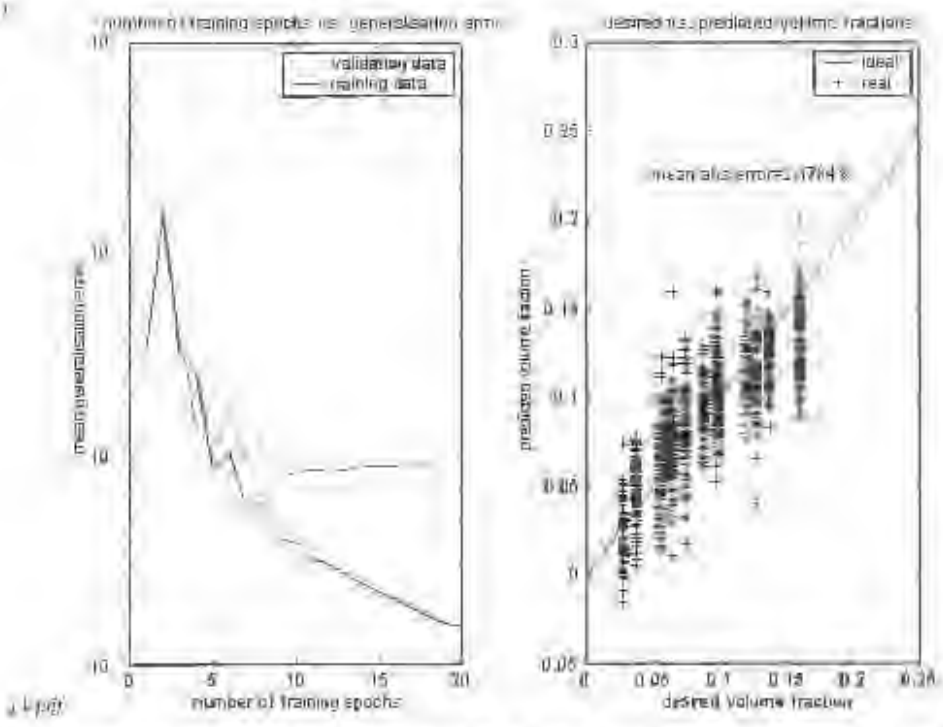


Figure 7-7: The results of using MLPs to make volume fraction predictions on the **line & guard electrodes** dataset. The figure shows two subplots; on the left, a graph showing the generalisation error for each epoch of weight adjustment during the resilient back propagation training algorithm. The blue line shows training dataset error, the green line, validation dataset error of one of the 10 MLPs trained. The subplot on the right shows a plot of the desired vs predicted volume fractions of an average performer of the 10 MLPs. Predictions made on the test dataset are represented by +'s, while the red line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error of all 10 MLPs appears written (as a percentage of the pipe's cross-sectional area) on the plot too.

MLP number											Mean overall error
Mean absolute error	1.98	1.99	2.02	3.91	1.86	2.28	2.03	1.93	1.79	1.93	2.18%

Table 7.3: A table displaying the results of training 10 separate MLPs on the line & guard electrodes dataset. The error is defined as the mean absolute error between real and predicted volume fractions. The average of these 10 error values appears in the last column.

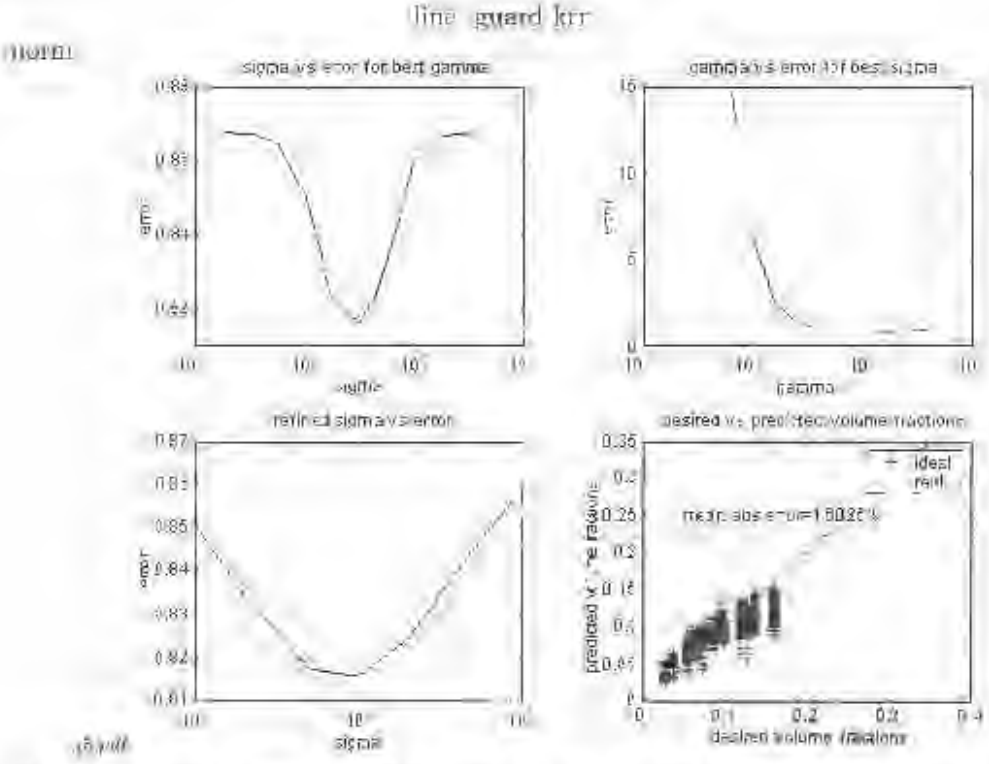


Figure 7-8: The results of using **KRR** to make volume fraction predictions based on the **line & guard electrodes** dataset. The figure shows four subplots, the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's crossectional area) on the graph too.

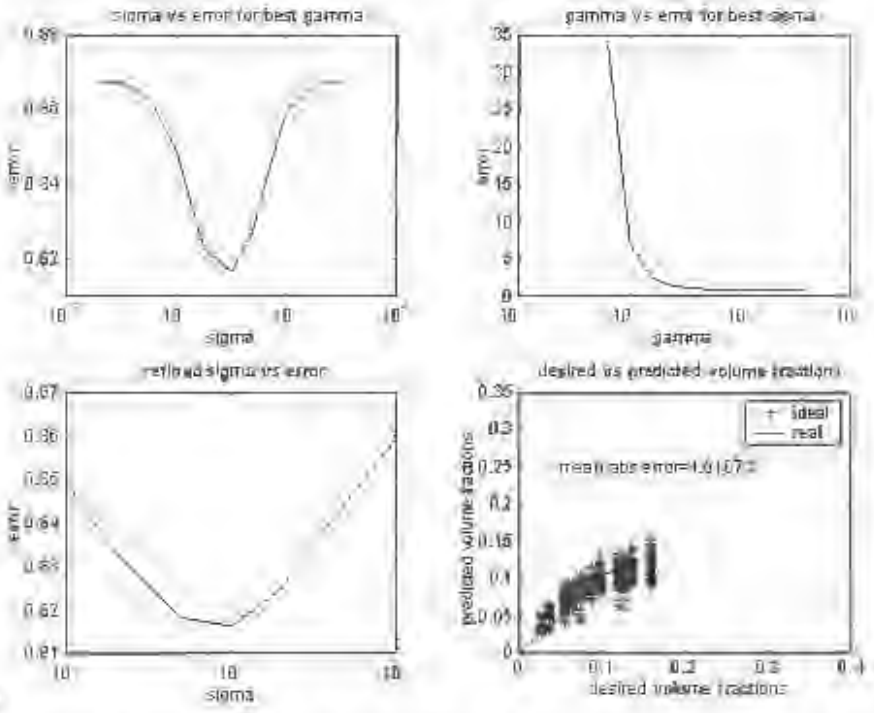


Figure 7-9: The results of using **KRR** to make volume fraction predictions based on the **line & guard electrodes** dataset. A singular value decomposition (**SVD**) was performed on the data—zeroing any singular values contributing less than 1% of the sum of all singular values. The figure shows four subplots; the top left and right graphs show a plot of error vs. σ (for best γ) and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using **KRR** with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by $+$'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's cross-sectional area) on the graph too.

7.1.6 Summary and discussion of static results

This section has demonstrated the results of using MLP NNs and KRR to predict the volume fractions of various statically captured datasets. All data was captured using the hardware developed by Giannopoulos [3], and upgraded by the author (i.e. a 16-electrode, FDM measurement strategy was employed). Only resistance data was captured, as opposed to both resistance and capacitance data being used in previous stages of the research. The reason for investigating different datasets has been to gain insight into the effects of using various electrode shapes and configurations in data collection. A summary of the results obtained appears in table 7.4:

Reconstruction method	Point electrodes	Line electrodes	Line & guard electrodes
MLP	2.75%	2.46%	2.17%
KRR	1.76%	1.79%	**1.60%**
KRR (SVD)	1.79%	1.80%	1.62%

Table 7.4: A table summarising the results of using KRR and MLPs to predict volume fractions of different datasets. The best performance appears in bold.

MLP NNs

In the project's antecedent stages, MLPs have proven to be useful volume fraction predictors. Specifically, the best results reported by Teague [1, 4] using a 16-electrode, bimodal, square-wave excitation, FDM EIT system, for 2-phase static mixtures was **1.15%**. Giannopoulos' best reported results obtained using an 8-electrode, bimodal, sine-wave excitation, FDM EIT system, for 2-phase static mixtures was **8.88%** [3]. Note that resistance data in both of the aforementioned research projects was obtained using point electrodes, and the performance measure used was the mean absolute error between predicted and desired volume fractions.

When applying the same MLP architecture and training method reported by Teague to the present system (16-electrode, unimodal, sine-wave excitation, FDM EIT) using point electrodes a performance of **2.75%** was obtained. This translates to a **2.4** times drop in performance (relative to Teague's reported 1.15%), which is expected, since only resistance data was used in reconstruction - meaning less information about the measurement space was available. An

improvement of **1.4 times** the result reported by Giannopoulos was experienced, which is significantly less than the performance difference relative to Teague's 16-electrode system. This indicates that a greater improvement in performance is experienced through the addition of capacitance data to a 16-electrode system than through the removal of capacitance data from an 8-electrode system and doubling of resistance electrode positions[2]. The fact that Teague used square-waves, which were proven to be outperformed by sine-waves [3], merely inflates the consequences of this observation.

The performance of MLPs was improved by **0.20%** when changing from point electrodes to line electrodes. A further improvement of **0.29%** was experienced with the addition of guard electrodes. This suggests that the performance of MLPs is improved when ensuring a more uniform field is used in the measurement space - a field with less fringing effects, which would otherwise further complicate the already highly complex mapping in EIT.

KRR

KRR has produced results almost equalling those reported by Teague. Oddly enough, the exact opposite ratios of performance improvement from Teague's system to the author's, and from the author's to Giannopoulos' system, than those obtained for MLPs has been attained (i.e. **1.4 times** degradation of performance from Teague's system to the present, and **2.4 times** improvement on Giannopoulos' system). Thus, in terms of KRR, it is evident that increasing the number of resistance electrode positions is more advantageous than including capacitance data.

KRR did not show any improvement in prediction accuracy when using line electrodes as opposed to point electrodes, and only a small improvement (**0.17%** and **0.16%** in the SVD case) was achieved through the addition of active guard electrodes. Although a slight degradation in performance was noticed between point and line electrodes, the difference is so small (0.03% and 0.03% in the SVD case) that this could easily be attributed to data-sampling errors. When performing SVD on the data before running KRR on it, a slight dip in performance was noticed

[2]Note that the addition of capacitance data to an EIT system does not mean that more electrode positions are added, but that more synchronous detectors are used in demodulation - reference signals orthogonal to excitation signals are used to extract the capacitance data, while in-phase coherent signals produce resistance data. Capacitance electrodes do however have to be mounted on the exterior of the vessel in the same positions as resistance electrodes.

in each data set. This shows that although there is a lot of, what appears to be irrelevant data in terms of SVD, it still holds information about the interior properties of the measurement space. This highlights the extent of ill-conditioning in EIT data.

Interestingly, in every KRR experiment performed by the author, an optimum σ- and γ-value of 10 was found through tenfold cross-validation. In each case, the error vs. σ curve possessed the characteristic shape of a kernel which 'fits' the data well (see figure 6-4). Considering that all data was normalised, a value as large as 10 at first seems like a hefty kernel width parameter for data with a standard deviation of 1. This is however not surprising considering the massive dimensionality of the input space. An optimum regularisation parameter (γ) of 10 seems large too; in fact it is the largest amount of regularisation the author has encountered in his experience with computational intelligence. The γ-value obtained is, however reinforced by the fact that the same amount of regularisation was found to be optimum by Herboldt [9] in his implementation of the MNR algorithm - this further highlights just how ill-conditioned EIT data is.

The most important thing to note about the results of using KRR in the experiments of this chapter, besides the improvements on MLPs, is the consistency of the optimum tuning parameter values obtained. This demonstrates the repeatability of results across various datasets, and further emphasizes the appropriateness of KRR to the task at hand.

7.2 Dynamic situation results for a 16-electrode unimodal FDM EIT system

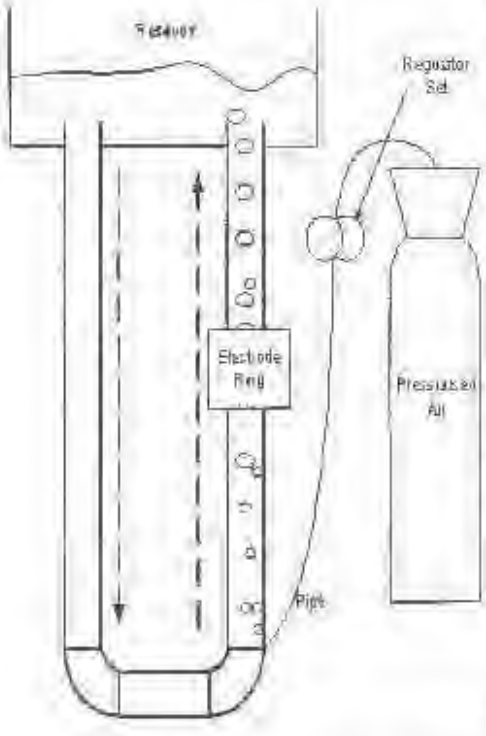
Although it has been the author's intention, since adopting the project, to perform dynamic tests at the CPUT pumping laboratory, delays and complications on their side motivated the author and project supervisor (Prof. J. Tapson) to design a simple testing apparatus of our own[3]. It was decided that the dynamic testing of the new rig should be performed on a scaled down version of the final application - an airlift. While a laboratory scale version of such an airlift will not be able to attain the same flow rates as expected in the final application, the

[3]Note that the use of the pumping rig at CPUT has not been abandoned, merely delayed to beyond the time frame of this thesis.

relative flow regime and conductivity distributions are expected to be similar.

Regarding the pumping laboratory at CPUT: although the originally intended tests have been delayed to a point beyond this thesis' scope, the author - thanks to the help of Dr. A. Wilkinson and Tim Long - managed to obtain real EIT data of a fast moving (up to $5m.s^{-1}$), 2-phase flow. The data was captured from the CPUT laboratory in December 2004 using Dr. A. Wilkinson's research group's measurement electronics [13] and electrode ring, and was supplied with the reconstructed conductivity distribution of each frame (obtained using the NOSER algorithm - a one step version of the MNR algorithm). Whilst this data was captured from a separate rig to the one used by the research group, using different measurement techniques, it is nonetheless *real* EIT data - providing a valuable resource on which the techniques developed in this document can be tested.

7.2.1 Airlift apparatus



The principle behind the simple laboratory scale airlift, designed by the author in conjunction with the project supervisor (Prof. J. Tapson), is illustrated in the accompanying diagram. Pressurised synthetic air is pumped through an airstone into the lower end of one side of a 'U-tube'. The air flow rate is controlled, rather finely, by a series of regulators. As air bubbles rise up the injection side of the U-tube, and through the electrode ring, water is transported upwards with the rising air - the displaced water is replaced by water from the reservoir travelling down the other side of the U-tube. Photographs of the apparatus, are included in the appendices.

Although the magnitude of flow velocities developed in this airlift rig will be far less than in the final application, the conductivity distribution and general flow regime is expected to be similar. Thus it should provide valuable insight into the sys-

tem's dynamic implementation. To compensate for the lower component velocities, a lower frame rate will be used to capture data. This will produce a flow to frame rate ratio more representative of the final application.

It should be noted that since gravel flow is the parameter of interest in the final application[1], air bubbles will be used to *simulate* rising gravel. This is viable because, relative to water, both air and gravel have practically zero resistance, so they will appear to have the same conductivity in a reconstruction based on purely resistance data. Hence, air is injected via an airstone to create small bubbles representing gravel pieces. Furthermore, the volume fractions of air in the pipe at any one time should be kept low in comparison to the surrounding medium, as is the case with the gravel in the real airlift.

Training data

At first, the author experimented with using the static training data collected in previous experiments to predict volume fractions of a dynamic flow. The results were poor, with unfit correlation between predicted volume fractions and varying flow rates. Performance was particularly poor at higher flow rates. The reasons for this were attributed to the fact that the bubbles used in the static database collection were unrepresentative of real flow. Even if smaller, more realistically shaped bubbles were manufactured, positioning them in scenarios representative of real flow would be near impossible. This is due to the fact that more than 10 bubbles may be present in the measurement space at one time. Ideally, networks trained using the 'real-world', empirical data approach should aim to collect training data from the very flow they are to make predictions about. This, however is extremely difficult to implement seeing as some means of obtaining target values (volume fractions) would need to be found, which is really what the project is trying to achieve in the first place.

It was thus decided to simulate training data using the finite element code developed by Herholdt [9] and modified by the author (documented in section 6.1.4). This would allow a training database to be generated rapidly. Also, more realistic bubble shapes, sizes and

[1]The reason why gravel flow rate is the parameter of interest and not air flow rate, was explained in section 5.1 on page 5). Briefly, it is due to the fact that the electrode ring could be placed below the air injection point, meaning that the hardware and reconstruction is simpler than if placed above, because only resistance data is necessary.

distributions would be attainable using this method. The use of simulated data has the added advantage of being generic, i.e. NNs trained on simulated data will not overfit data specific to a particular rig.

Since the proposed finite element simulations are based on a 2-d model of the measurement space, for reasons explained throughout the text, line electrodes would be used when testing. This would ensure that data captured from the real rig (test data) more closely resembles the training data (simulated measurements) used to predict its volume fractions.

Method

Much research has been done on using simulated data to train NNs in EIT. The techniques used to generate training data in this experiment are discussed below; they were adapted and modified for the task at hand using the methods documented in [19, 20, 21, 35, 36, 18].

Training data The following process is automated, and takes about 5 seconds to run on a 1.7GHz centrino^{TM} processor. The code is included in Appendix C - see "maketraindata.m", "buildtraindata.m", "maketraintestmatrix.m"

1. A range of bubble radii are specified - the range is chosen to suit the types of expected flow. For each bubble size, 100 sets of periphery measurements are simulated for different bubble positions such that the entire cross section of the vessel is filled with the superposition of these bubbles. Bubbles are simulated by creating an inhomogenity in a homogenous medium. More specifically, elements in a 1416 element mesh which fall inside a certain circle (bubble) are assigned a conductivity of 10^{-6} (approximate conductivity of air), while the surrounding medium is assigned a conductivity of 10^{-3} (approximate conductivity of tapwater).

2. Next, random combinations of up to 10 of the above bubble distributions are created and the corresponding measurements simulated. This process results in a training database of around 5000 instances. For each training instance, corresponding target values are determined by dividing the number of elements assigned to belong to bubbles by the total number of elements.

3. Gaussian noise is added to each training instance; the magnitude of which was determined by examining the noise present on the real system. The introduction of this noise was found to significantly improve performance of NNs in EIT by [18].

4. Each training instance (with added noise) has the simulated measurements of a homogeneous medium (conductivity = 10^{-3}) subtracted from it – this will ensure that only the *differences* caused by the introduction of bubbles will be used in training.

5. Finally, the training data is shuffled and normalised in the usual fashion.

Testing data Data was captured from the airlift rig at a frame rate of 10 $frames.s^{-1}$, 20 times slower than the 200 $frames.s^{-1}$ frame rate required by the final application. This was done because it was estimated that the highest bubble velocity in the laboratory rig was 20 times slower ($1\ m.s^{-1}$) than in the final application's[2] $20\ m.s^{-1}$. Two types of datasets were captured:

1. 10 sets of 300 frames of data were captured while the injection side of the U-tube was blocked – this was done so that the average flow rate could be measured for each set of 300 frames by measuring the amount of water displaced during that time period. During each data capture period (300 frames), the flow regulator valve was turned suddenly from zero to a flow setting indicated by the dial on the regulator (i.e. a step input). Each successive set of data had a slightly higher setting on the regulator's dial.

   This type of data set would provide means for a correlation to be made between average recorded flow rate and the average predicted volume fractions in each instance[3].

2. A considerably longer data set (3200 frames) was captured while the regulator's valve was 'modulated' in a particular pattern (e.g. square-wave followed by sawtooth-wave). The approximate pattern was recorded.

In each of the above, prior to real data being captured, 100 frames of data were captured from the homogenous carrier medium (tapwater). This was done so that the same calibration

[2] This specification (20 $frames.s^{-1}$) was stipulated by the project sponsors (DairTech) to Tungsten when he started research.

[3] The two are related as follows: $Flow\ rate = f(air\ flow) \times cross\text{-}sectional\ area$

feature as used in training would be performed (i.e. subtract the measurements of the homogeneous medium from each captured frame). By performing this sort of calibration, together with the usual data normalisation procedure[7], the discrepancies between real and simulated measurements - introduced by gains and offsets in the hardware's amplification chains - are cancelled.

Results

Both KRR and MLPs were trained using training data of the above nature. Because there were no precisely known target values for the test data set, the task of comparing performance was difficult. Furthermore, (in the case of MLPs) without known validation target values, early stopping could not be employed in training. This makes it difficult to ascertain whether the weight adjustment procedure is overfitting the training data, and whether the optimum number of training epochs has been reached. Thus, in the following only the results of using KRR on the test data are presented. This is due to a better correlation between mean volume fractions and average recorded flow rate being obtained by KRR, than for MLPs using the number of epochs investigated (20,30,40,50).

With respect to the tuning parameters for KRR, a value of 10^3 for both σ and γ was used in the graphs that follow. Other values of the same order were evaluated, but the trend of having $\sigma = \gamma = 10$ (seen throughout this document) appeared to hold true.

[7] Data normalisation procedure: from each column (or feature) in the training data, subtract the mean and divide by the standard deviation of that column; i.e. make the mean zero and the standard deviation 1.

[8] Note that a grid search for the optimum tuning parameters cannot be performed in experiments of this nature. This is owing to the lack of target values required in evaluating the error of a certain σ, γ combination. Hence, values need to be carefully chosen based on some *a priori* information about the data's characteristics.

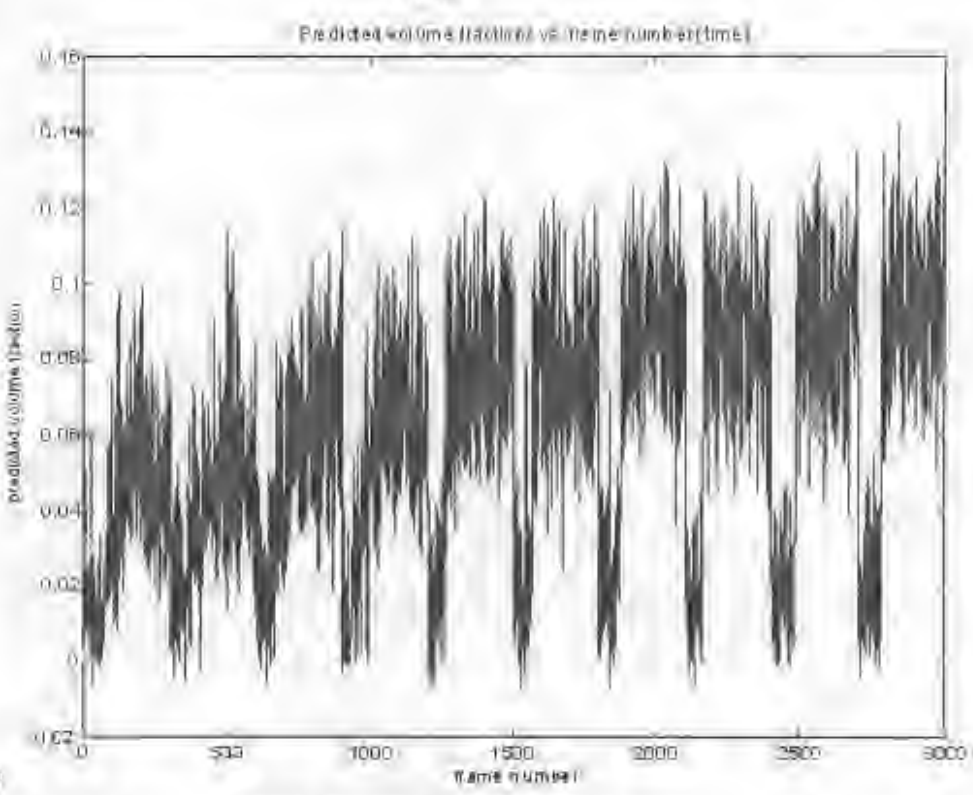


Figure 7.10: A figure showing a plot of the volume fractions predicted using KRR with $\sigma = \cdot, = 10$. The test dataset, which these predictions are based on, consisted of 10 subsets of 300 frames of data. Each set started out with zero airflow, followed by a sudden rise in airflow controlled by the flow regulator. Each successive set has a slightly higher setting on the regulator, giving the plot a 'rising staircase' effect. The approximate mean of each set's high volume fraction period is indicated by the horizontal lines in the data.

airlift rising mov

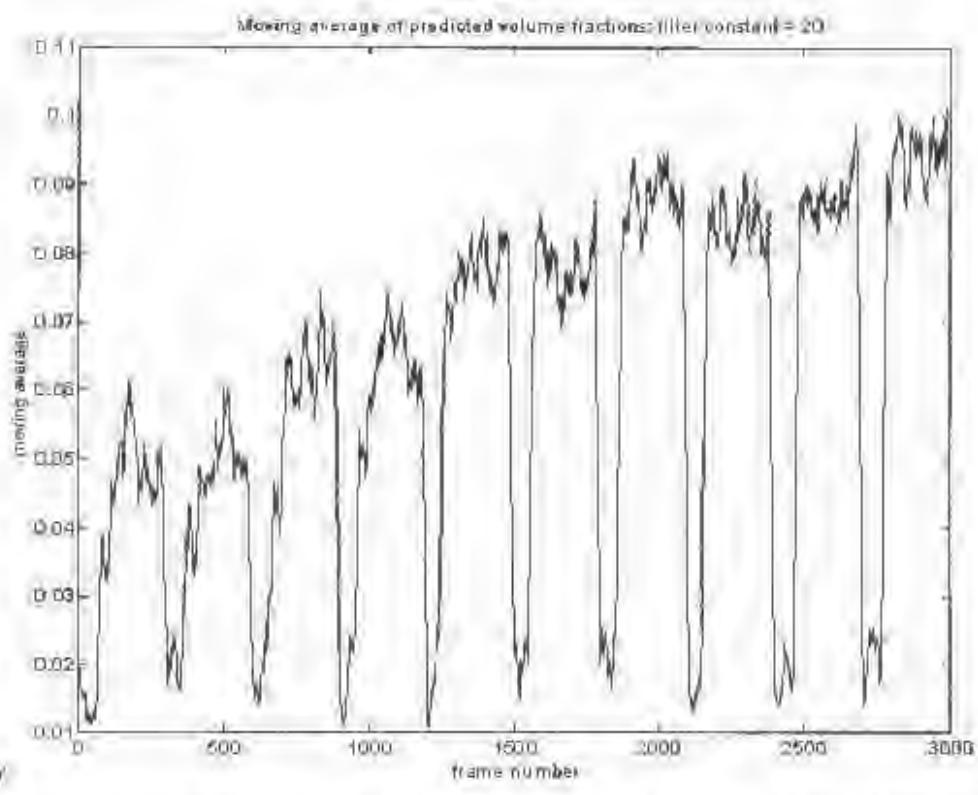


Figure 7-11: A moving average with filter constant equal to 20 - i.e. $M_{AVE}(n) = \frac{1}{20}\sum_{k=n-20}^{n} vol\_frac(k)$, for $n = 1$ to 3000. A clear jump is noticed between each dataset when they are concatenated, with flow regulator settings in ascending order, as they were for the test data.

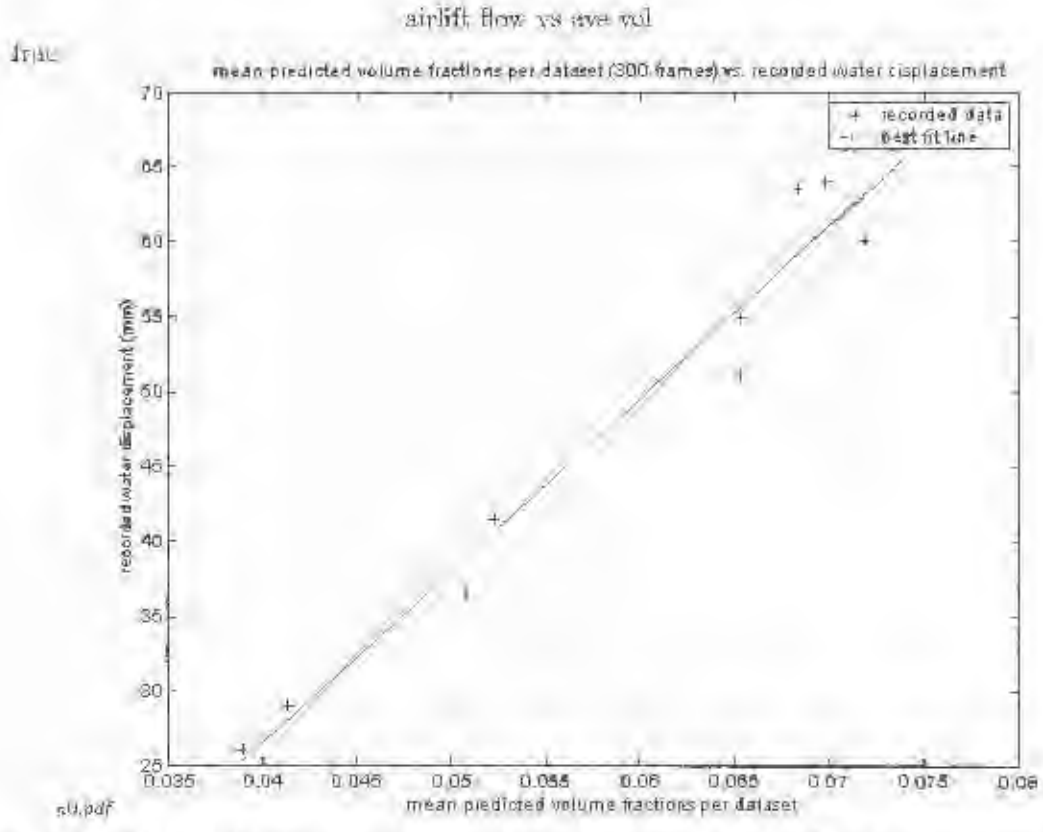


Figure 7-12: A figure showing data points and their best fitting line. The x-axis represents the mean of the volume fraction predictions for each dataset captured with a different setting on the flow regulator. The y-axis represents the amount of water displaced during data capture, while the injection side of the U-tube was blocked. A clear correlation can be seen in the data.

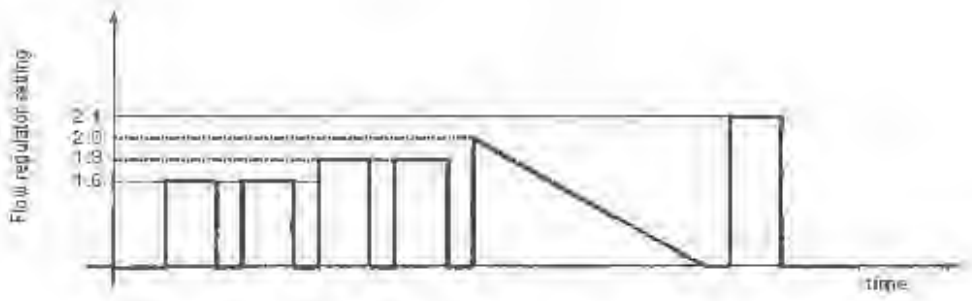


Figure 7-13: A sketch of the pattern with which the flow regulator's dial was modulated. The x-axis represents time, the y-axis shows the setting on the flow regulator dial.

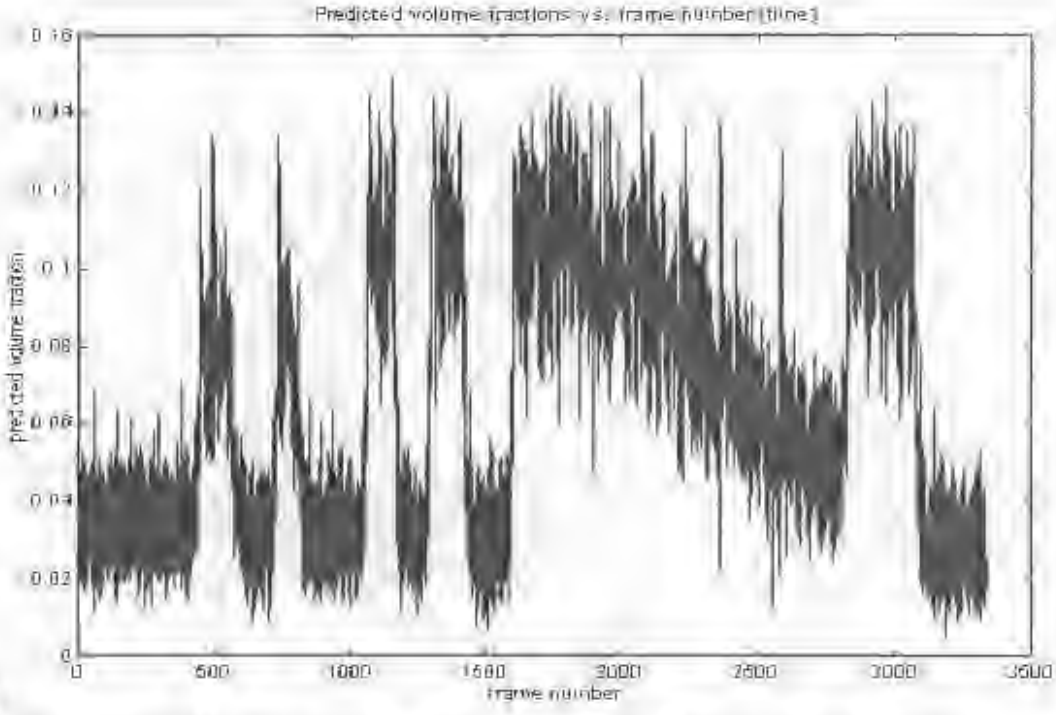


Figure 7-11: A figure showing the predictions made on the modulated test dataset vs. frame number. A clear and strong correlation between the modulation pattern applied to the salty, and the predictions of KRR is evident - even without taking a moving average.

7.2.2 Flow data

Method

The abovementioned data, captured from the UPUT pumping loop using Dr. A. Wilkinson's hardware ('flow data' hereafter), was used as training and testing datasets for both MLP NNs and KRR. The same procedures as detailed in the forerunning experiments were applied - i.e. data was randomly shuffled and normalised, followed by splitting the data into training and test sets; next, the training data was used to train10 MLPs with early stopping and a grid search for the optimum KRR tuning parameters was performed; the test data set (1000 'unseen' instances) was then used to produce a performance measure for each prediction technique.

In training (and testing), target values were obtained by calculating what percentage of elements (pixels) of the supplied reconstructed image were below a certain conductivity threshold. These values would then constitute the target values (volume fractions) relating to the corresponding frame. An example of a reconstructed image together with the measurement data used to obtain that image, and the thresholded version appears in the figure 7-13.

flow data

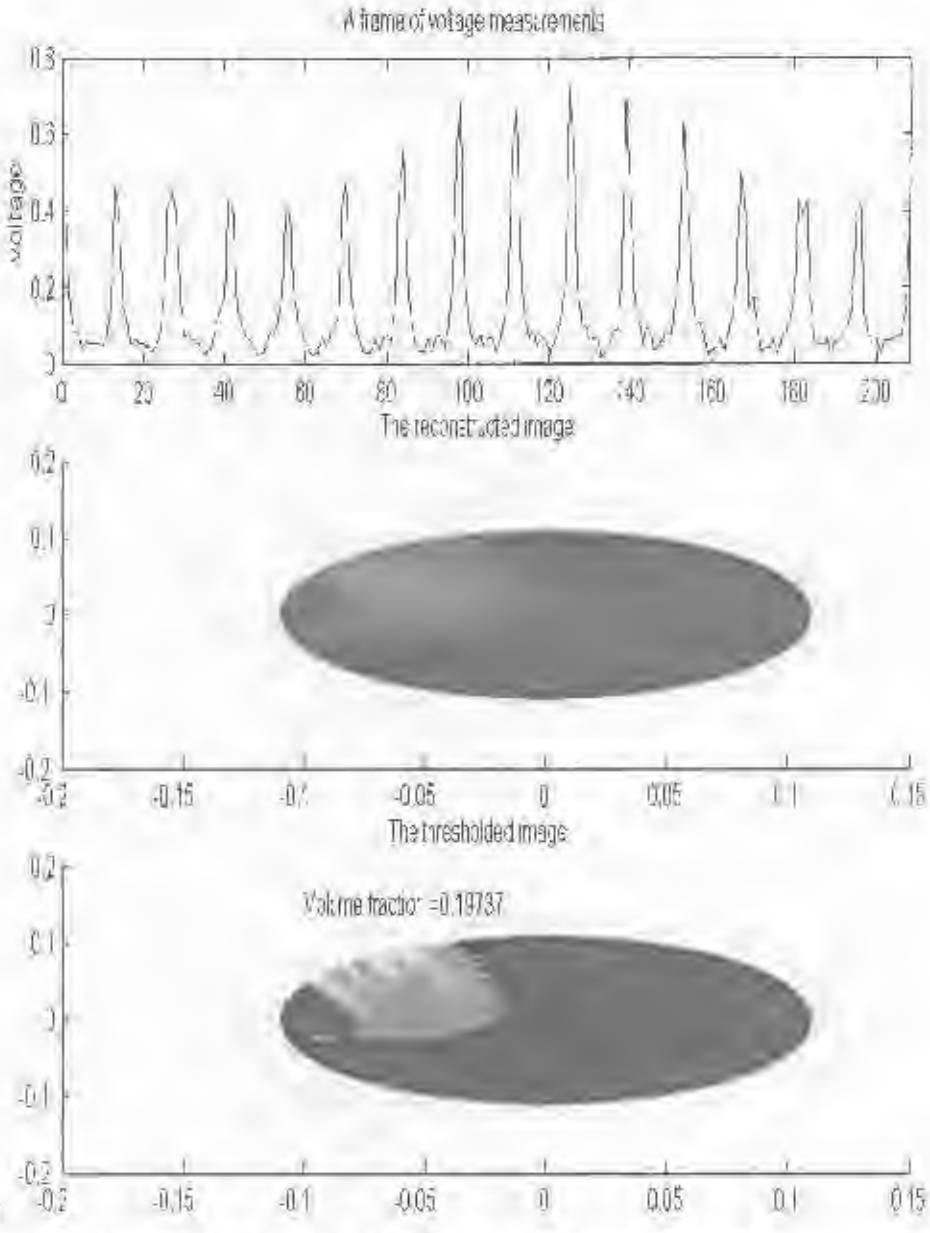


Figure 7-15: A figure showing 3 subplots. At the top is one frame of the flow measurement data - this is what is used for training data. In the middle, the reconstructed image performed by Dr. A Wilkinson's research group. At the bottom is the thresholded image showing how target values (volume fractions) are obtained.

Results

flow_data.mlp

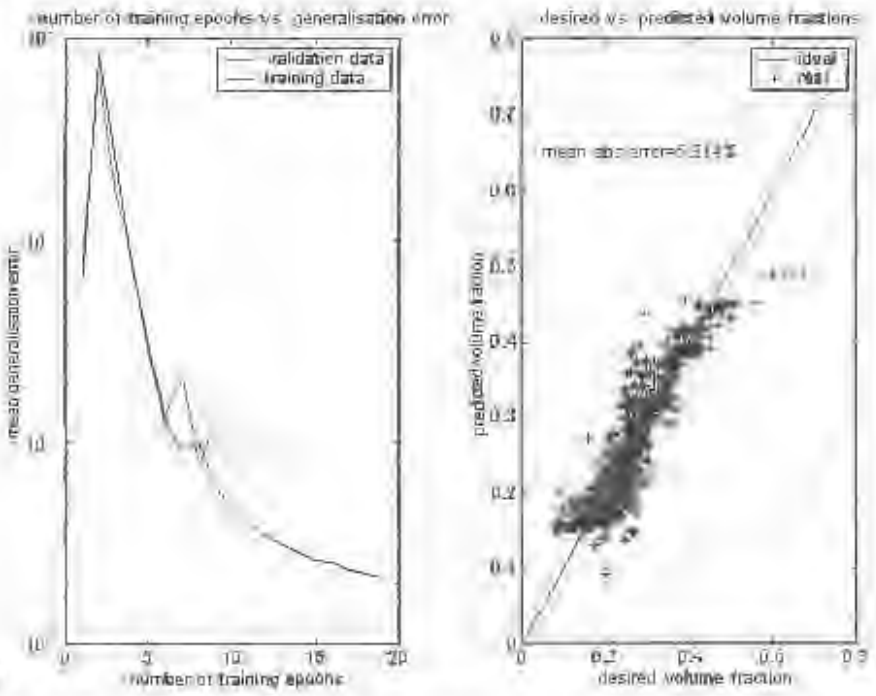


Figure 7-16: The results of using MLPs to make volume fraction predictions on the flow data dataset. The figure shows two subplots; on the left, a graph showing the generalisation error for each epoch of weight adjustment during the resilient back-propogation training algorithm. The blue line shows training dataset error, the green line, validation dataset error of one of the 10 MLPs trained. The subplot on the right shows a plot of the desired vs. predicted volume fractions of an average performer of the 10 MLPs. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error of all 10 MLPs appears written (as a percentage of the pipe's cross-sectional area) on the plot too.

MLP number	MLP1	MLP2	MLP3	MLP4	MLP5	MLP6	MLP7	MLP8	MLP9	MLP10	Mean overall
Mean absolute error	2.48	4.23	6.43	4.16	5.43	2.50	4.31	6.82	9.58	6.18	6.21%

Table 7.5: A table displaying the results of training 10 separate MLPs on the flow data dataset. The error is defined as the mean absolute errors between real and predicted volume fractions. The average of these 10 error values appears in the last column.

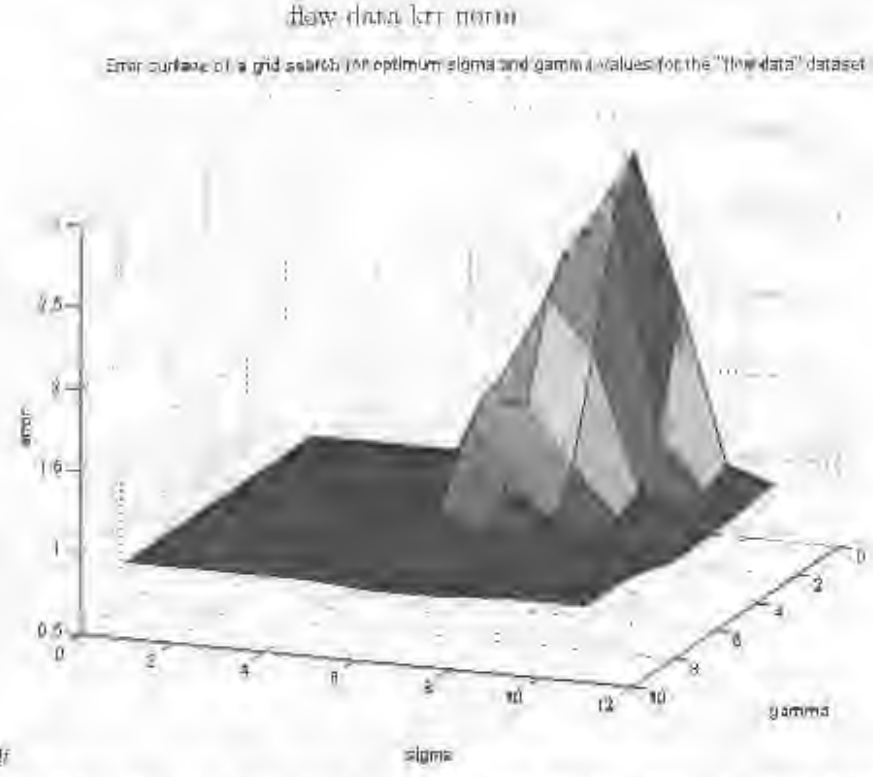


Figure 7-17: A surface plot of the mean generalisation error obtained using tenfold cross-validation and **KRR** for each σ, γ combination on the **flow data** dataset. The x-y plane represents σ and γ, the z-axis represents error. Note that the characteristic dip in error around a certain σ, γ combination is not visible in this surface plot. This is because the dip is relatively small with respect to the huge peak in error occuring in this figure. The dip blends into its surrounding plateau and is highlighted in later plots.

flow data krr

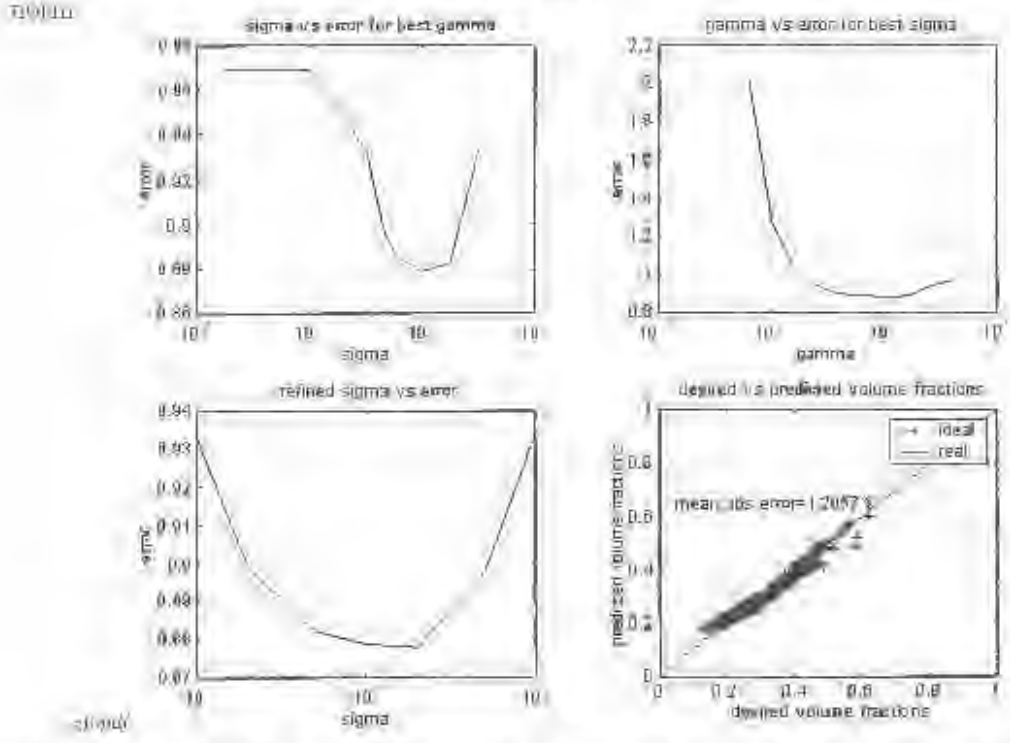


Figure 7-18: The results of using **KRR** to make volume fraction predictions based on the **flow data** dataset. The figure shows four subplots: the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by +'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipeline cross-sectional area) on the graph too.

flow data krr svd

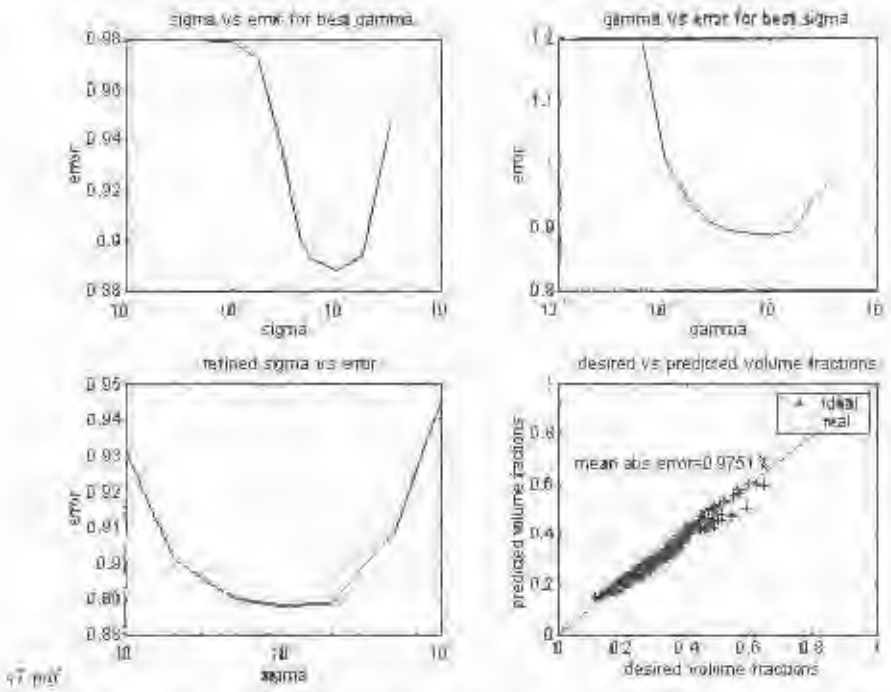


Figure 7-19: The results of using **KRR** to make volume fraction predictions based on the **flow data** dataset. A singular value decomposition (**SVD**) was performed on the data - zeroing any singular values contributing less than 1% of the sum of all singular values. The figure shows four subplots: the top left and right graphs show a plot of error vs. σ (for best γ), and error vs. γ (for best σ) respectively. The bottom left subplot shows the results of a refined σ search. The bottom right plot shows the desired vs. predicted volume fractions obtained using KRR with the optimum σ and γ tuning parameters. Predictions made on the test dataset are represented by '+'s, while the line shows the ideal, zero error response (i.e. the line $y = x$). The mean absolute error between desired and predicted values appears written (as a percentage of the pipe's cross-sectional area) on the graph too.

7.2.3 Summary and discussion of dynamic results

Airlift data

A simple airlift apparatus has been built and used to test the system's performance on real, dynamic flow situations. Simulated measurements have been used due to the complexity introduced by the training methods used previously. Further motivation for using simulated

measurements, with added noise, is that a more generic training database can be constructed - one which is not specifically representative of a certain rig.

MLPs and KRR have been trained with this data and used to make predictions on two test datasets: the first containing concatenated subsets of increasing average flow, the second a pattern of modulated flow. KRR was found to outperform MLPs, although an investigation into optimising MLPs to the task was limited due to the lack of desired target values needed in early stopping.

When comparing the mean of the volume fraction predictions made on each subset of the first data set with the average recorded flow rates associated with each subset, an almost constant ratio is obtained. This, together with the strong correlation between the predictions made from the second test data set and the modulation pattern applied to the regulator valve, suggests that the system holds potential for a volumetric flow meter.

Flow data

Further experiments were performed to investigate the use of images, produced by conventional ECT imaging techniques, to train computational intelligence (i.e. MLPs and KRR). Training data consisted of the periphery measurements captured during real, high velocity flow of a 2-phase mixture at the CPCT pumping laboratory. The measurements were captured from a different electrode ring using a different measurement strategy. Images were supplied courtesy of Dr. A Wilkinson's research group, and consisted of 836 conductivity pixels. Desired volume fractions were obtained by thresholding pixels below a certain conductivity.

Using the standard methods applied throughout this document, it was found that KRR outperformed MLPs, perhaps because an extensive MLP model search was not done. Model searches were not deemed necessary in previous investigations because Teague had already performed extensive model searches for data obtained using 16-electrode ECVM ECT techniques. If MLPs are to be extensively investigated for use with a new measurement strategy, another model search will be necessary.

KRR comes very close (0.97% mean absolute error) to predicting the volume fractions achieved by thresholding conventional images. Although the author realises that the images produced by Dr. A Wilkinson's reconstruction software should not be taken as 'gospel', the

results show that KRR and MLP NNs are performing very close to methods conventionally used for obtaining volume fractions. This allows an important and unique comparison to be drawn between the performance of conventional vs. computational intelligence methods - a discussion commonly occurring in the literature. It also demonstrates the ability of computational intelligence to extract underlying information from EIT data without first having to make an image.

The optimum tuning parameters found for KRR on the flow data were $\sigma = 20$ and $\gamma = 1$. This is interesting as it is the only data set which has produced σ and γ-values other than 10. The larger kernel width parameter (σ), is expected since the dimensionality of the input space has increased because of the different measurement strategy employed. A lower regularisation parameter value (γ) than other datasets is indicative of the fact that the data captured using the latter measurement strategy is less ill-conditioned, needing less regularisation.

Chapter 8

Conclusions

Based on the findings of this research, the following conclusions are drawn:

1. **The major factor effecting system resolution in EIT is measurement noise. When dealing with ill-conditioned problems, which EIT reconstruction inherently is, the signal to noise ratio must be maximised to ensure convergence.**

   A detailed investigation of the literature has indicated that the measurement protocol employed in the present system (2-electrode, voltage driven strategy), results in unknown contact impedance's between electrode and medium becoming significant. Thus, the system's maximal attainable resolution is fundamentally limited by the inclusion of these unknown's in the measurement loop. A 4-electrode, current driven strategy, which excludes contact impedances from the measurement loop, would appreciably diminish the system's measurement noise, thereby enhancing the conditioning of the reconstruction problem, and hence resolution.

2. **A new data acquisition system has been developed. It is provided with various sampling applications and should be extendable, without alteration, through the next few stages of the research.**

   Based on the recommendations of Teague's research, a more modern data acquisition module was purchased, and software developed to achieve the requirements of the sampling tasks at hand. The new system addresses Teague's concern of loss of synchronisation occurring between multiplexing address lines and sampling clock. The system includes

software applications suitable for online data capture and training database population. All sampling applications have been developed with extendability in mind, and are capable of performing on-the-fly averaging as well as the sampling of up to 2 planes of bimodal data.

3. **A new, open ended electrode ring has been fabricated which is capable of handling high velocity flows. It provides three types of electrode configurations, and is extendable to further research.**

   The electrode ring includes one plane of line electrodes with a plane of point-like guard electrodes on either side. Should the use of point electrodes be deemed necessary, one plane of guard electrodes can be used.

4. **Data collected using line electrodes, especially used concurrently with guard electrodes, has proved superior when used for volume fraction prediction. This is due to the non-linear effects of fringing fields being minimised.**

   It is known that the minimisation of fringing is necessary to ensure the accuracy of a 2-d Finite Element Method modelling approach. This is because a 2-d model takes no account of current flowing in a direction perpendicular to the measurement plane i.e. fringing. Thus line, and guard electrodes have been implemented in the new electrode ring, and must be used if 2-d Finite Element based methods are to be used (e.g. conventional reconstruction techniques or simulated training database population). The effects that increasing the electrode length has on contact impedance and averaging of fast moving flows have not been investigated, and may necessitate a return to point electrodes.

5. **A new meshing algorithm has been developed which, when used in conjunction with Herboldt's modified Finite Element code, produces reasonably accurate simulations of boundary measurements for a circular geometry, EIT rig.**

   The meshing algorithm is generic in that it can be adapted to suit any circular geometry. Only the function which places nodes on the actual electrodes would need to be altered if the size of the electrode ring was to change. Although the simulated boundary measurements are not precisely equal to measurements obtained from the real rig due to

the effects of unknown contact impedances, the ability to simulate these measurements provides a useful verification tool, and can be used in training database population. The discordances between real and simulated measurements, is however, too significant to produce useful images using the MNR algorithm.

6. **Kernel Ridge Regression consistently outperforms its Multi-Layer Perceptron Neural Network counterparts in volume fraction prediction of static situations.**

   A number of static experiments were performed in the same manner as throughout the project's history. Although results did not surpass those quoted by Teague, they came close to it, which is noteworthy since only unimodal data was used in the experiments of this thesis. The consistency of the tuning parameters found through the experiments performed, as well as the distinct dip in error around a consistent kernel width, shows that the results are repeatable and that kernel methods are well suited to EIT reconstruction.

7. **Boundary measurements simulated using the Finite Element Method (with added Gaussian noise), produce a more representative training database than the empirical training methods used previously, when used to predict volume fractions of real flow. Furthermore, simulated measurements can be produced much more efficiently than empirical ones.**

   Kernel Ridge Regression and Multi-Layer Perceptron Neural Networks were trained using the empirically acquired static training data acquired in forerunning experiments. The results were poor with no consistent correlation between predictions and recorded flow rate. This was believed to be the case since the bubble shapes and configurations used in static databases were so much unlike those of a real flow. Hence, to obtain a more realistic flow representation, simulated measurements were used - the results displayed a strong correlation between mean volume fractions and recorded average flow.

   One drawback of simulated measurements, which is one of the reasons why they have been avoided throughout the project's history, is that discrepancies between real and simulated measurements - owing to modelling errors - may degrade performance. However, a more generic training database is attainable using simulated measurements, one which is not specifically bias towards a specific rig.

8. **Kernel Ridge Regression, using simulated training data, appears to perform well in predicting volume fractions of a rising-air-bubbles flow; a strong correlation between mean predicted volume fractions and average flow rate is obtained.**

9. **Computational intelligence, in particular Kernel Ridge Regression, when trained using conventional images of a flow, can be used to predict volume fractions of similar flows directly.**

   Using measurement data supplied by Dr. A. Wilkinson's research group, together with the reconstructed images produced by their world class ERT system, a training database was populated - instances consisted of measurement data; target values were obtained by thresholding pixels below a certain conductivity. Multi-Layer Perceptron neural networks and Kernel Ridge Regression were trained on some of the data to predict volume fractions of the rest. Kernel Ridge Regression predicted volume fractions of the thresholded images to within a mean absolute error of less than 1%.

Chapter 9

Recommendations for further research

Based on the findings and conclusions of the research, the following recommendations for further development are made:

1. **Investigation into a 4-electrode, current driven measurement strategy for frequency division multiplexed EIT should be initiated.**

   While frequency division multiplexing offers a unique advantage in EIT, the way in which it is currently implemented by the research group (2-electrode, voltage driven) limits reconstruction resolution. Moreover, if using simulated measurements to train computational intelligence, the application of a more accurate measurement strategy will certainly produce a more representative training database, and hence better prediction accuracy.

2. **The system should be tested, as originally intended for this project, at the high velocity pumping loop at CPUT.**

   Although complications delayed the testing of the present system at CPUT to beyond the scope of this project, good relations are still held with Mr. A. Sutherland, and intensions are that high velocity flow data is to be captured from their pumping rig as soon as possible.

3. **The real-time implementation issues of kernel methods should be investigated.**

Although Kernel Ridge Regression has proved to be a successful volume fraction prediction tool in the context of this research, its real-time performance may be an issue. This, due to the time taken to evaluate the distance between each training point - of which there need to be many for kernel methods to be effective - and each test point. However, sophisticated methods of adjusting the kernel weights, such as Support Vector Regression, exist. Support Vector Machines, through the zeroing of kernel weights which do not significantly effect the decision function, achieve data compression whereby only the 'support vectors' are used in function evaluation. This is known to appreciably reduce computation time and has been implemented in similar research.

4. **Simulated measurements should become the method used in training database population for prediction of real flows.**

   This project has shown that the empirical methods used previously are ill-suited to making volume fraction predictions about real flow data. Unless realistic bubble shapes - as expected in the final application - can be fabricated and efficiently distributed in the measurement space, simulated measurements are expected to outperform empirical ones. As emphasised throughout the text, the use of simulated measurements ensures that the training database is a generic one, which allows more sound conclusions to be drawn about the developed methods' applicability to the final application.

5. **A package to reconstruct EIT images in the conventional manner should be obtained by the research group.**

   Off-the-shelf, optimised, conventional EIT image reconstruction packages are readily available and often free for research purposes (e.g. EIDORS). Although this project's goals are far removed from those of the conventional image-reconstruction-type approach, it would be beneficial for the research group to have the ability to easily reconstruct images. This would go a long way in verifying system performance and, as shown in the experiments of this research, may be useful in training computational intelligence to directly predict volume fractions.

   One factor to consider before attaining such a package, would be that the EIT community at large considers the 4-electrode current driven strategy as the norm. As a result, data

acquired through a different strategy (such as the one used in the present system) is ineffectual, when applied to conventional methods. Hence as recommended above, the measurement strategy would need to be altered.

Bibliography

[1] G. Teague, "Neural network reconstruction for electrical capacitance tomography system," 2000. B.Sc. Thesis.

[2] Q. Smit, "Material phase detection using capacitance tomography," Master's thesis, University of Cape Town, 2000.

[3] A. Giannopoulos, "Investigation of sine-wave inputs for an FDM EIT system," Master's thesis, University of Cape Town, 2003.

[4] G. Teague, *Mass Flow Measurement of Multi-Phase Mixtures by Means of Tomographic Techniques*. PhD thesis, University of Cape Town, 2002.

[5] G. Teague, J. Tapson, and Q. Smit, "Neural network reconstruction for tomography of a gravel-air-seawater mixture," *Measurement Science and Technology*, vol. 12, pp. 1102–1109, 2001.

[6] J. Tapson, "Neural networks and stochastic search methods applied to industrial capacitive tomography," *Control Engineering Practice*, vol. 7, pp. 117–121, 1999.

[7] V. Capindissa, "Monitoring an industrial hydrocyclone using frequency domain multiplexing electrical impedance tomography," Master's thesis, University of Cape Town, 2005.

[8] J. Webster, *Electrical Impedance Tomography*, ch. 1 - Electrical Impedance Imaging, pp. 1 7. IOP Publishing, 1990.

[9] S. Herboldt, "Image reconstruction on electrical resistance tomography," Master's thesis, University of Cape Town, 2004.

[08] A. Vauhkonen, *Electrical Impedance Tomography*, ch. 8 – Data Collection Methods, pp. 75–88. IOP Publishing, 1999.

[11] S. J. Northling, E. W. Randall, J. J. Cilliers, W. Xie, and A. J. Wilkinson, "Electrical resistance tomography using a bi-directional current pulse technique," *Measurement Science and Technology*, vol. 12, pp. 997–1001, 2001.

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

[13] A. J. Wilkinson, E. W. Randall, D. Durrett, T. Naidoo, and J. J. Cilliers, "The design of a 1000 frames per second ERT data capture system and calibration techniques employed," in *3rd World Congress on Industrial Process Tomography*, Banff, 2003.

[14] G. Teague and J. Tapson, "High frame-rate electrical impedance tomography using frequency division multiplexing," 2002. University of Cape Town.

[15] E. L. Yuen, R. Mann, T. A. York, and B. D. Grieve, "Electrical resistance tomography (ERT) imaging of a metal-walled solid-liquid filter," in *Proceedings of The 2nd World Congress on Industrial Process Tomography*, (Hannover), 2001. As cited by Teague.

[16] 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," *Transactions on Medical Imaging*, vol. 12 no. 2, pp. 137–146, 1993.

[17] C. G. Xie, A. L. Stott, A. Plaskowski, and M. S. Beck, "Design of capacitance electrodes for concentration measurement of two-phase flow," *Measurement Science and Technology*, vol. 1, pp. 65–78, 1990.

[18] A. Adler and R. Guardo, "A neural network image reconstruction technique for electrical impedance tomography," *IEEE Transactions on Medical Imaging*, vol. 13, no. 4, pp. 593–599, 1994. As cited by Teague.

[19] A. Y. N. B. S. Hoyle and N. J. Bailey, "Neural network for pattern association in electrical capacitance tomography," *IEEE Proceedings on Circuits Devices and Systems*, vol. 141, no. 6, pp. 517–524, 1994.

[20] B. S. Hoyle, N. J. Bailey, and A. Y. Nooralahiyan, "Performance of neural network in capacitance-based tomographic process measurement systems," *Measurement and Control*, vol. 28, pp. 109–112, 1995.

[21] A. Y. Nooralahiyan and B. S. Hoyle, "Three-component tomographic flow imaging using artificial neural network reconstruction," *Clinical Engineering Science*, vol. 52, no. 11, pp. 2139–2148, 1997.

[22] H. Demuth and M. Beale, "Neural network toolbox user's guide," 2004. The MathWorks.

[23] J. Greene, "Course notes of EEE4965 - neural, fuzzy and evolving systems," 2003. A Course Taught in the Department of Electrical Engineering, University of Cape Town.

[24] P. Hua and E. J. Woo, *Electrical Impedance Tomography*, ch. 10 - Reconstruction Algorithms, pp. 97–137. IOP Publishing, 1990.

[25] T. Long, "A software front end for a real-time electrical resistance tomography imaging system," 2003. B.Sc. Thesis.

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

[27] E. J. Woo, *Electrical Impedance Tomography*, ch. 11 - Computational Complexity, pp. 138–157. IOP Publishing, 1990.

[28] N. Polydorides and W. R. B. Lionheart, "A matlab toolkit for three-dimensional electrical impedance tomography: A contribution to the electrical impedance and diffuse optical reconstruction software project," *Measurement Science and Technology*, vol. 13, pp. 1871–1883, 2002.

[29] R. M. West, H. S. Tapp, D. M. Spink, M. A. Bennett, and R. A. Williams, "Application-specific optimization of regularization for electrical impedance tomography," *Measurement Science and Technology*, vol. 12, pp. 1050–1058, 2001.

[30] J. L. Wheeler, W. Wang, and M. Tang, "A comparison of methods for measurement of spatial resolution in two-dimensional circular EIT images," *Physiological Measurement*, vol. 23, pp. 169–176, 2002.

[36] T. J. Yorkey, J. G. Webster, and W. J. Tompkins, "Comparing reconstruction algorithms for electrical impedance tomography," *IEEE Transactions on Biomedical Engineering*, vol. 34, pp. 843–852, 1987. As cited by Herbolds.

[37] K. H. Cho, S. Kim, and Y. J. Lee, "Impedance imaging of two-phase flow field with mesh grouping method," *Nuclear Engineering and Design*, vol. 204, pp. 57–67, 2001.

[33] D. de Ridder, R. P. W. Duin, P. W. Verbeek, and L. J. V. Vliet, "The applicability of neural networks to non-linear image processing," *Pattern Analysis and Applications*, vol. 2, pp. 111–128, 1999.

[34] B. H. Blott, A. S. Miller, and T. K. Hames, "Neural networks for electrical impedance tomography image characterisation," *Clinical Physics and Physiological Measurement* vol. 13, Supplement A, pp. 119–123, 1992.

[35] N. J. Bailey, B. S. Hoyle, D. M. Spink, and A. Y. Noorelahiyan, "Neural networks in process tomography," 1998. electronic reference: http://www.cc.leeds.ac.uk/homes/NJB/Research/bergen/bergen.html. As cited by Teague.

[36] E. Ratajewicz-Mikolajczak, G. B. Shirkoohi, and J. Sikora, "Two ANN reconstruction methods for electrical impedance tomography," *IEEE Transactions on Magn.*, vol. 34, pp. 2964–7, 1998. As cited by Teague.

[37] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[38] R. Plonsey and R. Collin, "Electrode guarding in electrical impedance measurements of physiological systems-a critique," *Medical and Biological Engineering and Computing*, vol. 15, pp. 519–527, 1977.

[39] X. Deng, F. Dong, L. J. Xu, X. P. Liu, and L. A. Xu, "The design of a dual-plane ERT system for cross correlation measurement of bubbly Gas/Liquid pipe flow," *Measurement Science and Technology*, vol. 12, pp. 1021–1031, 2001. As cited by Teague.

[40] D. J. Nowicki, *Electrical Impedance Tomography*, ch. 4 - Current Generators, pp. 29–42. IOP Publishing, 1990.

[41] P. Horowitz and W. Hill, *The Art of Electronics.* Cambridge University Press, 1995.

Appendix A

Circuit Diagrams

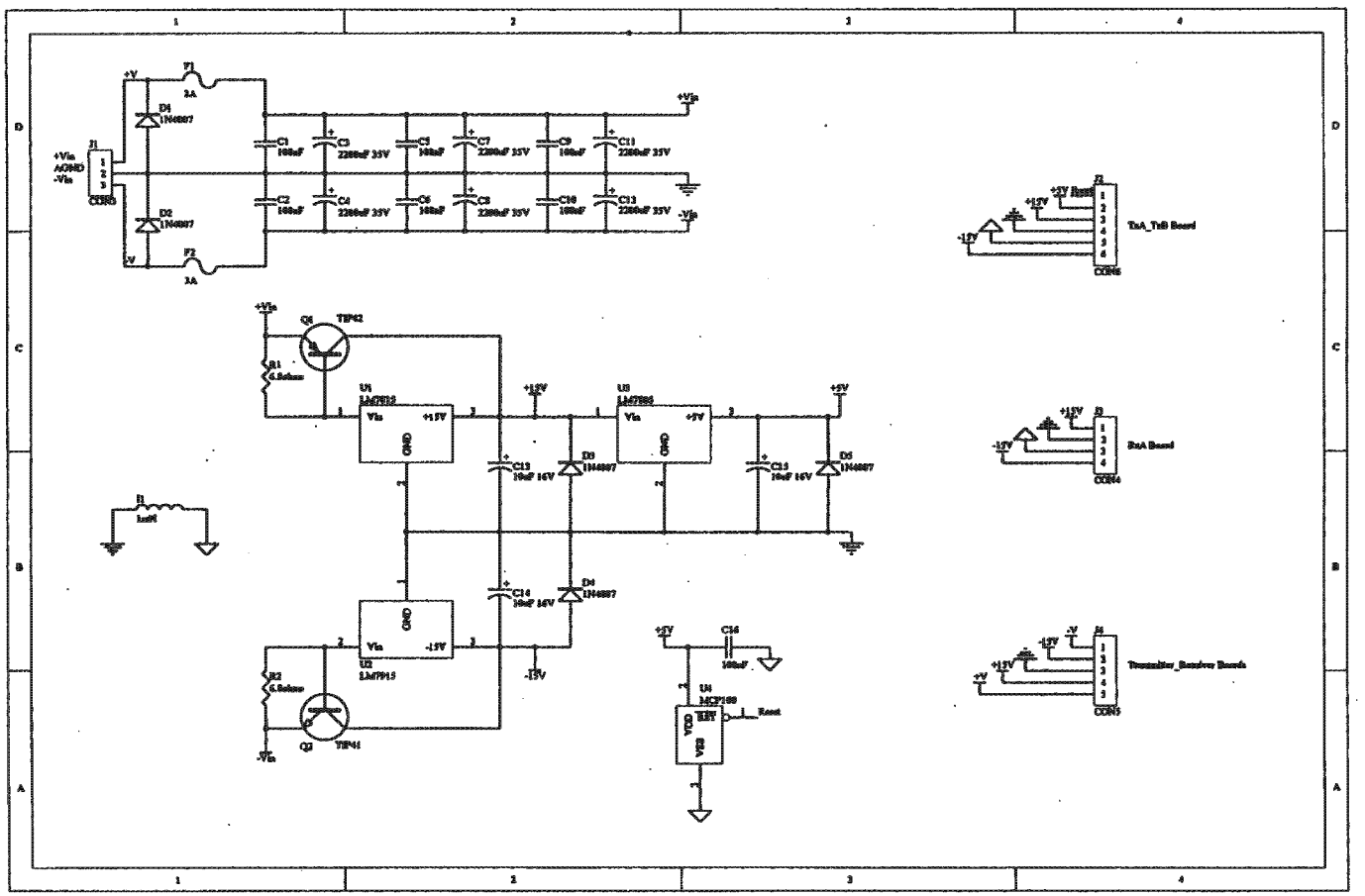
The circuit diagrams presented in this appendix document the state of the measurement hardware as was the case at the commencement of this thesis.

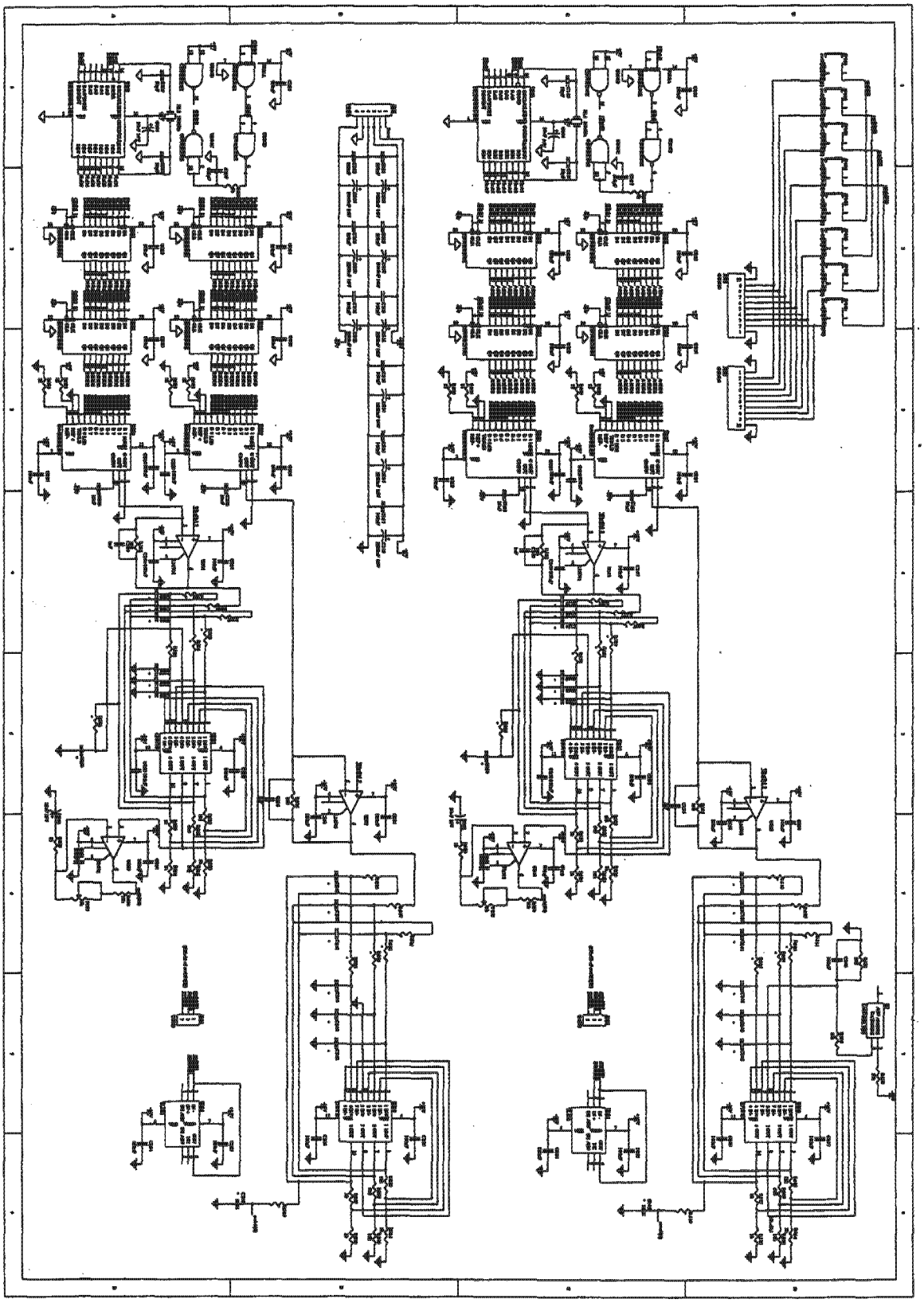
The circuit diagrams of those systems designed by Giannopoulos and upgraded by the author, but without significant alteration, are presented in the form of Giannopoulos' circuit diagrams from which the PCBs were constructed. The changes made by the author have been documented in the body of the text and further in Appendix B. Those sub-systems are:

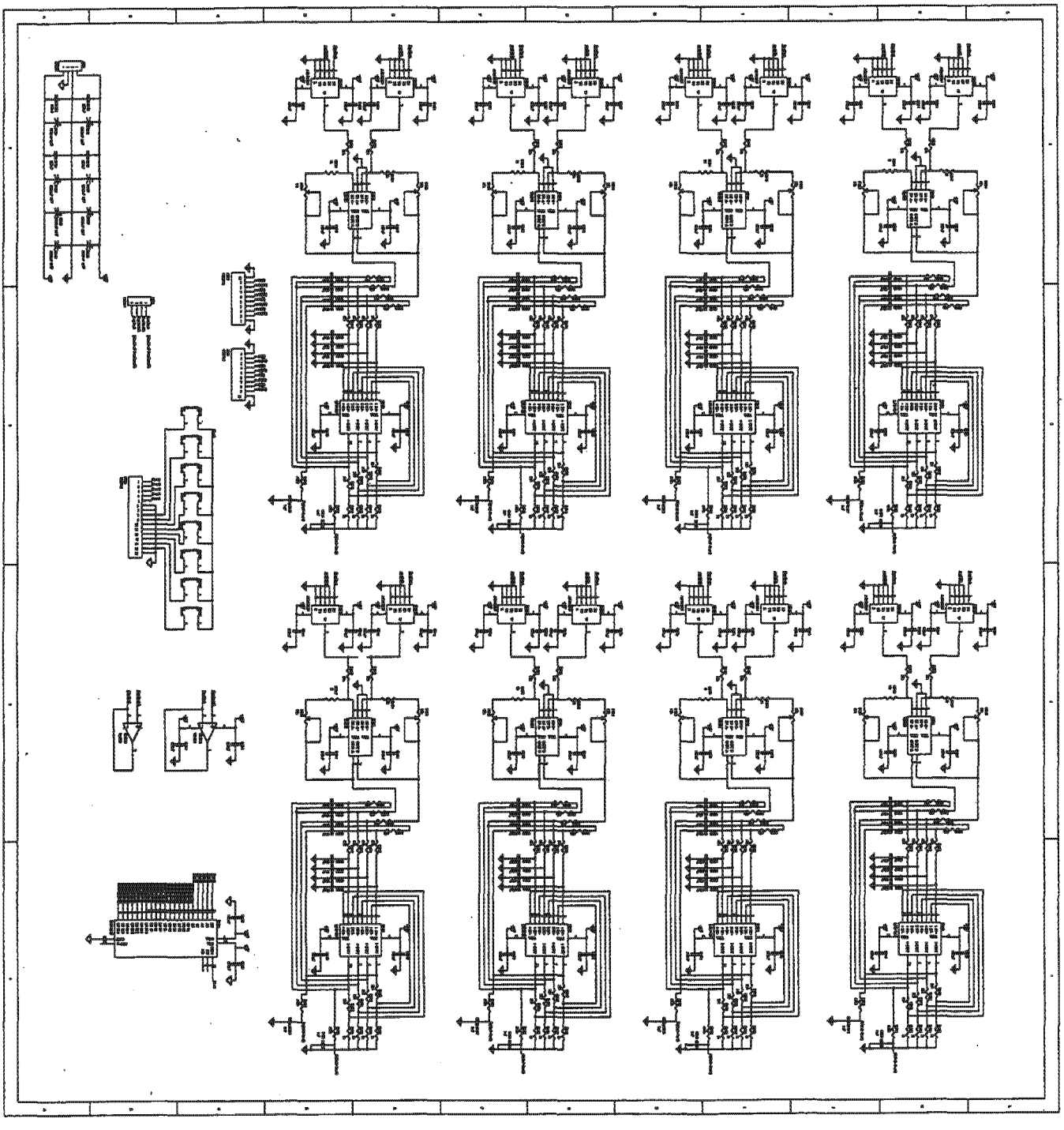
1. Power supply
2. Frequency Generation Board
3. Demodulation Board

   They appear in the above order in the pages that follow. The sub-systems redesigned and built by the author follow those, and appear as redrawn circuit diagrams in the following order:

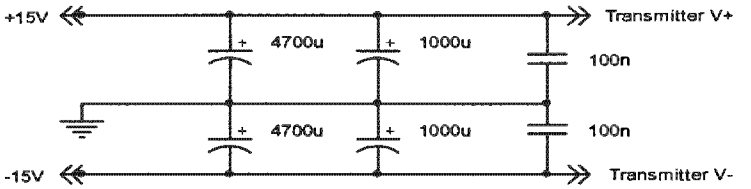
4. Transmitter Amplifier
5. Receiver Amplifier
6. Sample Controller Board


+Vin
-Vin
GND
+15V
-15V
+5V
TxA_TxB Board
RxA Board
Transmitter_Receiver Boards

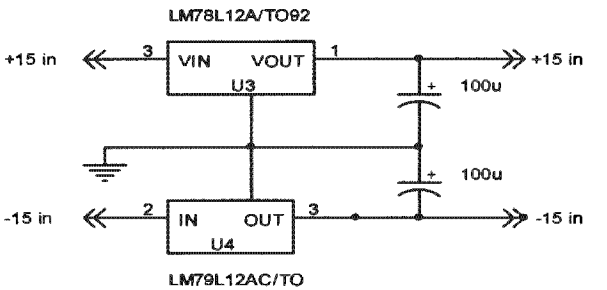




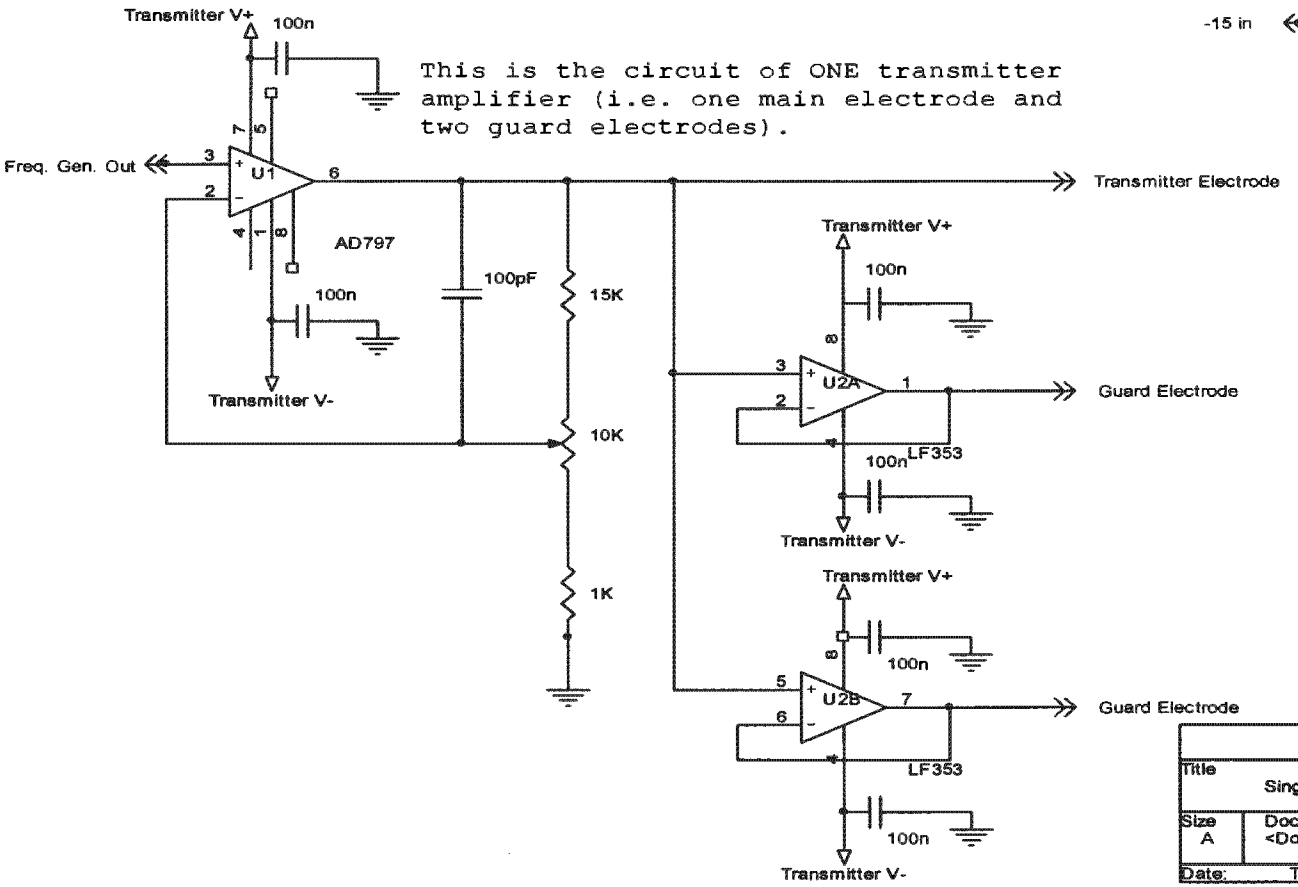
One bank of smoothing caps for all 8 transmitters on one board.



One regulator circuit per transmitter chan

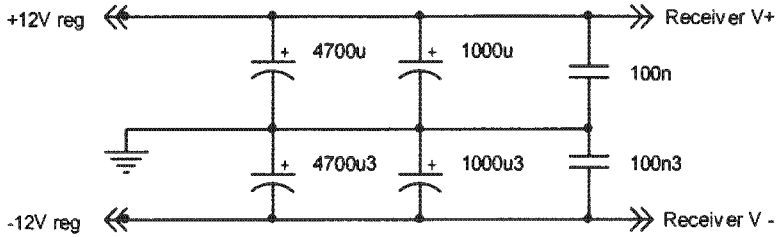


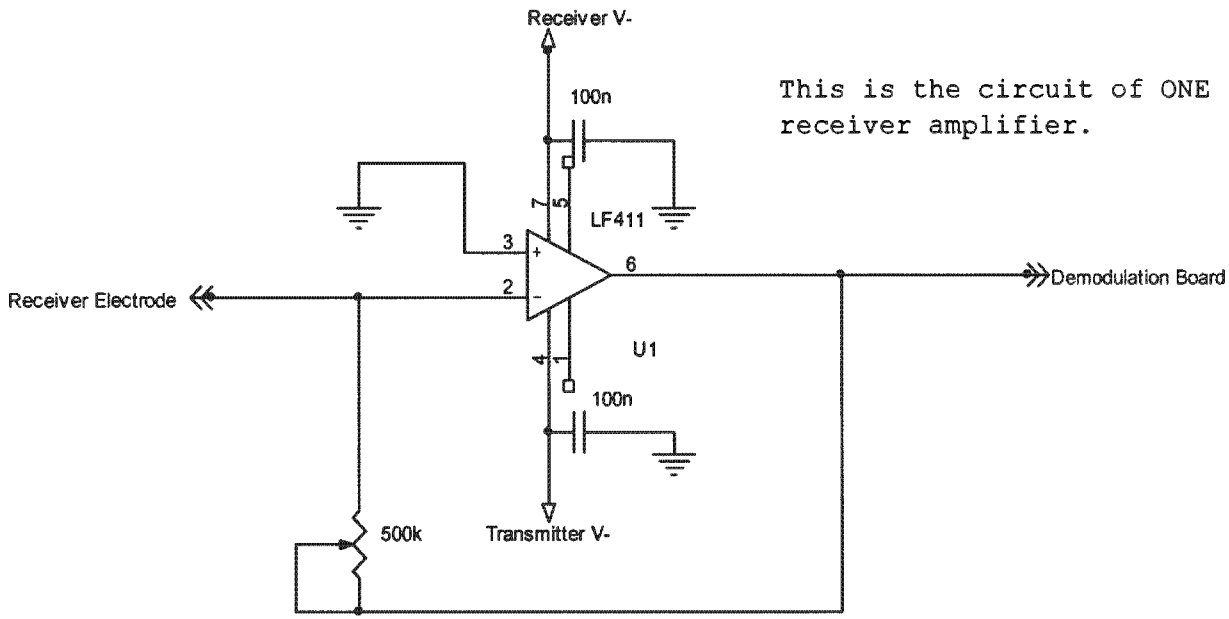
This is the circuit of ONE transmitter amplifier (i.e. one main electrode and two guard electrodes).



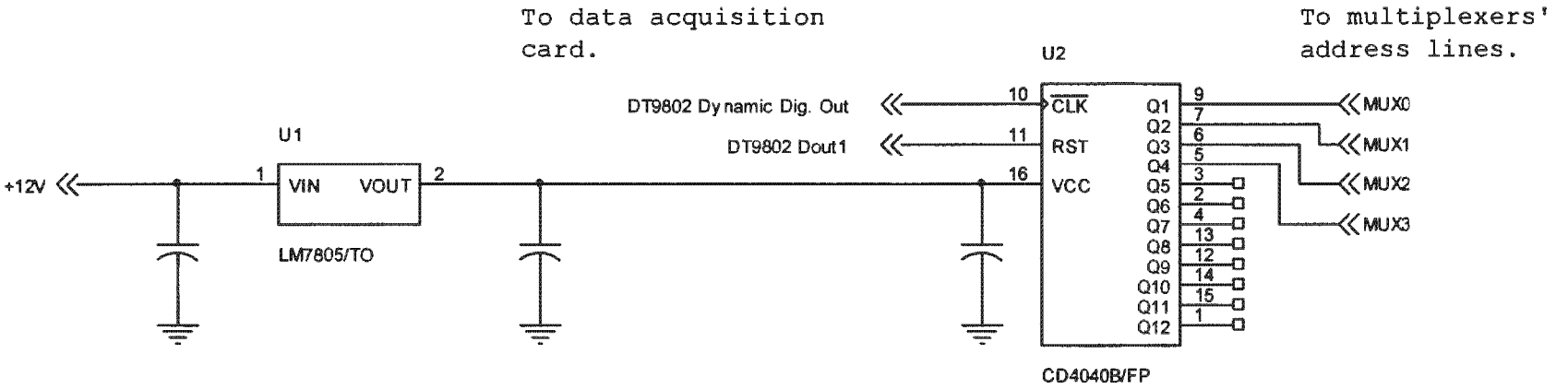
Title	Single transmitter channel	
Size A	Document Number <Doc>	
Date: Tuesday, July 19, 2005	Sheet 1 of 1	

One bank of smoothing caps for all 8 receivers on one board.





This is the circuit of ONE receiver amplifier.



Title

Sample Controller Board

Appendix B

A Guide to the System's Measurement Hardware

This thesis has been mainly concerned with the theoretical issues involved with the implementation of a 16-electrode FDM EIT system. Little attention has been paid to explaining the workings of the measurement hardware, even though a significant time of the author's research was involved with gaining an understanding of it and upgrading it. The reasons for not focusing on the measurement hardware in the body of this document are numerous; firstly, the principles of how the hardware achieves resistance or permittivity measurement of the space has been well documented in the projects forerunning this work [4, 3, 7]. Furthermore, the major complexities concerned with EIT reconstruction are involved with the data manipulation and signal processing of the data captured from the measurement hardware - these complexities have been the focus of this thesis, deeming the documentation of the measurement hardware subordinate.

Nonetheless, the author felt it important to include some practical explanations of the hardware - for completeness, and to demonstrate the work done by the author (on the measurement hardware), which has not been documented in the body of this text. Thus, this appendix is dedicated to the explanation of the principles of operation of the system's measurement hardware, and the documentation of the work done by the author on it. Note that all the present system's circuit diagrams appear in the previous appendix and are thus not repeated when they

are discussed below.

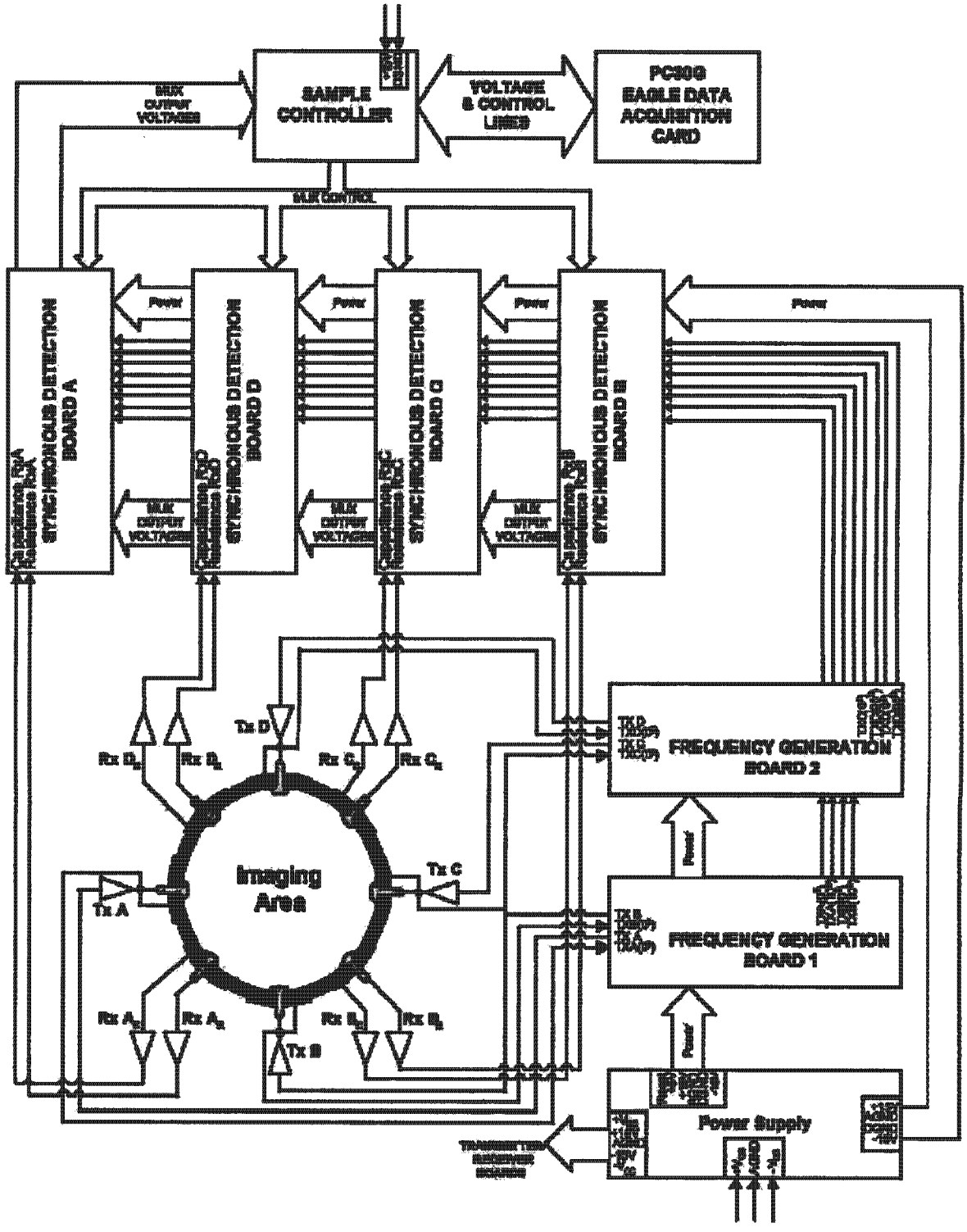
B.1 System overview

Figure B-1 is a system block diagram (extracted from [3]) of the measurement hardware as the author received it. The diagram shows the operation and interconnection of the various sub-systems of an 8-electrode, bimodal, FDM measurement system.

In figure B-1, the power supply regulates voltage and supplies power to all the other sub-systems. Each frequency generation board produces 4 voltage varying signals - 2 driving signals (to be injected into the rig) and their quadratures (to be used in capacitance demodulation). There are 2 of these, hence 8 signals are produced (4 injection signals and their quadratures). The injection signals are sent to the transmitter boards (TxA, TxB, TxC, TxD), where they are buffered, amplified and injected into the measurement space. Together with their quadratures, these signals are also sent to the demodulation boards - where they will be used as reference signals for arrays of synchronous detectors.

The receiver boards (RxA_c, RxA_r, RxB_c, RxB_r, RxC_c, RxC_r, RxD_c, RxD_r) amplify the current received by the capacitance and resistance electrodes by means of trans-impedance amplifiers, and send these signals to the demodulation boards. Note that each receiver electrode pair (resistance and capacitance) is sent to *one* demodulation board. Here (on each demodulation board), an array of synchronous detectors demodulate the strength of the various frequency signals received by the receiver electrode in question. If there are 4 transmitter frequencies injected into the rig, then the synchronous detection of 4 in-phase and 4 quadrature components will be performed by each board. Thus, for an 8-electrode (4 transmitters, 4 receivers) system, each demodulation board will produce 8 data points (synchronous detector output voltages) to be sampled. The entire system will produce $8 \times 4 = 32$ data points. The entire number of data points available from the system at one time is said to constitute a frame of data in EIT.

Clearly, the number of data points to be sampled by this system (32) is higher than the number of available input channels in most commercial data acquisition modules (typically 16), thus some form of multiplexing is necessary. Thus, each demodulation board is fitted with a 16-



140

Figure B-1: Giannopoulos' system block diagram.

channel multiplexer[1] - the 8 data points available per demodulation board are connected to the multiplexer input channels. In order for synchronisation to be achieved between sampling the data acquisition input channels and clocking the multiplexer address lines, a sample controller board is necessary. In short, this board operates by using an external counter to clock the multiplexer address lines.

B.2 Frequency generation boards

As received by the author on commencement of the project, there were 2 frequency generation boards. Each board generated 2 different frequency signals and their quadrature (4 signals in total). This section explains how Giannopoulos achieved this frequency generation, his documented circuit diagrams, the undocumented changes he made to the boards and the author's alterations.

B.2.1 Principles of operation

The way in which Giannopoulos generated the required frequencies is illustrated in figure B-2. Firstly, a PIC micro-controller was hard coded to output a time series of 8-bit digital outputs. This time series was structured to represent a 'choppy' sine-wave - the actual binary numbers output on the PIC's port and their decimal equivalents are shown in the figure. These binary numbers are then held using a latch-type setup of D-type flip flops, and fed in parallel to DACs where their values are converted to analogue currents. These currents are then offset and scaled by a transimpedance amplifier and fed to a 6-pole butterworth low-pass filter to produce a 'harmonically clean' sine-wave.

Note that the figure represents one frequency being generated - and its quadrature. Thus it represents one half of a frequency generation board. Note too that each channel (one frequency + its quadrature) is identical, except where different crystal's are required for different frequencies being generated by the PICs, and where different gains are required in each opamp stage of the butterworth filters. See [41, pp.247] for the required gains and passive component

[1]Even though only 8 data points need to be sampled per demodulation board, 16 channel multuplexors were used for extendability reasons - the extra channels would be necessary if more transmitter frequencies are introduced.

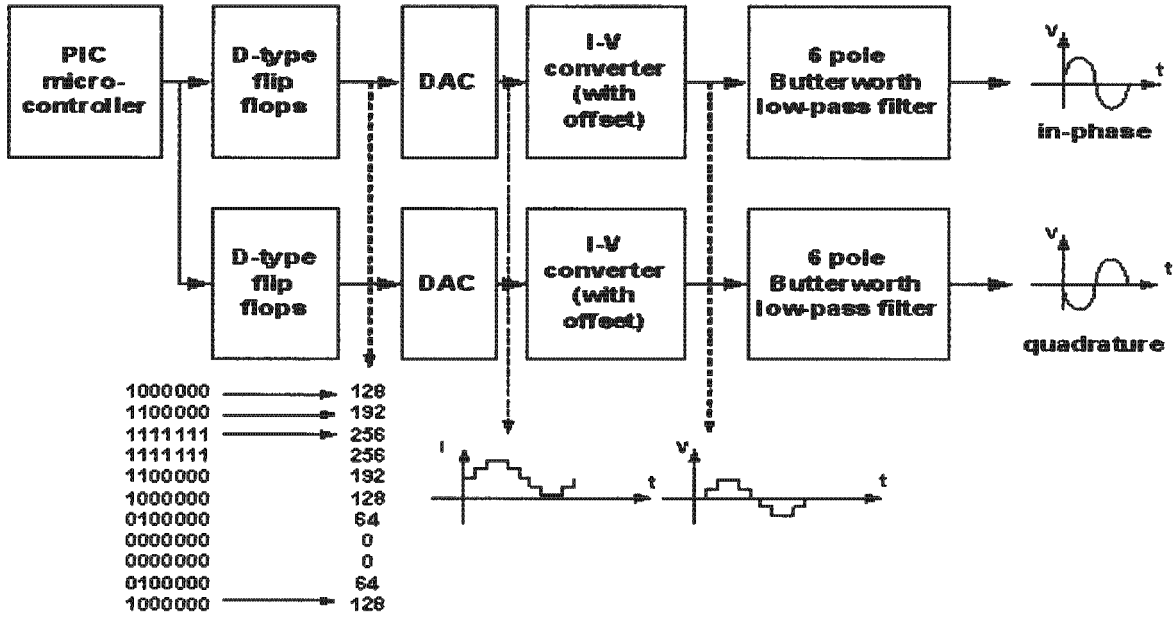


Figure B-2: A block diagram illustrating how frequency generation is acheived on the boards designed by Giannopoulos and used in the present system. An indication of the type of signals present throughout the chain is given (for the in-phase component only). Note that the diagram represents half of an actual board as it stands.

values.

The outputs of all channels are brought together on different pins (determined by jumper settings) of two 10-way molex type connectors on each board. The first and second two boards are then linked together by a 10-way ribbon cable bus, such that the first two frequency generation boards fill their 10 way bus with the signals: TxA, TxA_90, TxB, TxB_90, TxC, TxC_90, TxD, TxD_90. The remaining of the 10 lines are ground. These signals then get sent to the demodulation boards, where they are used as reference signals for the synchronous detector networks of receivers A, B, C and D. The other two frequency generation boards are linked in the same manner and fill their busses with the signals: TxE, TxE_90, TxF, TxF_90, TxG, TxG_90, TxH, TxH_90. This gets sent to the other half of the demodulation boards (previously unpopulated by Giannopoulos) for demodulation of receivers E, F, G and H.

B.2.2 A note on the shielding of cables

In Giannopoulos' system, recall that he was interested in obtaining bimodal data (i.e. resistance and capacitance data). Due to the tiny magnitudes of the signals involved with capacitance data, it was necessary to bootstrap (tie cable's shields to a buffered version of signal) any signal carrying cables. This was due to the fact that the stray capacitance of these cables would significantly increase the signal to noise ratio of capacitance readings obtained from the measurement system. It is for the same reason that transmitter and receiver boards were previously realised on separate substrates - so that these boards could be placed as close as possible to their associated electrodes, avoiding the use of long wires.

On Giannopoulos' boards, any necessary bootstrapping (i.e. on cables between frequency generation boards and transmitters) was performed on the board that the signals were sent to - the signal was buffered on the receiving board and tied to the cable's shield.

In the author's research, only unimodal (resistance) data is required, and hence bootstrapping was deemed unnecessary. Cable shields were thus grounded on both receiving and transmitting ends of cables.

B.2.3 Undocumented changes as performed by Giannopoulos

There were 3 changes carried out on the PCBs printed from their circuit diagrams in Appendix A. These changes involved the re-routing of signals and the addition of components onto the boards. Due to the hardy nature of PCBs, these alterations were sometimes destructive in nature (e.g. when tracks had to be cut or even gouged out of the substrate). The changes Giannopoulos made were laboriously traced out and redrawn by the author - often, a reasonable explanation as to why the changes were made could not be found. Nonetheless, the changes are described in terms of the functionality of the circuits below:

1. **Generation of offset voltages for the I→V converters:**

   Due to the fact that the DACs in each channel output an *offset* version of a 'choppy' sine-wave, before low pass filtering, and during current to voltage conversion, Giannopoulos performed level shifting of the signal. This was done by, instead of tieing the non-inverting input of the transimpedance amplifier to ground, creating a DC offset voltage and using this as the reference.

   In his documented circuit diagrams, the generation of these offset voltages was realised independently on each channel using a buffered voltage divider circuit. In reality, on the physical boards, this was not the case; instead, offset voltage references were created using an LM336 Z2.2 zener voltage reference IC - one offset reference voltage was realised on each physical PCB. This as opposed to a reference for each channel on each board (i.e. four per PCB). The single reference created per board was buffered and fed to all four channels on that board. As a result, the footprints and drilled holes in each channel of each PCB, designed to accept the previous voltage divider circuit, were left unpopulated. Instead, the voltage reference circuit was built (in a free hand manner) and raised off the board, using the holes designed for one of the originally intended voltage dividers per PCB. The reference was then fed to all the transimpedance amplifiers' non-inverting input via ordinary wire underneath the boards. A circuit diagram of how these reference voltages was created in reality appears in figure B-3.

2. **Bypassing of final RC network:**

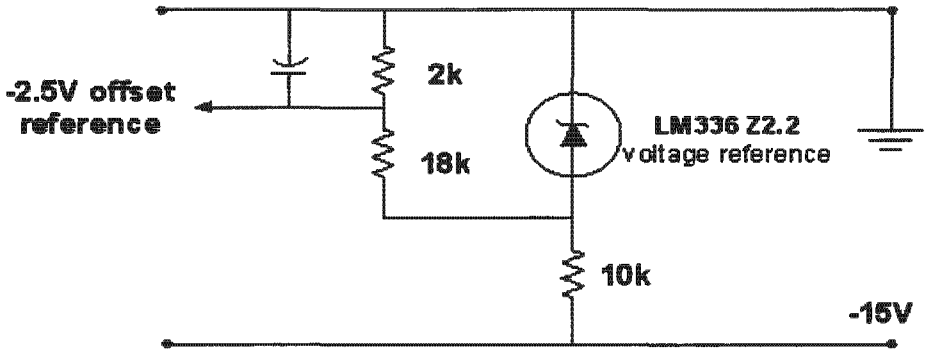


Figure B-3: A circuit diagram showing how Giannopoulos generated the offset reference voltage used to level shift the sinusoid.

One simple, passive sub-circuit which was not implemented in PCB format, was the final RC low pass filter of each channel. This was not really necessary as the signal has already gone through a 6 pole butterworth filter. Thus, the output was merely fed straight from the output of the butterworth filter to the transmitter amplifier boards.

3. **Alteration of cable shield bootstrapping:**

   For some reason, which the author was not able to ascertain, Giannopoulos added a pull down resistor to each of the 'bootstrapped'[2] cable shields which run between the frequency generation boards and final transmitter amplifier boards. The buffering required in bootstrapping or guarding of these cable shields was performed on the board the signal was being sent *to*. Thus when signals injected on the electrodes was sent back to the frequency generation boards to be placed on the 10-way reference signal bus, they were buffered on the frequency generation boards.

   The pull down resistor added by Giannopoulos was repeated by the author when populating new boards to ensure consistency with the rest of the system. The pull down resistor added by Giannopoulos is detailed in figure B-4.

[2] 'Bootstrapping' or 'guarding' is the process of buffering the transmitted signal and placing the buffered version on the cable shield.

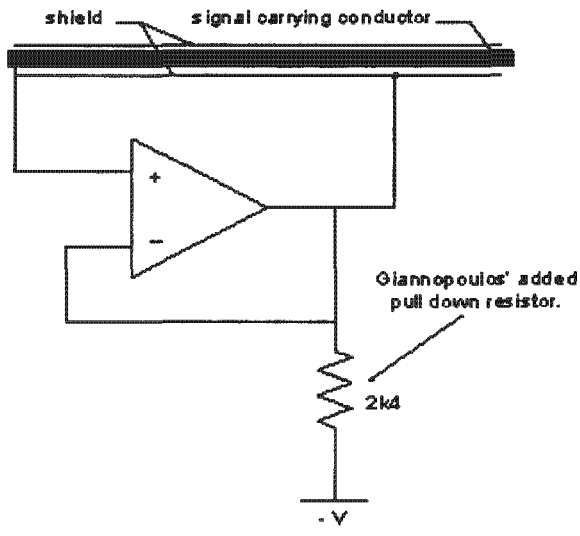


Figure B-4: A circuit diagram showing where Giannopoulos placed the pull down resistor.

B.2.4 Changes as performed by the author (present state of the hardware)

1. **Carrying through of Giannopoulos' changes to new boards:**

   As was mentioned in point 3 above, where changes were present in the boards used by Giannopoulos, they were carried through when populating new boards. This was done to ensure system integration and uniformity. Most of the changes Giannopoulos doctored the boards with were improvements (nonetheless messy ones) in any case.

2. **Grounding of cable shields:**

   Instead of bootstrapping cable shields, for reasons justified in section B.2.2, the author decided to, more simply and conveniently, ground them. Thus, whenever a cable shield enters a PCB in the new system, it is immediately soldered to the ground plane.

3. **Replacement of transimpedance opamp:**

   To convert the current output of the DACs to voltages, Giannopoulos had used a standard LM741 opamp. This opamp was upgraded to the pin equivalent LF351. The spec's are merely improved and should improve (although very slightly) signal fidelity.

B.3 Transmitter and receiver boards

B.3.1 Principles of operation

The amplification hardware for injection and measurement of signals on electrodes has been completely re-designed and built by the author for this project. The transmitter and receiver circuits are simple modules which use little more than an opamp and few passive components per electrode. The circuit diagrams provided in Appendix A should be sufficient to explain their operation and functionality. Transmitters use AD797 high power, ultra-low distortion, ultra-low noise opamps in the non-inverting mode. Receivers use LF411 low noise opamps as transimpedance amplifiers. A dual, vanilla, low noise opamp (LF355) is used to create two buffered versions of the main electrode signal. These are used to drive the guard electrodes. Where point electrodes are the required electrode choice, the main drivers (AD797s) should be used.

B.3.2 Changes from the previous boards

Although the basic circuit details remained similar to the previous amplifiers, decoupling and voltage regulation was realised differently. Due to the fact that multiple transmitters and receivers are built on a single substrate, the banks of capacitors previously applied across the incoming power on each amplifier board were reduced to one. 100nF ceramic decoupling capacitors were however placed close to power pins in the usual decoupling fashion. Furthermore, because lower currents were expected due to the smaller geometry of the rig, the receiver amplifiers' regulation was performed with a single 1 amp LM7812 (or LM7912 for negative voltages) at the power supply circuit. The transmitters' regulation was still implemented at each transmitter using LM78L12/LM79L12s. Thus, an additional two mandatory output capacitors (100uF) were added per transmitter channel as required by the regulators

As far as the amplifier circuit details are concerned, two changes were made: one to the receiver amplifier and one to the transmitter. Giannopoulos had previously used a buffer in the feedback loop of the receiver transimpedance amplifier to avoid opamp saturation with the high received currents in his larger rig. Due to the scaling down of the new system, this was scrapped. In the transmitter chain, the Zobel network placed between the previous amplification IC

(LM1875) and the electrode was not implemented. As far as the author could ascertain, Giannopoulos had merely put this there as it is recommended in the LM1875's datasheet. The LM1875, is however, designed for audio applications, which usually require a Zobel network for stability. As no inductance is present in an EIT load this was not implemented in the author's system.

B.4 Demodulation boards

B.4.1 Principles of operation

The principles of operation of the demodulation boards have been well documented in Teague's Ph.D. Thesis [4]. Basically, arrays of synchronous detectors - each with their own reference signal - are used per receiver. The number of synchronous detectors depends on the number of signals injected and the modality of the system (i.e. resistance data or resistance & capacitance data). The resistance between a receiver and a transmitter electrode, is then proportional to the DC output voltage of a synchronous detector circuit with that transmitter's signal used as a reference. The capacitance or permittivity between the two electrodes, is proportional to a synchronous detector's output voltage when supplied with the amplified signal from a receiver electrode, with that transmitter's 90° phase shifted signal used as a reference. Hence the need to generate, but not inject, the 90 degree phase shifted versions of signals on the frequency generation boards.

In the case of Giannopoulos' system, only 8 electrodes (4 transmitters and 4 receivers) were used and hence only 8 reference signals were ever fed to the demodulation boards (TxA, TxB, TxC, TxD and their quadratures). The way in which his demodulation boards were designed was such that one board was intended for one receiver electrode, and the maximum number of synchronous detectors able to be populated per board was designed to meet the requirements of a 16-electrode, bimodal system (i.e. 8 in-phase reference signals and 8 quadrature reference signals). The board would then demodulate a receiver electrodes received signal, to produce 16 readings - 8 resistance and 8 capacitance. This is why the sample controller board's counter is 4-bit (i.e. 16 counts) - so that it can switch through all 16 readings per demodulator board.

Of course Giannopoulos' system only used 4 transmitter electrodes and hence 4 receiver

electrodes. The demodulation boards designed however, were compatible with a 16-electrode system - thus, half of each board was left unpopulated.

B.4.2 Undocumented changes as performed by Giannopoulos

The only changes from the documented circuit diagrams performed by Giannopoulos, were the addition of pull down resistors on the bootstrapped received signal cables, and the bypassing of the final RC low pass filter. Similar changes had been performed on the frequency generation boards and are documented in earlier in this appendix see figure B-4.

B.4.3 Changes as performed by the author (present state of the hardware)

The main changes were discussed in the body of this document (section 5.5.5), but are documented again here for clarity. It has been emphasised that the demodulation PCBs designed by Giannopoulos were intended to service *one* electrode per board; thus, the amplified signal from *one* receiver electrode would be fed to each board. Thus, one board would have an array of 16 synchronous detectors. Each synchronous detector on the same demodulation board would be supplied with the incoming, amplified signal received by a receiver. The reference signals would comprise the 8 in-phase signals injected into the rig on each transmitter electrode (resistance data), and their quadratures (capacitance data).

Because the boards designed by Giannopoulos had a number of pitfalls and needed to be redesigned in future research, the author decided to use the unpopulated half of each demodulation board - originally intended for the demodulation of the 4 resistance and 4 capacitance signals which would be added with the upgrade of Giannopoulos' system from a 4-electrode to an 8-electrode rig. The unpopulated half was then populated and the routing of reference signals altered.

In the present system, one demodulation board services 2 receiver electrodes, but only demodulates resistance data. This is done by 'tricking' the synchronous detectors (originally supplied with 90° phase shifted reference signals) into receiving the additional frequencies added with more transmitter electrodes. A diagram illustrating this concept appears in figure B-5.

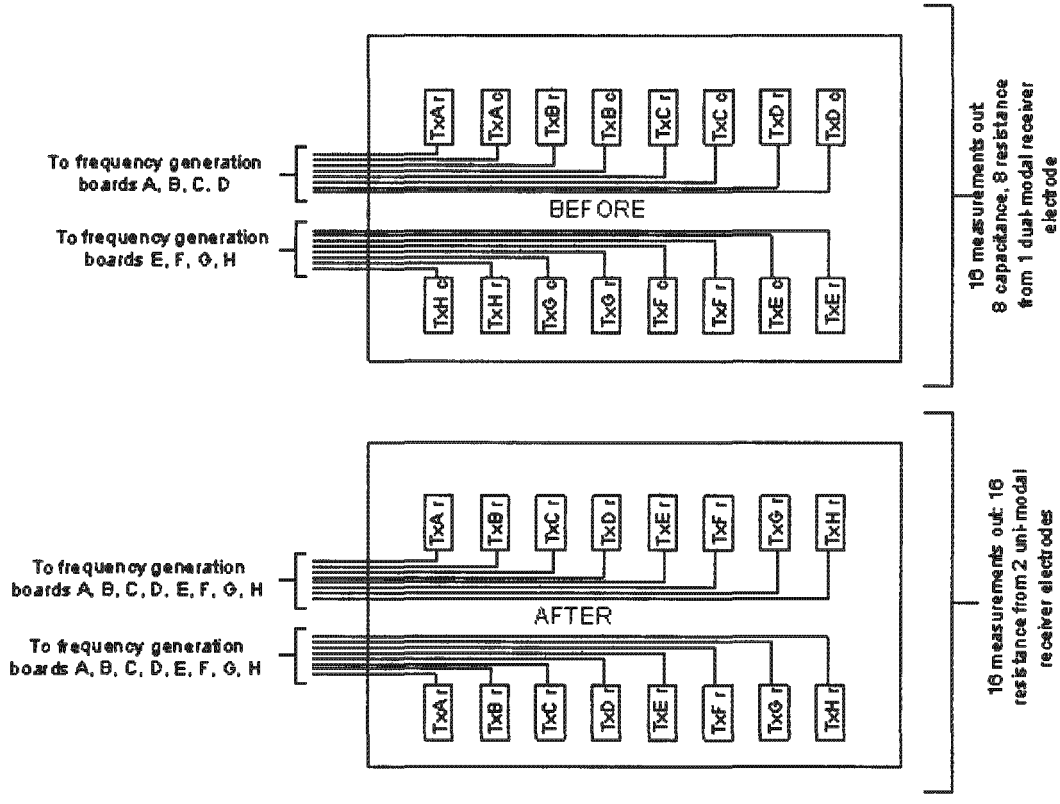


Figure B-5: A block diagram illustrating the differences between Giannopoulos' demodulation boards, and the new (altered) boards. The large blocks represent an actual PCB demodulation board as it exists in the system. The small blocks contained therein are synchronous detectors, demodulating the incoming signal with respect to their reference signal. Each synchronous detector is labelled so as to discribe its task e.g. "TxC c" means that this synchronous detector is demodulating the capacative component of the signal transmitted by transmitter C, and received by this demodulation board's receiver electrode. Note that the layout of the boards has not changed, merely the routing of synchronous detector's reference signals has.

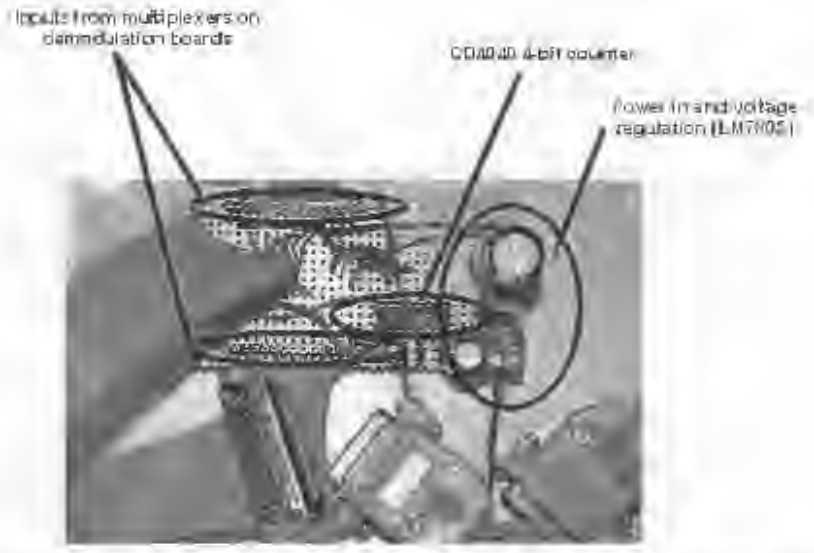


Figure B-6: A photograph of the sample controller board. The components have been labelled – note that there are enough inputs to accommodate a dual-plane, 16-electrode, dual-modal system (i.e. 32).

B.5 Sample controller board

The sample controller board and data acquisition hardware have been discussed in-depth in section 5.3.3. An annotated photograph of the board appears in figure B-6.

Appendix C

Code

While the collection of code which contributed to the results of this thesis (written by the author and others) is too copious to document fully, the more significant code fragments are presented in this appendix. All code is presented in its commented version - the comments (where occurring) should be used as an explanation of the code. The following sub-systems' code appears here:

1. Data acquisition - **C**
2. Kernel Ridge Regression - MATLAB
3. Finite Element Method (and meshing algorithm) - MATLAB
4. Simulated training data - MATLAB

C.1 Data acquisition

PROGRAM: contsoft

PURPOSE:
    Open Layers data acquisition example, shows
        continuous ADC operation.

FUNCTIONS:

    WinMain()   gets handle to ADC subsystem, configures ADC for
                        continuous operation, opens a dialog box to receive
                        window messages, and performs continuous operation
                        on the ADC.

    GetDriver()   callback to get board name and open board, opens
                        the first available Open Layers board.

    OutputBox()   dialog box to handle information and error window
                        messages from the ADC subsystem.

/*---- Comments --

  Modified for use by Tim Long 27 Aug 2004
  Further modified for use by Gareth Coldwell 2 May 2005

--*/

```
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <olmem.h>
#include <olerrors.h>
#include <oldaapi.h>
#include "resource.h"


#define SAMPLERATE  60000.0
#define CLOCKFREQ   3000.0
#define DUTYCYCLE   10.0


#define STRLEN 80           /* string size for general text manipulation   */
char strSTRLEN[];           /* global string for general text manipulation */
#define FRAMESIZE 256
```

```
/* Error handling macros */
#define SHOW_ERROR(ecode) MessageBox(HWND_DESKTOP,olDaGetErrorStr(ecode),
                  &str,STRLEN),"Error", MB_ICONEXCLAMATION | MB_OK);

#define CHECKERROR(ecode) if ((board.status = (ecode)) != OLNOERROR)\
                  {\
                  SHOW_ERROR(board.status);\
                  olDaReleaseDASS(board.hdass);\
                  olDaTerminate(board.hdrvr);\
                  return (UINT)NULL;}

#define CLOSEONERROR(ecode) if ((board.status = (ecode)) != OLNOERROR)\
                  {\
                  SHOW_ERROR(board.status);\
                  olDaReleaseDASS(board.hdass);\
                  olDaTerminate(board.hdrvr);\
                  EndDialog(hDlg, TRUE);\
                  return (TRUE);}



/* simple structure used with board */
typedef struct tag_board {
   HDEV  hdrvr;          /* device handle              */
   HDASS hdass;          /* sub system handle -> open adc       */
   HDASS hclock;          /* handle to clock subsytem */
   HDASS hdoutdas.        /*g handle to the digital output sub-system*/
   ECODE status;         /* board error status         */
   char name[STRLEN];    /* string for board name      */
   char entry[STRLEN];   /* string for board name      */
} BOARD;


typedef BOARD* LPBOARD;

static BOARD board;                                    // the board structure
static ULNG count = 0;                                 // the number of boards
static UINT numberofbuffers = 20;                      // the number of buffers
static UINT framestoaverage=10;                        // the number of frames to average
static FILE *file = NULL;                              // pointer to the file we're saving to
static BOOL bStopped = FALSE;                          // indicates whether the system has been stopp
ed
static UINT framestocapture = 100;                     // the number of frames to capture
static UINT framescaptured;                            // the number of frames captured already
static BOOL framemask[FRAMESIZE];                      // the framemask
char filename[FILENAME_MAX];                           // filename to save to
char maskfilename[FILENAME_MAX];                       // the mask filename
static BOOL bGotMask = FALSE;
static BOOL bGotFilename = FALSE;

static OPENFILENAME ofnFileStruct;                     // save file name structure (needed to give a
nelon box)

BOOL CALLBACK
GetDriver( LPSTR lpszName, LPSTR lpszEntry, LPARAM lParam )
/*
this is a callback function of olDaEnumBoards. it gets the
strings of the open layers board and attempts to initialize
the board.  if successful, enumeration is halted.
*/
{
    LPBOARD lpboard = (LPBOARD)(LPVOID)lParam;

    /* fill in board strings */
    lstrcpyn(lpboard->name,lpszName,STRLEN);
    lstrcpyn(lpboard->entry,lpszEntry,STRLEN);

    /* try to open board */
    lpboard->status = olDaInitialize(lpszName,&lpboard->hdrvr);
    if(lpboard->hdrvr != NULL)
        return FALSE;           /* false to stop enumerating */
    else
```

```
        return TRUE;            // true to continue
}

VOID Stop()
{
    HBUF  hBuffer = NULL;
    UINT i=0;
   /*
     get the input buffers from the ADC subsystem and
     free the input buffers
   */

   CHECKERROR (olDaFlushBuffers(board.hDass));

   for (i=0;i<numberOfBuffers;i++)
   {
     CHECKERROR (olDaGetBuffer(board.hDass, &hBuffer));
     CHECKERROR (olDaFreeBuffer(hBuffer));
   }

   if(file)
   {
       fclose(file);
   }

   return (UINT)NULL;


BOOL CALLBACK
InputDlgProc( HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam
/*
This function processes messages for "Input" dialog box
*/
{
    UINT    i=0,j=0,k=0;                    // counters
    DBL     min=0,max=0;                    // min and max limits voltage inputs
    DBL     volts[FRAMESIZE];               // buffer storing the voltages for one (averaged) frame
    DLNG    samples;                        // the number of samples in the buffer
    UINT    encoding=0,resolution=0;        // encoding type(bin or 2's comp), resolution (16 bits)
    HBUF    hBuffer = NULL;                 // handle to a buffer to take off done queue
    PWORD   pBuffer = NULL;                 // handle to the measurements in the buffer
    char    WindowTitle[128];               // string holding the window title
    DASSINFO ssInfo;                        // subsystem info struct
    DWORD   errorValue;                     // place to save error codes to
    HWND    hDlgItem;                       // handle to a dialog item, used to change dialog control
parameters
    UINT    buffersize;                     // the size of one buffer
    UINT    sampleSize;                     // the size (in bytes) of each sample in a buffer
    UINT    divisor = FramesToAverage;

    //  these variables used for the digital out subsystem
    BOOL    value;
    UINT    channel_dout=0;
    DBL     data_dout=1.0;


    switch (message)
    {
    case WM_INITDIALOG:    /* message: initialize dialog box */
        {
            /* set the title to the board name */

            olDaGetDASSInfo(board.hDass,&ssInfo);

            sprintf(WindowTitle, "Tomography Data Capture");

            SetWindowText(hDlg,WindowTitle);                    // set window title

            hDlgItem = GetDlgItem(hDlg, IDC_START_AD);          // get handle to the start button
            EnableWindow(hDlgItem, FALSE);                      // disable start button
```

```
            hDlgItem = GetDlgItem(hDlg, IDC_FRAMES2AVE );      // get handle to frames to ave edit bo
x
            sprintf(str, "%d", framestoaverage);               // set text to the number of frames to
 ave
            SetWindowText(hDlgitem, str);

            hDlgitem = GetDlgItem(hDlg, IDC_FRAMESTOCAPTURE);// get handle to frames to capture ed
it box
            sprintf(str, "%d", framestocapture);               // set text to the number of frames to
 capture
            SetWindowText(hDlgitem, str);

            /* set window handle */
            CLOSEONERROR (olDaSetWndHandle(board.hdass, hDlg,(UINT)NULL));

            // configure AD subsystem
            CHECKERROR (olDaConfig(board.hdass));

            return (TRUE);    /* Did process a message */
        }
    case OLDA_WM_BUFFER_REUSED:    /* message: buffer reused  */
            break;

    case OLDA_WM_BUFFER_DONE:      /* message: buffer done  */
        {
            /* get buffer off done list */
            CHECKERROR (olDaGetBuffer(board.hdass, &hBuffer));

            /* if there is a buffer */
            if( hBuffer )
            {
                /* get sub system information for code/volts conversion */
                CLOSEONERROR (olDaGetRange(board.hdass,&max,&min));
                CLOSEONERROR (olDaGetEncoding(board.hdass,&encoding));
                CLOSEONERROR (olDaGetResolution(board.hdass,&resolution));

                /* get max samples in input buffer */
                CLOSEONERROR (olDmGetValidSamples( hBuffer, &samples ) );

                /* get pointer to the buffer */
                CLOSEONERROR (olDmGetBufferPtr( hBuffer,(LPVOID*)&pBuffer));


                if(framestoaverage<10)         // if we're only averaging a few frames, the buffer si
ze
                {                              // is 10 time larger than normal
                    for(i=0;i<10;i++)          // so we must make sure that we get everything
                    {
                        // clear voltage buffer
                        ZeroMemory(volts, FRAMESIZE*sizeof(DBL));
                        for(j=0;j<FRAMESIZE*framestoaverage;j++)           // copy all the samples to
 a buffer
                        {
                            volts[j%FRAMESIZE] += ((float)max-(float)min)/(1L<<resolution)*pBuffer
[k] + (float)min;
                            k++;
                        }
                        for(j=0;j<FRAMESIZE;j++)
                        {
                            if(framemask[j])
                                fprintf(file, "%lf, ", volts[j]/framestoaverage);
                        }

                        fprintf(file, "\n");
                        framescaptured++;

                    }
                }
                else
                {
                    // clear voltage buffer
                    ZeroMemory(volts, FRAMESIZE*sizeof(DBL));
```

```
                    for(j=0;j<samples;j++)      // copy all the samples to a buffer
                    {
                        volts[j%FRAMESIZE] += ((float)max-(float)min)*(ll<<resolution*pBuffer[j]
+ (float)min;
                    }
                    for(j=0;j<FRAMESIZE;j++)
                    {
                        if(framemask[j])
                            fprintf(file, "%f, ", volts[j]/framestoaverage);
                    }
                    fprintf(file, "\n");
                    framescaptured++;
                }
                /* put buffer back to ready list */
                CHECKERROR (olDaPutBuffer(board.hdaqs, hBuffer));

                /*  convert value to volts */
                for(j = 0;j<16;j++)
                {
                    /* display value */
                    sprintf(str,"%.3f Volts",volts[j]/framestoaverage);
                    SetDlgItemText (hDlg, IDD_VALUE0+j, str);
                }


                if(framescaptured >= framestocapture)
                {
                    CLOSECONERROR (olDaStop(board.hdaqs))       // kill AD process
                    CLOSECONERROR (olDaStop(board.hclock))      // kill clock process

                    bStopped = TRUE;
                }

                if(bStopped)                    // if we've stopped the data capture
                {                               // close the file
                    bldn();                     // and free buffers
                }

            }

            return (TRUE);    /* Did process a message */
        }
    case OLDA_WM_QUEUE_DONE:
    case OLDA_WM_QUEUE_STOPPED:
        {
            /* using wrap multiple - if these message are received */
            /* acquisition has stopped.                            */

            EndDialog(hDlg, TRUE);

            return (TRUE);    /* Did process a message */
        }

    case OLDA_WM_TRIGGER_ERROR:
        {
            /* Process trigger error message */
            MessageBox(hDlg,"Trigger error: acquisition stopped", "Error", MB_ICONEXCLAMATION | MB
_OK);

            EndDialog(hDlg, TRUE);

            return (TRUE);    /* Did process a message */
        }

    case OLDA_WM_OVERRUN_ERROR:
        {
            /* Process underrun error message */

            MessageBox(hDlg,"Input overrun error: acquisition stopped", "Error", MB_ICONEXCLAMATI
N | MB_OK);

            EndDialog(hDlg, TRUE);    /* Did process a message */
```

```
            return (TRUE);    /* Did process a message */
        }

    case WM_COMMAND:        /* message: received a command */
        {
        switch (LOWORD(wParam))
        {

        case IDC_BROWSE_BTN:
            {
                filename[0] = 0;

                ZeroMemory(&savefileStruct, sizeof(OPENFILENAME));
                savefileStruct.lStructSize = sizeof(OPENFILENAME);        // set size
                savefileStruct.hwndOwner = hDlg;                          // set owner
                savefileStruct.lpstrFilter = "CSV files\0*.csv\0All files\0*.*\0\0";
    // set filter text
                savefileStruct.nFilterIndex = 1;                          // set to the first fi
lter

                savefileStruct.lpstrFile = filename;
                savefileStruct.nMaxFile = FILENAME_MAX;
                savefileStruct.lpstrFileTitle = NULL;                     // pointer the filename
                savefileStruct.nMaxFileTitle = 0;                         // max length of filenam
e buffer

                savefileStruct.lpstrDefExt = "csv";                       // pointer to default fil
e name

                if(GetSaveFileName(&savefileStruct))
                {
                    hDlgItem = GetDlgItem(hDlg, IDC_FILENAMEBOX);
                    SetWindowText(hDlgItem, savefileStruct.lpstrFile);
                    bGotFilename = TRUE;
                    if(bGotMask && bGotFilename)
                    {
                        hDlgItem = GetDlgItem(hDlg, IDC_START_B);
                        EnableWindow(hDlgItem, TRUE);
                    }
                }
                else
                {
                    errorvalue = CommDlgExtendedError();
                    MessageBox(hDlg, "Can't savefile", "Error", MB_OK);
                }

                return (TRUE);
            }

        case IDC_BROWSE_BTN2:
            {
                filename[0] = 0;

                ZeroMemory(&savefileStruct, sizeof(OPENFILENAME));
                savefileStruct.lStructSize = sizeof(OPENFILENAME);        // set size
                savefileStruct.hwndOwner = hDlg;                          // set owner
                savefileStruct.lpstrFilter = "Mask files\0*.msk\0All files\0*.*\0\0";
    // set filter text
                savefileStruct.nFilterIndex = 1;                          // set to the first fi
lter

                savefileStruct.lpstrFile = maskfilename;
                savefileStruct.nMaxFile = FILENAME_MAX;
                savefileStruct.lpstrFileTitle = NULL;                     // pointer the filename
                savefileStruct.nMaxFileTitle = 0;                         // max length of filenam
e buffer

                savefileStruct.lpstrDefExt = "msk";                       // pointer to default fil
e name

                savefileStruct.Flags = OFN_EXPLORER;

                if(GetOpenFileName(&savefileStruct))
                {
                    hDlgItem = GetDlgItem(hDlg, IDC_MASKNAMEBOX);
                    SetWindowText(hDlgItem, savefileStruct.lpstrFile);
                    bGotMask = TRUE;
                    if(bGotMask && bGotFilename)
```

```
                hDlgItem = GetDlgItem(hDlg, IDC_START_AD);
                EnableWindow(hDlgItem, TRUE);

            }
            else
            {
                errorvalue = CommDlgExtendedError();
                MessageBox(hDlg, "Can't Get File", "Error", MB_OK
            }

            return (TRUE);
        }
    case IDC_START_AD:
        {
            // open the mask file
            GetDlgItemText(hDlg, IDC_MASKNAMEBOX, maskfilename, FILENAME_MAX);

            file = fopen(maskfilename, "rt");
            if(file == NULL)
            {
                sprintf(str, "Cannot open %s", maskfilename);
                MessageBox(hDlg, str, "File Error", MB_ICONEXCLAMATION);
                return (FALSE);
            }

            for(i=0; i<FRAMESIZE;)++)
            {
                fscanf(file, "%d", &framemask[i]);
            }

            fclose(file);
            file = NULL;


//          for(i=0;i<FRAMESIZE;i++)
//              framemask[i] = TRUE;

            // open the file
            GetDlgItemText(hDlg, IDC_FILENAMEBOX, filename, FILENAME_MAX);

            file = fopen(filename, "wt");
            if(file == NULL)
            {
                sprintf(str, "Cannot open %s", filename);
                MessageBox(hDlg, str, "File Error", MB_ICONEXCLAMATION);
                return (FALSE);
            }

            // find out how many frames to average
            GetDlgItemText(hDlg, IDC_FRAMES2AVE, str, STRLEN);
            framestoaverage = atoi(str);

            GetDlgItemText(hDlg, IDC_FRAMESTOCAPTURE, str, STRLEN);
            framestocapture = atoi(str);

            if(framestoaverage < 10)
            {
                buffersize = 256 * framestoaverage * 10;
            }
            else
            {
                buffersize = 256 * framestoaverage;
            }

            // 200 fps: buffers cycle 1/s
            numberofbuffers = 200/framestoaverage;

            // find out if we're using 2 bytes or 4
            CHECKERROR(olDaGetResolution(board,bdBase,&samplesize));
            if (samplesize > 16)
                samplesize=4; // bytes.,// e.g. 24 bits = 4 bytes
            else
```

```
                samplesize=(
                                                    // e.g. if 12 bits = 2 bytes

                // allocate buffers
                for (i=0;i<numberofbuffers;i++)
                {
                    CHECKERROR (olDmAllocBuffer(0,0,(CLPC) &buffersize,&samplesize,&number));
                    CHECKERROR (olDaPutBuffer(board->hDass, &buffer));
                }


                // start AD sub system
                CLOSEONERROR (olDaStart(board->hDass));

                // start trigger subsystem
                CLOSEONERROR (olDaStart(board->hClock));

                hDlgItem = GetDlgItem(hDlg, IDC_START_AD);
                EnableWindow(hDlgItem, FALSE);                  // disable start button
                hDlgItem = GetDlgItem(hDlg, IDC_STOP_ADC);
                EnableWindow(hDlgItem, TRUE);                   // enable stop button

                bStopped = FALSE;

                return (TRUE);
            }
        case IDC_STOP_ADC:
            {

                CLOSEONERROR (olDaStop(board->hDass));          // kill AD process
                CLOSEONERROR (olDaStop(board->hClock));         // kill clock process

                bStopped = TRUE;


                return (TRUE);                                  // processed the message
            }
        case IDOK:
        case IDCANCEL:
            {
//              CLOSEONERROR (olDaAbort(board->hDass));         // kill AD process
//              CLOSEONERROR (olDaAbort(board->hClock));        // kill clock process

//              Stop();                                         // close file and free buffers
                EndDialog(hDlg, TRUE);                          // close the dialog box
                return (TRUE);
            }
        default:
            break;
        }

    }

default:
    break;
}
return (FALSE);                 /* Didn't process a message */
}


int WINAPI
WinMain( HINSTANCE hInstance,
         HINSTANCE hPrevInstance,
         LPSTR lpCmdLine,
         int nCmdShow )
/*-+
This is our apps entry point.  It calls the open layers functions
to execute the required operation
--*/
{
    DBL  samplerate = SAMPLERATE;
    DBL  clockfreq = CLOCKFREQ;
    DBL  dutycycle = DUTYCYCLE;
```

```
UINT ema,gainsup, maxcnt[/*/]
UINT i;

UINT channel = 0;
UINT numberADs = 0;
UINT currentAD = 0;
DBL gain = 1.0;
HBUF  hBuffer = NULL;
ECODE ec=OLNOERROR;


UINT temp =0;
char text[256];

BOOL value;
UINT channel_dout=0;
DBL  gain_dout=1.0;  /*g*/



 // Clear savefile struct
 ZeroMemory(&savefileStruct, sizeof(OPENFILENAME));

board.hdrvr = NULL;

 // Get cmdline board
 if (lpCmdLine[0]!=0)
 {
  CHECKERROR( olDaInitialize(lpCmdLine,&board.hdrvr) );
  strcpy(board.name,lpCmdLine);
 }
 else
 {
/* Get first available Open Layers board */
CHECKERROR (olDaEnumBoards(GetDriver,(LPARAM)(LPBOARD)&board));
 }
/* check for error within callback function */

CHECKERROR (board.status);

/* check for NULL driver handle - means no boards */

if (board.hdrvr == NULL){
   MessageBox(HWND_DESKTOP, " No Open Layer boards!!!", "Error",
         MB_ICONEXCLAMATION | MB_OK);
   return ((UINT)NULL);
}

/* Get handle to first clock/counter sub system */
 CHECKERROR (olDaGetDASS(board.hdrvr, OLSS_CT, 0, &board.hdass));



/* get handle to first available ADC sub system */
CHECKERROR(olDaGetDevCaps(board.hdrvr,OLDC_ADELEMENTS,&numberADs));

 while(1)    // Enumerate through all the A/D subsystems and try and select the first availabl
one...
 {
       ec=olDaGetDASS(board.hdrvr,OLSS_AD,currentAD,&board.hdass);
       if (ec!=OLNOERROR)
       {
       // busy subsys etc...
       currentAD++;
       if (currentAD>numberADs)
       {
           MessageBox(HWND_DESKTOP,"No Available A/D Subsystems!","Error",MB_ICONEXCLAMATION|MB_
OK);
             return ((UINT)NULL);
       }

       }
       else
```

```
        break;

    }

    //
       Set up the ADC - multiple wrap so we can get buffer reload
       window messages.
    */

    CHECKERROR (olDaGetSSCaps(board.hDass,OLSSC_MAXTHROUGHPUT,&samplefreq));
    CHECKERROR (olDaGetSSCaps(board.hDass,OLSSC_NUMDMACHANS,&dma));
    CHECKERROR (olDaGetSSCaps(board.hDass,OLSSC_SUP_PROGRAMGAIN,&gainsup));
    //?
    CHECKERROR (olDaGetSSCaps(board.hDass,OLSSC_MAX_SIMULTSOCKET_VALUE,&maxout));
    dma  = min (1, dma);              // try for one dma channel    */

     if(samplefreq < SAMPLEFREQ)
     {
         sprintf(str, "The selected sample frequency isn't supported.\nSetting the frequency to %.2
f", samplefreq);
         MessageBox(HWND_DESKTOP, str, "Error", MB_ICONEXCLAMATION);

     else
         samplefreq = min (SAMPLEFREQ, samplefreq);


    // setup channels
    CHECKERROR (olDaSetChannelType(board.hDass, 0L, CHNT_SINGLEENDED));    // set channels to be sing
le ended

    // setup data acq
    CHECKERROR (olDaSetDataFlow(board.hDass,OL_DF_CONTINUOUS));
    CHECKERROR (olDaSetClockFreq(&dev)board.hDass,samplefreq));
    CHECKERROR (olDaSetClockSource(board.hDass, OL_CLK_INTERNAL));

    // setup retriggering
    CHECKERROR (olDaSetTriggeredScanUnit(board.hDass, TRUE));              // Enable triggered scan
    CHECKERROR (olDaSetRetriggerMode(board.hDass, OL_RETRIGGER_EXTRA));  // set retrigger to be ex
ternal
    CHECKERROR (olDaSetMultiScanCount(board.hDass, 1));                    // only do one scan per tr
igger
    CHECKERROR (olDaSetRetrigger(board.hDass, DT_TRIG_RATE));              // set external source

//    CHECKERROR (olDaSetRetriggerFrequency(board.hDass, retriggerfreq));     // set the retrigger f
req


    // setup clock
    CHECKERROR (olDaSetClockMode(board.hClock, OL_CLOCK_RATE));            // set the clock mode to r
ate
    CHECKERROR (olDaSetClockSource(board.hClock, 0, OL_CLK_INTERNAL));     // set the clock source as
 internal
    CHECKERROR (olDaSetClockFrequency(board.hClock, clockfreq));
    CHECKERROR (olDaSetPulseType(board.hClock, OL_PLS_HIGHTOLOW));         // set pulse low to high
    CHECKERROR (olDaSetPulseWidth(board.hClock, dutycycle));               // set duty cycle to be 50
%
    CHECKERROR (olDaSetGateType(board.hClock, OL_GATE_NONE));              // set the gate type


    // setup buffers
    CHECKERROR (olDaSetWrapMode(board.hDass,OL_WRP_MULTIPLE));               // wrap mode single
    CHECKERROR (olDaSetDmaUsage(board.hDass,dma));

    //g enable dynamic digital output
    CHECKERROR (olDaSetSynchronousDigitalIOUsage(board.hDass, TRUE));


    for(i=0; i<16; i++)
    {
        CHECKERROR (olDaSetChannelListEntry(board.hDass,i,i));             // setup the input list

        CHECKERROR (olDaSetGainListEntry(board.hDass,i,gain));             // set gain as 1 at the be
ginning
```

```
        //g
        if (j != 0)
        {
         CHECKERROR (olDaSetDigitalIOListEntry(board.hDass,j,0));
        }
        else
        {
            CHECKERROR (olDaSetDigitalIOListEntry(board.hDass,j,mask));
        }
        //j
    }

    CHECKERROR (olDaGetSSCaps(board.hDass, OLSSC_CHLDEPTH, &temp));
    sprintf(text, "Supported channel depth is %d", temp);
//  MessageBox(GWKD_DESKTOP, text, NULL, MB_OK);
   CHECKERROR (olDaSetChannelListSize(board.hDass,16));                    // get list is size
}

   // config A/D subsystem
   CHECKERROR (olDaConfig(board.hDass));

   // configure xxto subsystem
   CHECKERROR (olDaConfig(board.hClock));

//g*****************************************************************
      /* get handle to DOUT sch system */

   CHECKERROR (olDaGetDASS(board.hDrvr,OLSS_DOUT,0,&board.hDoutss));

   /* set subsystem for single value operation */

   CHECKERROR (olDaSetDataFlow(board.hDoutss,OL_DF_SINGLEVALUE));
   CHECKERROR (olDaConfig(board.hDoutss));

   //Enable external counter reset
   value=1;
   CHECKERROR (olDaPutSingleValue(board.hDoutss,value,channel_dout,gain_dout));

//g*****************************************************************



   /*
      use a dialog box to collect information and error
      messages from the subsystem
   */


   DialogBox(hInstance, (LPCSTR)DLG_INPUTBOX, HWND_DESKTOP, inputDlg);


   /* release the subsystem and the board */

   CHECKERROR (olDaReleaseDASS(board.hClock));      // release the clock ss
   CHECKERROR (olDaReleaseDASS(board.hDass));
   CHECKERROR (olDaReleaseDASS(board.hDoutss));// release the A/D ss
   CHECKERROR (olDaTerminate(board.hDrvr));

   /* all done - return */

   return (ULONG)NULL;
}
```

C.2 Kernel Ridge Regression

krr.m

```
% extracted from Prof. Greene's "Neural, Fuzzy and Evolving Systems" - a cour
se taught in the Department of Electrical Engineering, University of Cape Tow
n, 2003.

function [yte]=krr(Xtr,ttr,sigma,gamma,Xte)

D2=dist2(Xtr,Xtr);
K=exp(-D2/(2*sigma^2));
K=K+gamma*eye(size(K));
alpha=K\ttr;
D2=dist2(Xte,Xtr);
K=exp(-D2/(2*sigma^2));
yte=K*alpha
```

krrtraintest.m

```
% extracted from Prof. Greene's "Neural, Fuzzy and Evolving Systems" - a cour
se taught in the Department of Electrical Engineering, University of Cape Tow
n, 2003.

function err=krrtraintest(Xtr,ttr,Xte,tte,sigma,gamma)

D2=dist2(Xtr,Xtr);
K=exp(-D2/(2*sigma^2));
K=K+gamma*eye(size(K));
alpha=K\ttr;
D2=dist2(Xte,Xtr);
K=exp(-D2/(2*sigma^2));
yte=K*alpha;
err=mean((tte-yte).^2);
```

normal_svd.m

```
function [U, S, V, Dnorm, meanD, stdD] = normal_svd(D, cutoff)

% Written by G Goldswain June 2004
%
% [U, S, V, Dnorm, meanD, stdD] = normal_svd(D, cutoff)
%
% Normalise the input data matrix and perform a singular value decomposition
% on it.  The returned S matrix will have all singular values which contribut
e
% less than 'cutoff'% the data's total variance zeroed.  U and V are the usua
l
% remaining svd matrices.
% Dnorm is the normalised Data matrix where meanD and stdD are the columnwise
 mean
% and standard deviation of D.
% NB to note is that this function expects the input matrix to be the trainin
g
% data (X) concatenated with the target values (t).

%r=# of rows (instances) of the input data matrix (D)
r=size(D,1);
%c=# of columns (features + 1) of the input data matrix (D)
c=size(D,2);

%mean/stdX is a row vector with each column containing the mean/std dev
%of the input matrix's columns (features)
meanD = mean(D,1);
stdD = std(D);

e=ones(r,1);
%subtract the mean and divide by the std dev of each column
Dnorm = (D - e*meanD);
Dnorm = Dnorm ./ (e*(stdD + (stdD==0) ));

%perform a compact svd all rows and all but the last column (which is reserve
d
%for target values)
[U S V]=svd(Dnorm(:, 1:c-1),0);
total=trace(S);
diagS=diag(S);

for i = 1:length(diagS)
```

```
    contrib=(diagS(i)/total)*100;
    if contrib < cutoff
       S(i,i)=0;
    end
end
```

scansg10fold.m

```
%Written by Gareth Goldswain, June 2005.
%Used to perform a grid search for optimum tuning parameters of KRR.

function [E, sig, gam, bestsig, bestgam] =scansg10fold(X,t)
sig=[10^-1.5 10^-1 10^-0.5 10^0 10^0.5 10^1 20 10^1.5 10^2 10^2.5 10^3];
gam=[10^-6 10^-5 10^-4 10^-3 10^-2 10^-1 10^0 10^1 10^2 10^3];

E=[];
for s=1:length(sig)
    for g=1:length(gam)
        ss=sig(s);
        gg=gam(g);
        e=tenfoldkrr(X,t,ss,gg);
        E(s,g)=e
    end
end

[r,c]=find( E==min( min(E) ) );
bestsig=sig(r);
bestgam=gam(c);
```

tenfoldkrr.m

```
% extracted from Prof. Greene's "Neural, Fuzzy and Evolving Systems" - a cour
se taught in the Department of Electrical Engineering, University of Cape Tow
n, 2003.

function E=tenfoldkrr(X,t,s,g)

D=[X,t];
[r,c]=size(X);
n=round(0.9*r);
Err=[];

for j=1:10
    X1=D(1:n,1:c);
    t1=D(1:n,c+1);
    Q1=D(n+1:r,1:c);
    y1=D(n+1:r,c+1);

    err=krrtraintest(X1,t1,Q1,y1,s,g);
    Err=[Err,err];

    D=[[Q1;X1],[t1;y1]];
end

E=mean(Err);
```

DIST2.m

```
function n2 = dist2(x, c)
%DIST2  Calculates squared distance between two sets of points.
%
%   Description
%   D = DIST2(X, C) takes two matrices of vectors and calculates the
%   squared Euclidean distance between them.  Both matrices must be of
%   the same column dimension.  If X has M rows and N columns, and C has
%   L rows and N columns, then the result has M rows and L columns.  The
%   I, Jth entry is the  squared distance from the Ith row of X to the
%   Jth row of C.
%
%   See also
%   GMMACTIV, KMEANS, RBFFWD
%

%   Copyright (c) Ian T Nabney (1996-2001)

[ndata, dimx] = size(x);
[ncentres, dimc] = size(c);
if dimx ~= dimc
    error('Data dimension does not match dimension of centres')
end

n2 = (ones(ncentres, 1) * sum((x.^2)', 1))' + ...
  ones(ndata, 1) * sum((c.^2)',1) - ...
  2.*(x*(c'));

% Rounding errors occasionally cause negative entries in n2
if any(any(n2<0))
  n2(n2<0) = 0;
end
```

C.3 Finite Element Method (and meshing algorithm)

```
function [sigma, ne] = elmtprop
% written by S. Herholdt 2004.
% modified for use by G. Goldswain 2005.
%
%   THIS FUNCTION IS WHERE DATA SPECIFIC TO EACH PROBLEM IS ENTERED.
%   THE OTHER FUNCTIONS ARE NOT MEANT TO BE CHANGED, UNLESS THE TYPE OF PROBL
EM COMPLETELY CHANGES -
%   - IF THAT HAPPENS,NEW CASES NEED TO BE ADDED TO THE VARIOUS FUNCTIONS
%
% This function returns the element properties. N (number of nodes per elemen
t), gauss_order, no_elm
% (number of elements), no_nodes (number of nodes), p, t, rxtx (boundary node
 information; see
% outering.m) are all globals.
%
% [sigma, ne] = elmtprop
%
% returns in ne the number of element edges per electrode.  This is used by s
umnodes.m
% returns in sigma a vector containing each elements conductivity.  This cond
uctivity vector is created
% in elmtprop as a homogeonous medium.

% Data for different resolution meshes has been precomputed and saved.  This
function  loads the data
% each time it is called instead of recalculating it each time.  To load diff
erent mesh data, uncomment
% the desired lines of code below.

global N gauss_order no_elm no_nodes p t rxtx



% 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.
%     {Siggy forgot to mention here that there actually exists another row in
%      his p_matrix - it contains the projection number}
%
%     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 th
e case of
%     conductivity.


% CURRENT EXPERIMENT IS BASED ON DATA BELOW  (DATE:04 April 2005 - G.Goldswai
n)

N=3;
% load mesh data for a mesh with "mesh_()" elements "ne()" elements per elect
rode/electrode gap
% and "nr()" rings of elements
%  load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_70_ne1_nr3
%  ne=1;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_108_ne1_nr4
% ne=1;
%load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_158_ne1_nr5
%ne=1;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_252_ne2_nr6
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_328_ne2_nr7
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_416_ne2_nr8
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_516_ne2_nr9
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_662_ne3_nr10
```

```
% ne=3;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_328_ne2_nr7
load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_1416_ne3_nr15
ne=3;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_2516_ne4_nr20
% ne=4;

%pad the t matrix's 4th row (subdomain number according to Matlab) with 0's
dummy  = zeros(1,length(t));
t=[t;dummy];
%add another row to t.  This row contains the conductivity value for each ele
ment
cond=ones(1,length(t));
cond=cond*0.001;
t=[t;cond];

%            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 [a_soln,F_soln, rxtx, ne] = fem(varargin)

% Written by Siggy Herholdt, 2004
% Modified for use by Gareth Goldswain, 2005
%
% [a_soln,F_soln, rxtx, ne] = fem(varargin)
%
% This function is the heart of Sigg's FEM.
%
% It returns in a_soln the calculated node voltages, and in F_soln the calcul
ated node currents.
% In the case of our research group, it is the current vector (F_soln) which
is of interest as it is currents
% that we read from the rig.  Note that both a_soln and F_soln will be pby8 m
atrices, where p is the number
% of nodes in the mesh.
%
% The reason it has 8 columns is because of what Siggy called projections. I
n his case the 1st projection
% would be the case where transmitter electrode 1 was active. The next projec
tion would be when transmitter
% electrode 2 was active.  Each projection appears in the corresponding colum
n number.
% In FDM it is the same: i.e. we treat each transmitter as if it was the only
 active one - which is sort of
% the case as instead of being separated in time, it is separated in frequenc
y.

global N gauss_order no_elm no_nodes p t rxtx

[sig, ne] = 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, hen
ce update sig.
    sig = varargin{:}; % This case is used for calculation of jacobian for th
e newton-raphson method.
else
    error('incorrect input arguments');  % if any other number of input argum
ents, 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
    size(p,1)
    error('No p defined or you probably forgot to define the boundary conditi
ons 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)];  %dp
hi/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(e
lm) * (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




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 ar
e 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 th
e 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 indi
ces of pre_idx_all whose nodes are non_zero, i.e. have active voltages applie
d.

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

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

% %FIND THE FIRST NODE THAT CAN BE USED AS A REFERENCE NODE
% remove = [pre_idx_act;pre_idx_grnd];
% rmidx = [1:no_nodes]';    % create a basic index matrix
% rmidx(remove) = 0;       % set all indices that are in 'remove' to zero.
% rmidx = rmidx(find(rmidx));
% rmnode = rmidx(1);
%
% %SET REFERENCE NODE DIAGONAL TO 1 AND COLUMN/ROW TO 0
% Kmod(:,rmnode)=zeros(size(Kmod,1),1);
% Kmod(rmnode,:)=zeros(1,size(Kmod,2));
% Kmod(rmnode,rmnode)=1;

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

    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
```

C.4 Simulated training data

maketraindata.m

```
%written by Gareth Goldswain, July 2005

function [train, sigdist] = maketraindata(bubsize, ne, nr)

%bubsize is the radius of the required bubble in terms of a percentage of the
 radius of the pipe

[p, t, rxtx] = mesh16(ne, nr);
t_cent = centroids(t, p);
%create a course mesh - the centroids of this mesh's elements will be the var
ious
%bubble position centres used
[pc, tc, rxtxc] = mesh16(1, 5);
t_centc = centroids(tc, pc);
num_bub_pos=length(t_centc);
train=[];
sigdist=[];

homog = ones(1,length(t));
homog_sig = 0.001*homog;
[uh, ih] = fem(homog_sig);
Yh=sumnodes(ih,rxtx,ne);
homog_readings=[Yh(:,1);Yh(:,2);Yh(:,3);Yh(:,4);Yh(:,5);Yh(:,6);Yh(:,7);Yh(:,
8)];

for bub_pos=1:num_bub_pos
    sig=homog;
    centre=[t_centc(bub_pos,1), t_centc(bub_pos,2)];
    bubrad=((bubsize+randn)/100)*(0.057/2);
    for i=1:length(t)
        x=t_cent(i,1);
        y=t_cent(i,2);
        if ((x-centre(1))^2 + (y-centre(2))^2 <= bubrad^2) & (sqrt(x^2 + y^2)
 < 0.025)
            sig(i)=0; %air
        end
    end

    sigdist=[sigdist;sig];

    volfrac=length(find(sig==0))/length(sig);
    sig(find(sig==0))=10e-6;
```

```
    sig(find(sig==1))=0.001;
    [v,c]=fem(sig);
    Y=sumnodes(c,rxtx,ne);
    readings=[Y(:,1);Y(:,2);Y(:,3);Y(:,4);Y(:,5);Y(:,6);Y(:,7);Y(:,8)];
    diff=homog_readings-readings;
    train=[train;[readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,sig)
    subplot(1,2,2)
    plot(readings)
    hold on
    plot(diff*20-10^-3,'g')
    hold off
    drawnow
end
```

buildtraindata.m

```
%written by Gareth Goldswain, July 2005

%function [D, train_2bub, train_3bub, train_4bub, train_5bub, train_6bub, tra
in_7bub, train_8bub, train_9bub, train_10bub] = buildtraindata(low_bubsize, h
igh_bubsize, ne, nr)

low_bubsize=0.01;
high_bubsize=0.035;
ne=3;
nr=15;
D=[];
sigdist=[];
train_2bub=[];
train_3bub=[];
train_4bub=[];
train_5bub=[];
train_6bub=[];
train_7bub=[];
train_8bub=[];
train_9bub=[];
train_10bub=[];
%create the fine mesh for solving forward problem
[p, t, rxtx] = mesh16(ne, nr);
t_cent = centroids(t, p);

%change the following vector according to the bubble sizes you want to train
for
rads=linspace(low_bubsize, high_bubsize,5);  %bubble sizes in terms of vol fr
ac
rads=rads.*(0.057^2);
rads=sqrt(rads); % do this so training data has linearly spaced vol fracs
rads=rads./0.057;
rads=rads.*100;
[train, sig] = maketraindata(rads(1), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
[train, sig] = maketraindata(rads(2), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
[train, sig] = maketraindata(rads(3), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
```

```
[train, sig] = maketraindata(rads(4), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
[train, sig] = maketraindata(rads(5), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(6), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(7), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(8), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(9), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(10), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];

homog = ones(1,length(t));
homog_sig = 0.001*homog;
[uh, ih] = fem(homog_sig);
Yh=sumnodes(ih,rxtx,ne);
homog_readings=[Yh(:,1);Yh(:,2);Yh(:,3);Yh(:,4);Yh(:,5);Yh(:,6);Yh(:,7);Yh(:,
8)];

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));

%make random combinations of 2 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    new_sig=or(sig1, sig2);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
```

```
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_2bub=[train_2bub;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));


%make random combinations of 3 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_3bub=[train_3bub;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
```

```
end
randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));

%make random combinations of 4 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_4bub=[train_4bub;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));

%make random combinations of 5 bubbles
```

```
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_5bub=[train_5bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
%make random combinations of 6 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
```

```
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    sig6=sigdist(randpick6(i), :);
    sig6=not(sig6);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=or(new_sig, sig6);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_6bub=[train_6bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
randpick7=randperm(size(D,1));
%make random combinations of 7 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
```

krr.m

```
% extracted from Prof. Greene's "Neural, Fuzzy and Evolving Systems" - a cour
se taught in the Department of Electrical Engineering, University of Cape Tow
n, 2003.

function [yte]=krr(Xtr,ttr,sigma,gamma,Xte)

D2=dist2(Xtr,Xtr);
K=exp(-D2/(2*sigma^2));
K=K+gamma*eye(size(K));
alpha=K\ttr;
D2=dist2(Xte,Xtr);
K=exp(-D2/(2*sigma^2));
yte=K*alpha
```

krrtraintest.m

```
% extracted from Prof. Greene's "Neural, Fuzzy and Evolving Systems" - a cour
se taught in the Department of Electrical Engineering, University of Cape Tow
n, 2003.

function err=krrtraintest(Xtr,ttr,Xte,tte,sigma,gamma)

D2=dist2(Xtr,Xtr);
K=exp(-D2/(2*sigma^2));
K=K+gamma*eye(size(K));
alpha=K\ttr;
D2=dist2(Xte,Xtr);
K=exp(-D2/(2*sigma^2));
yte=K*alpha;
err=mean((tte-yte).^2);
```

normal_svd.m

```
function [U, S, V, Dnorm, meanD, stdD] = normal_svd(D, cutoff)

% Written by G Goldswain June 2004
%
% [U, S, V, Dnorm, meanD, stdD] = normal_svd(D, cutoff)
%
% Normalise the input data matrix and perform a singular value decomposition
% on it.  The returned S matrix will have all singular values which contribut
e
% less than 'cutoff'% the data's total variance zeroed.  U and V are the usua
l
% remaining svd matrices.
% Dnorm is the normalised Data matrix where meanD and stdD are the columnwise
 mean
% and standard deviation of D.
% NB to note is that this function expects the input matrix to be the trainin
g
% data (X) concatenated with the target values (t).

%r=# of rows (instances) of the input data matrix (D)
r=size(D,1);
%c=# of columns (features + 1) of the input data matrix (D)
c=size(D,2);

%mean/stdX is a row vector with each column containing the mean/std dev
%of the input matrix's columns (features)
meanD = mean(D,1);
stdD = std(D);

e=ones(r,1);
%subtract the mean and divide by the std dev of each column
Dnorm = (D - e*meanD);
Dnorm = Dnorm ./ (e*(stdD + (stdD==0) ));

%perform a compact svd all rows and all but the last column (which is reserve
d
%for target values)
[U S V]=svd(Dnorm(:, 1:c-1),0);
total=trace(S);
diagS=diag(S);

for i = 1:length(diagS)
```

```
    contrib=(diagS(i)/total)*100;
    if contrib < cutoff
       S(i,i)=0;
    end
end
```

scansg10fold.m

```
%Written by Gareth Goldswain, June 2005.
%Used to perform a grid search for optimum tuning parameters of KRR.

function [E, sig, gam, bestsig, bestgam] =scansg10fold(X,t)
sig=[10^-1.5 10^-1 10^-0.5 10^0 10^0.5 10^1 20 10^1.5 10^2 10^2.5 10^3];
gam=[10^-6 10^-5 10^-4 10^-3 10^-2 10^-1 10^0 10^1 10^2 10^3];

E=[];
for s=1:length(sig)
    for g=1:length(gam)
        ss=sig(s);
        gg=gam(g);
        e=tenfoldkrr(X,t,ss,gg);
        E(s,g)=e
    end
end

[r,c]=find( E==min( min(E) ) );
bestsig=sig(r);
bestgam=gam(c);
```

tenfoldkrr.m

```
% extracted from Prof. Greene's "Neural, Fuzzy and Evolving Systems" - a cour
se taught in the Department of Electrical Engineering, University of Cape Tow
n, 2003.

function E=tenfoldkrr(X,t,s,g)

D=[X,t];
[r,c]=size(X);
n=round(0.9*r);
Err=[];

for j=1:10
    X1=D(1:n,1:c);
    t1=D(1:n,c+1);
    Q1=D(n+1:r,1:c);
    y1=D(n+1:r,c+1);

    err=krrtraintest(X1,t1,Q1,y1,s,g);
    Err=[Err,err];

    D=[[Q1;X1],[t1;y1]];
end

E=mean(Err);
```

DIST2.m

```
function n2 = dist2(x, c)
%DIST2  Calculates squared distance between two sets of points.
%
%   Description
%   D = DIST2(X, C) takes two matrices of vectors and calculates the
%   squared Euclidean distance between them.  Both matrices must be of
%   the same column dimension.  If X has M rows and N columns, and C has
%   L rows and N columns, then the result has M rows and L columns.  The
%   I, Jth entry is the  squared distance from the Ith row of X to the
%   Jth row of C.
%
%   See also
%   GMMACTIV, KMEANS, RBFFWD
%

%   Copyright (c) Ian T Nabney (1996-2001)

[ndata, dimx] = size(x);
[ncentres, dimc] = size(c);
if dimx ~= dimc
    error('Data dimension does not match dimension of centres')
end

n2 = (ones(ncentres, 1) * sum((x.^2)', 1))' + ...
  ones(ndata, 1) * sum((c.^2)',1) - ...
  2.*(x*(c'));

% Rounding errors occasionally cause negative entries in n2
if any(any(n2<0))
  n2(n2<0) = 0;
end
```

C.3 Finite Element Method (and meshing algorithm)

```
function [sigma, ne] = elmtprop
% written by S. Herholdt 2004.
% modified for use by G. Goldswain 2005.
%
%   THIS FUNCTION IS WHERE DATA SPECIFIC TO EACH PROBLEM IS ENTERED.
%   THE OTHER FUNCTIONS ARE NOT MEANT TO BE CHANGED, UNLESS THE TYPE OF PROBL
EM COMPLETELY CHANGES -
%   - IF THAT HAPPENS,NEW CASES NEED TO BE ADDED TO THE VARIOUS FUNCTIONS
%
% This function returns the element properties. N (number of nodes per elemen
t), gauss_order, no_elm
% (number of elements), no_nodes (number of nodes), p, t, rxtx (boundary node
 information; see
% outering.m) are all globals.
%
% [sigma, ne] = elmtprop
%
% returns in ne the number of element edges per electrode.  This is used by s
umnodes.m
% returns in sigma a vector containing each elements conductivity.  This cond
uctivity vector is created
% in elmtprop as a homogeonous medium.

% Data for different resolution meshes has been precomputed and saved.  This
function  loads the data
% each time it is called instead of recalculating it each time.  To load diff
erent mesh data, uncomment
% the desired lines of code below.

global N gauss_order no_elm no_nodes p t rxtx



% 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.
%     {Siggy forgot to mention here that there actually exists another row in
%      his p_matrix - it contains the projection number}
%
%     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 th
e case of
%     conductivity.


% CURRENT EXPERIMENT IS BASED ON DATA BELOW  (DATE:04 April 2005 - G.Goldswai
n)

N=3;
% load mesh data for a mesh with "mesh_()" elements "ne()" elements per elect
rode/electrode gap
% and "nr()" rings of elements
%  load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_70_ne1_nr3
%  ne=1;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_108_ne1_nr4
% ne=1;
%load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_158_ne1_nr5
%ne=1;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_252_ne2_nr6
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_328_ne2_nr7
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_416_ne2_nr8
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_516_ne2_nr9
% ne=2;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_662_ne3_nr10
```

```
% ne=3;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_328_ne2_nr7
load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_1416_ne3_nr15
ne=3;
% load c:\MSc\Matlab\MyFEM\Mesh\Mesh_data\mesh_2516_ne4_nr20
% ne=4;

%pad the t matrix's 4th row (subdomain number according to Matlab) with 0's
dummy  = zeros(1,length(t));
t=[t;dummy];
%add another row to t.  This row contains the conductivity value for each ele
ment
cond=ones(1,length(t));
cond=cond*0.001;
t=[t;cond];

%            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 [a_soln,F_soln, rxtx, ne] = fem(varargin)

% Written by Siggy Herholdt, 2004
% Modified for use by Gareth Goldswain, 2005
%
% [a_soln,F_soln, rxtx, ne] = fem(varargin)
%
% This function is the heart of Sigg's FEM.
%
% It returns in a_soln the calculated node voltages, and in F_soln the calcul
ated node currents.
% In the case of our research group, it is the current vector (F_soln) which
is of interest as it is currents
% that we read from the rig.  Note that both a_soln and F_soln will be pby8 m
atrices, where p is the number
% of nodes in the mesh.
%
% The reason it has 8 columns is because of what Siggy called projections. I
n his case the 1st projection
% would be the case where transmitter electrode 1 was active. The next projec
tion would be when transmitter
% electrode 2 was active.  Each projection appears in the corresponding colum
n number.
% In FDM it is the same: i.e. we treat each transmitter as if it was the only
 active one - which is sort of
% the case as instead of being separated in time, it is separated in frequenc
y.

global N gauss_order no_elm no_nodes p t rxtx

[sig, ne] = 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, hen
ce update sig.
    sig = varargin{:}; % This case is used for calculation of jacobian for th
e newton-raphson method.
else
    error('incorrect input arguments');  % if any other number of input argum
ents, 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
    size(p,1)
    error('No p defined or you probably forgot to define the boundary conditi
ons 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)];  %dp
hi/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(e
lm) * (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




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 ar
e 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 th
e 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 indi
ces of pre_idx_all whose nodes are non_zero, i.e. have active voltages applie
d.

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

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

% %FIND THE FIRST NODE THAT CAN BE USED AS A REFERENCE NODE
% remove = [pre_idx_act;pre_idx_grnd];
% rmidx = [1:no_nodes]';    % create a basic index matrix
% rmidx(remove) = 0;       % set all indices that are in 'remove' to zero.
% rmidx = rmidx(find(rmidx));
% rmnode = rmidx(1);
%
% %SET REFERENCE NODE DIAGONAL TO 1 AND COLUMN/ROW TO 0
% Kmod(:,rmnode)=zeros(size(Kmod,1),1);
% Kmod(rmnode,:)=zeros(1,size(Kmod,2));
% Kmod(rmnode,rmnode)=1;

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

    currentidx_of_preidxact = find(p(5,pre_idx_act)==proj_num) ; %find the no
des with active voltages for each projection
    pre_idx = [pre_idx_act(currentidx_of_preidxact); pre_idx_grnd]; %concaten
ate the index of the node where the active voltage is applied and the grounde
d nodes
    num_pre = size(pre_idx,1);  %this variable contains the number of predete
rmined 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 ma
trix

    %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_p
reidxact)); %write the boundary conditions into the a vector (from p matrix t
hat 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
```

C.4 Simulated training data

maketraindata.m

```
%written by Gareth Goldswain, July 2005

function [train, sigdist] = maketraindata(bubsize, ne, nr)

%bubsize is the radius of the required bubble in terms of a percentage of the
 radius of the pipe

[p, t, rxtx] = mesh16(ne, nr);
t_cent = centroids(t, p);
%create a course mesh - the centroids of this mesh's elements will be the var
ious
%bubble position centres used
[pc, tc, rxtxc] = mesh16(1, 5);
t_centc = centroids(tc, pc);
num_bub_pos=length(t_centc);
train=[];
sigdist=[];

homog = ones(1,length(t));
homog_sig = 0.001*homog;
[uh, ih] = fem(homog_sig);
Yh=sumnodes(ih,rxtx,ne);
homog_readings=[Yh(:,1);Yh(:,2);Yh(:,3);Yh(:,4);Yh(:,5);Yh(:,6);Yh(:,7);Yh(:,
8)];

for bub_pos=1:num_bub_pos
    sig=homog;
    centre=[t_centc(bub_pos,1), t_centc(bub_pos,2)];
    bubrad=((bubsize+randn)/100)*(0.057/2);
    for i=1:length(t)
        x=t_cent(i,1);
        y=t_cent(i,2);
        if ((x-centre(1))^2 + (y-centre(2))^2 <= bubrad^2) & (sqrt(x^2 + y^2)
 < 0.025)
            sig(i)=0; %air
        end
    end

    sigdist=[sigdist;sig];

    volfrac=length(find(sig==0))/length(sig);
    sig(find(sig==0))=10e-6;
```

```
    sig(find(sig==1))=0.001;
    [v,c]=fem(sig);
    Y=sumnodes(c,rxtx,ne);
    readings=[Y(:,1);Y(:,2);Y(:,3);Y(:,4);Y(:,5);Y(:,6);Y(:,7);Y(:,8)];
    diff=homog_readings-readings;
    train=[train;[readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,sig)
    subplot(1,2,2)
    plot(readings)
    hold on
    plot(diff*20-10^-3,'g')
    hold off
    drawnow
end
```

buildtraindata.m

```
%written by Gareth Goldswain, July 2005

%function [D, train_2bub, train_3bub, train_4bub, train_5bub, train_6bub, tra
in_7bub, train_8bub, train_9bub, train_10bub] = buildtraindata(low_bubsize, h
igh_bubsize, ne, nr)

low_bubsize=0.01;
high_bubsize=0.035;
ne=3;
nr=15;
D=[];
sigdist=[];
train_2bub=[];
train_3bub=[];
train_4bub=[];
train_5bub=[];
train_6bub=[];
train_7bub=[];
train_8bub=[];
train_9bub=[];
train_10bub=[];
%create the fine mesh for solving forward problem
[p, t, rxtx] = mesh16(ne, nr);
t_cent = centroids(t, p);

%change the following vector according to the bubble sizes you want to train
for
rads=linspace(low_bubsize, high_bubsize,5);  %bubble sizes in terms of vol fr
ac
rads=rads.*(0.057^2);
rads=sqrt(rads); % do this so training data has linearly spaced vol fracs
rads=rads./0.057;
rads=rads.*100;
[train, sig] = maketraindata(rads(1), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
[train, sig] = maketraindata(rads(2), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
[train, sig] = maketraindata(rads(3), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
```

```
[train, sig] = maketraindata(rads(4), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
[train, sig] = maketraindata(rads(5), 3, 15);
D=[D;train];
sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(6), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(7), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(8), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(9), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];
% [train, sig] = maketraindata(rads(10), 3, 15);
% D=[D;train];
% sigdist=[sigdist;sig];

homog = ones(1,length(t));
homog_sig = 0.001*homog;
[uh, ih] = fem(homog_sig);
Yh=sumnodes(ih,rxtx,ne);
homog_readings=[Yh(:,1);Yh(:,2);Yh(:,3);Yh(:,4);Yh(:,5);Yh(:,6);Yh(:,7);Yh(:,
8)];

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));

%make random combinations of 2 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    new_sig=or(sig1, sig2);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
```

```
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_2bub=[train_2bub;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));


%make random combinations of 3 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_3bub=[train_3bub;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
```

```
end
randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));

%make random combinations of 4 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_4bub=[train_4bub;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));

%make random combinations of 5 bubbles
```

```
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_5bub=[train_5bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
%make random combinations of 6 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
```

```
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    sig6=sigdist(randpick6(i), :);
    sig6=not(sig6);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=or(new_sig, sig6);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_6bub=[train_6bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
randpick7=randperm(size(D,1));
%make random combinations of 7 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
```

```
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    sig6=sigdist(randpick6(i), :);
    sig6=not(sig6);
    sig7=sigdist(randpick7(i), :);
    sig7=not(sig7);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=or(new_sig, sig6);
    new_sig=or(new_sig, sig7);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_7bub=[train_7bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
randpick7=randperm(size(D,1));
randpick8=randperm(size(D,1));
%make random combinations of 8 bubbles
```

```
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    sig6=sigdist(randpick6(i), :);
    sig6=not(sig6);
    sig7=sigdist(randpick7(i), :);
    sig7=not(sig7);
    sig8=sigdist(randpick8(i), :);
    sig8=not(sig8);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=or(new_sig, sig6);
    new_sig=or(new_sig, sig7);
    new_sig=or(new_sig, sig8);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_8bub=[train_8bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
```

```
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
randpick7=randperm(size(D,1));
randpick8=randperm(size(D,1));
randpick9=randperm(size(D,1));

%make random combinations of 8 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    sig6=sigdist(randpick6(i), :);
    sig6=not(sig6);
    sig7=sigdist(randpick7(i), :);
    sig7=not(sig7);
    sig8=sigdist(randpick8(i), :);
    sig8=not(sig8);
    sig9=sigdist(randpick9(i), :);
    sig9=not(sig9);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
    new_sig=or(new_sig, sig6);
    new_sig=or(new_sig, sig7);
    new_sig=or(new_sig, sig8);
    new_sig=or(new_sig, sig9);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_9bub=[train_9bub ;[new_readings',volfrac]];
```

```
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end

randpick1=randperm(size(D,1));
randpick2=randperm(size(D,1));
randpick3=randperm(size(D,1));
randpick4=randperm(size(D,1));
randpick5=randperm(size(D,1));
randpick6=randperm(size(D,1));
randpick7=randperm(size(D,1));
randpick8=randperm(size(D,1));
randpick9=randperm(size(D,1));
randpick10=randperm(size(D,1));
%make random combinations of 8 bubbles
for i = 1:length(D)
    sig1=sigdist(randpick1(i),:);
    sig1=not(sig1);
    sig2=sigdist(randpick2(i), :);
    sig2=not(sig2);
    sig3=sigdist(randpick3(i),:);
    sig3=not(sig3);
    sig4=sigdist(randpick4(i), :);
    sig4=not(sig4);
    sig5=sigdist(randpick5(i), :);
    sig5=not(sig5);
    sig6=sigdist(randpick6(i), :);
    sig6=not(sig6);
    sig7=sigdist(randpick7(i), :);
    sig7=not(sig7);
    sig8=sigdist(randpick8(i), :);
    sig8=not(sig8);
    sig9=sigdist(randpick9(i), :);
    sig9=not(sig9);
    sig10=sigdist(randpick10(i), :);
    sig9=not(sig10);
    new_sig=or(sig1, sig2);
    new_sig=or(new_sig, sig3);
    new_sig=or(new_sig, sig4);
    new_sig=or(new_sig, sig5);
```

```
    new_sig=or(new_sig, sig6);
    new_sig=or(new_sig, sig7);
    new_sig=or(new_sig, sig8);
    new_sig=or(new_sig, sig9);
    new_sig=or(new_sig, sig10);
    new_sig=not(new_sig);
    volfrac=length(find(new_sig==0))/length(new_sig);
    new_sig(find(new_sig==0))=10e-6;
    new_sig(find(new_sig==1))=0.001;
    [new_u, new_i]=fem(new_sig);
    new_Y=sumnodes(new_i,rxtx,ne);
    new_readings=[new_Y(:,1);new_Y(:,2);new_Y(:,3);new_Y(:,4);new_Y(:,5);new_
Y(:,6);new_Y(:,7);new_Y(:,8)];
    new_diff=homog_readings-new_readings;
    train_10bub=[train_10bub ;[new_readings',volfrac]];
    subplot(1,2,1)
    pdesurf(p,t,new_sig)
    subplot(1,2,2)
    plot(new_readings)
    hold on
    plot(new_diff*20,'g')
    hold off
    drawnow
end
```

Appendix D

Datasheets

Ultralow Distortion, Ultralow Noise Op Amp

AD797*

FEATURES

Low Noise
- 0.9 nV/√Hz typ (1.2 nV/√Hz max) Input Voltage Noise at 1 kHz
- 50 nV p-p Input Voltage Noise, 0.1 Hz to 10 Hz

Low Distortion
- –120 dB Total Harmonic Distortion at 20 kHz

Excellent AC Characteristics
- 800 ns Settling Time to 16 Bits (10 V Step)
- 110 MHz Gain Bandwidth (G = 1000)
- 8 MHz Bandwidth (G = 10)
- 280 kHz Full Power Bandwidth at 20 V p-p
- 20 V/μs Slew Rate

Excellent DC Precision
- 80 μV max Input Offset Voltage
- 1.0 μV/°C V_{OS} Drift

Specified for ±5 V and ±15 V Power Supplies

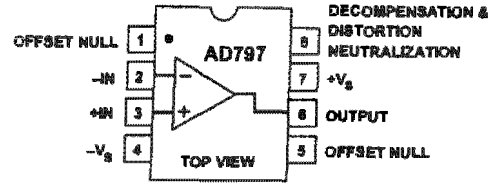
High Output Drive Current of 50 mA

APPLICATIONS

Professional Audio Preamplifiers
IR, CCD, and Sonar Imaging Systems
Spectrum Analyzers
Ultrasound Preamplifiers
Seismic Detectors
ΣΔ ADC/DAC Buffers

CONNECTION DIAGRAM

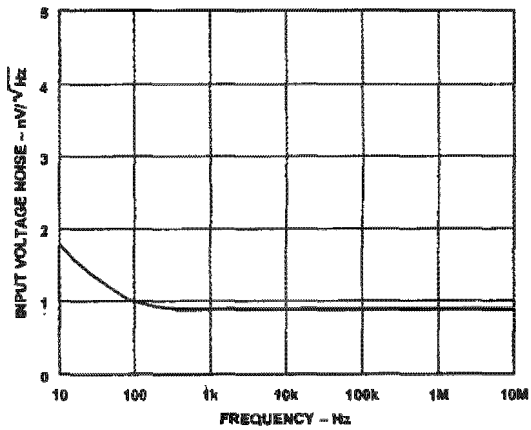
8-Pin Plastic Mini-DIP (N), Cerdip (Q) and SOIC (R) Packages



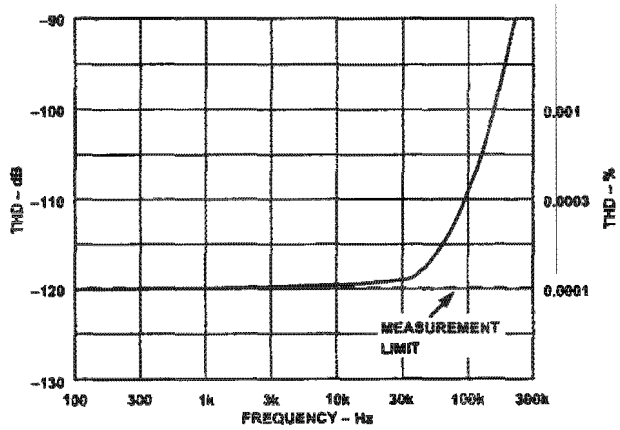
PRODUCT DESCRIPTION

The AD797 is a very low noise, low distortion operational amplifier ideal for use as a preamplifier. The low noise of 0.9 nV/√Hz and low total harmonic distortion of –120 dB at audio bandwidths give the AD797 the wide dynamic range necessary for preamps in microphones and mixing consoles. Furthermore, the AD797's excellent slew rate of 20 V/μs and 110 MHz gain bandwidth make it highly suitable for low frequency ultrasound applications.

The AD797 is also useful in IR and Sonar Imaging applications where the widest dynamic range is necessary. The low distortion and 16-bit settling time of the AD797 make it ideal for buffering the inputs to ΣΔ ADCs or the outputs of high resolution DACs especially when they are used in critical applications such as seismic detection and spectrum analyzers. Key features such as a 50 mA output current drive and the specified power supply voltage range of ±5 to ±15 volts make the AD797 an excellent general purpose amplifier.



AD797 Voltage Noise Spectral Density



THD vs. Frequency

*Patent pending.

REV. C

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 617/329-4700 Fax: 617/326-8703

AD797—SPECIFICATIONS (@ T_A = +25°C and V_S = ±15 V dc, unless otherwise noted)

Model	Conditions	V_S	AD797A/S[1] Min	AD797A/S[1] Typ	AD797A/S[1] Max	AD797B Min	AD797B Typ	AD797B Max	Units
INPUT OFFSET VOLTAGE		±5 V, ±15 V		25	80		10	40	μV
	T_{MIN} to T_{MAX}			50	125/180		30	60	μV
Offset Voltage Drift		±5 V, ±15 V		0.2	1.0		0.2	0.6	μV/°C
INPUT BIAS CURRENT		±5 V, ±15 V		0.25	1.5		0.25	0.9	μA
	T_{MIN} to T_{MAX}			0.5	3.0		0.25	2.0	μA
INPUT OFFSET CURRENT		±5 V, ±15 V		100	400		80	200	nA
	T_{MIN} to T_{MAX}			120	600/700		120	300	nA
OPEN-LOOP GAIN	V_{OUT} = ±10 V R_{LOAD} = 2 kΩ	±15 V	1	20		2	20		V/μV
	T_{MIN} to T_{MAX}		1	6		2	10		V/μV
	R_{LOAD} = 600 Ω		1	15		2	15		V/μV
	T_{MIN} to T_{MAX}		1	5		2	7		V/μV
	@ 20 kHz[2]		14000	20000		14000	20000		V/V
DYNAMIC PERFORMANCE									
Gain Bandwidth Product	G = 1000	±15 V		110			110		MHz
	G = 1000[2]	±15 V		450			450		MHz
-3 dB Bandwidth	G = 10	±15 V		8			8		MHz
Full Power Bandwidth[3]	V_O = 20 V p-p,								
	R_{LOAD} = 1 kΩ	±15 V		280			280		kHz
Slew Rate	R_{LOAD} = 1 kΩ	±15 V	12.5	20		12.5	20		V/μs
Settling Time to 0.0015%	10 V Step	±15 V		800	1200		800	1200	ns
COMMON-MODE REJECTION	V_{CM} = CMVR	±5 V, ±15 V	114	130		120	130		dB
	T_{MIN} to T_{MAX}		110	120		114	120		dB
POWER SUPPLY REJECTION	V_S = ±5 V to ±18 V		114	130		120	130		dB
	T_{MIN} to T_{MAX}		110	120		114	120		dB
INPUT VOLTAGE NOISE	f = 0.1 Hz to 10 Hz	±15 V		50			50		nV p-p
	f = 10 Hz	±15 V		1.7			1.7	2.5	nV/√Hz
	f = 1 kHz	±15 V		0.9	1.2		0.9	1.2	nV/√Hz
	f = 10 Hz–1 MHz	±15 V		1.0	1.3		1.0	1.2	μV rms
INPUT CURRENT NOISE	f = 1 kHz	±15 V		2.0			2.0		pA/√Hz
INPUT COMMON-MODE		±15 V	±11	±12		±11	±12		V
VOLTAGE RANGE		±5 V	±2.5	±3		±2.5	±3		V
OUTPUT VOLTAGE SWING	R_{LOAD} = 2 kΩ	±15 V	±12	±13		±12	±13		V
	R_{LOAD} = 600 Ω	±15 V	±11	±13		±11	±13		V
	R_{LOAD} = 600 Ω	±5 V	±2.5	±3		±2.5	±3		V
Short-Circuit Current		±5 V, ±15 V		80			80		mA
Output Current[4]		±5 V, ±15 V	30	50		30	50		mA
TOTAL HARMONIC DISTORTION	R_{LOAD} = 1 kΩ, C_N = 50 pF f = 250 kHz, 3 V rms	±15 V		-98	-90		-98	-90	dB
	R_{LOAD} = 1 kΩ f = 20 kHz, 3 V rms	±15 V		-120	-110		-120	-110	dB
INPUT CHARACTERISTICS									
Input Resistance (Differential)				7.5			7.5		kΩ
Input Resistance (Common Mode)				100			100		MΩ
Input Capacitance (Differential)[5]				20			20		pF
Input Capacitance (Common Mode)				5			5		pF
OUTPUT RESISTANCE	A_V = +1, f = 1 kHz			3			3		mΩ
POWER SUPPLY									
Operating Range			±5		±18	±5		±18	V
Quiescent Current		±5 V, ±15 V		8.2	10.5		8.2	10.5	mA

NOTES
[1]See standard military drawing for 883B specifications.
[2]Specified using external decompensation capacitor, see Applications section.
[3]Full Power Bandwidth = Slew Rate/2 π V_{PEAK}.
[4]Output Current for $|V_S - V_{OUT}| > 4$ V, $A_{OL} > 200$ kΩ.
[5]Differential input capacitance consists of 1.5 pF package capacitance and 18.5 pF from the input differential pair.

Specifications subject to change without notice.

National Semiconductor

February 1995

LF411 Low Offset, Low Drift JFET Input Operational Amplifier

General Description

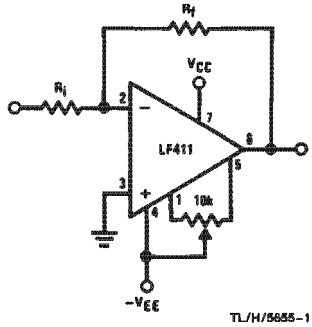
These devices are low cost, high speed, JFET input operational amplifiers with very low input offset voltage and guaranteed input offset voltage drift. They require low supply current yet maintain a large gain bandwidth product and fast slew rate. In addition, well matched high voltage JFET input devices provide very low input bias and offset currents. The LF411 is pin compatible with the standard LM741 allowing designers to immediately upgrade the overall performance of existing designs.

These amplifiers may be used in applications such as high speed integrators, fast D/A converters, sample and hold circuits and many other circuits requiring low input offset voltage and drift, low input bias current, high input impedance, high slew rate and wide bandwidth.

Features

■ Internally trimmed offset voltage	0.5 mV(max)
■ Input offset voltage drift	10 μV/°C(max)
■ Low input bias current	50 pA
■ Low input noise current	0.01 pA/√Hz
■ Wide gain bandwidth	3 MHz(min)
■ High slew rate	10V/μs(min)
■ Low supply current	1.8 mA
■ High input impedance	$10^{12}\Omega$
■ Low total harmonic distortion $A_V = 10$, $R_L = 10k$, $V_O = 20$ Vp-p, BW = 20 Hz – 20 kHz	<0.02%
■ Low 1/f noise corner	50 Hz
■ Fast settling time to 0.01%	2 μs

Typical Connection



TL/H/5655–1

Ordering Information

LF411XYZ

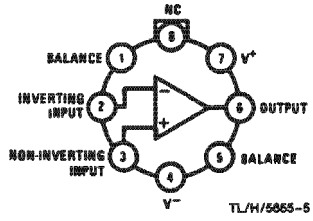
X indicates electrical grade

Y indicates temperature range
"M" for military
"C" for commercial

Z indicates package type
"H" or "N"

Connection Diagrams

Metal Can Package



TL/H/5655–5

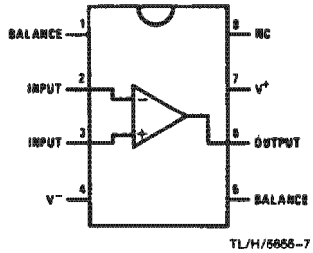
Top View

Note: Pin 4 connected to case.

Order Number LF411ACH or LF411MH/883*

See NS Package Number H08A

Dual-In-Line Package



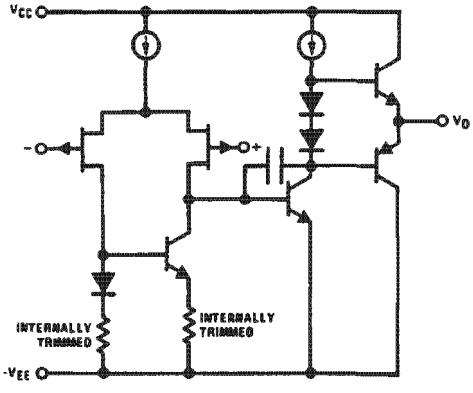
TL/H/5655–7

Top View

Order Number LF411ACN, LF411CN or LF411MJ/883*

See NS Package Number N08E or J08A

Simplified Schematic



TL/H/5655–6

BI-FET II™ is a trademark of National Semiconductor Corporation.

*Available per JM38510/11904

RRD-B30M115/Printed in U. S. A.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications. (Note 8)

	LF411A	LF411
Supply Voltage	±22V	±18V
Differential Input Voltage	±38V	±30V
Input Voltage Range (Note 1)	±19V	±15V
Output Short Circuit Duration	Continuous	Continuous

	H Package	N Package
Power Dissipation (Notes 2 and 9)	670 mW	670 mW
$T_j$max	150°C	115°C
θ_{jA}	162°C/W (Still Air) 65°C/W (400 LF/min Air Flow)	120°C/W
θ_{jC}	20°C/W	
Operating Temp. Range	(Note 3)	(Note 3)
Storage Temp. Range	$-65°C \le T_A \le 150°C$	$-65°C \le T_A \le 150°C$
Lead Temp. (Soldering, 10 sec.)	280°C	260°C
ESD Tolerance		Rating to be determined.

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions		LF411A			LF411			Units
				Min	Typ	Max	Min	Typ	Max	
V_{OS}	Input Offset Voltage	$R_S = 10\ k\Omega$, $T_A = 25°C$			0.3	0.5		0.8	2.0	mV
$\Delta V_{OS}/\Delta T$	Average TC of Input Offset Voltage	$R_S = 10\ k\Omega$ (Note 5)			7	10		7	20 (Note 5)	µV/°C
I_{OS}	Input Offset Current	$V_S = \pm 15V$ (Notes 4, 8)	$T_j = 25°C$		25	100		25	100	pA
			$T_j = 70°C$			2			2	nA
			$T_j = 125°C$			25			25	nA
I_B	Input Bias Current	$V_S = \pm 15V$ (Notes 4, 6)	$T_j = 25°C$		50	200		50	200	pA
			$T_j = 70°C$			4			4	nA
			$T_j = 125°C$			50			50	nA
R_{IN}	Input Resistance	$T_j = 25°C$			10^{12}			10^{12}		Ω
A_{VOL}	Large Signal Voltage Gain	$V_S = \pm 15V$, $V_O = \pm 10V$, $R_L = 2k$, $T_A = 25°C$		50	200		25	200		V/mV
		Over Temperature		25	200		15	200		V/mV
V_O	Output Voltage Swing	$V_S = \pm 15V$, $R_L = 10k$		±12	±13.5		±12	±13.5		V
V_{CM}	Input Common-Mode Voltage Range			±18	+19.5		±11	+14.5		V
					−16.5			−11.5		V
CMRR	Common-Mode Rejection Ratio	$R_S \le 10k$		80	100		70	100		dB
PSRR	Supply Voltage Rejection Ratio	(Note 7)		80	100		70	100		dB
I_S	Supply Current				1.8	2.8		1.8	3.4	mA

AC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	LF411A			LF411			Units
			Min	Typ	Max	Min	Typ	Max	
SR	Slew Rate	$V_S = \pm 15V$, $T_A = 25°C$	10	15		8	15		V/µs
GBW	Gain-Bandwidth Product	$V_S = \pm 15V$, $T_A = 25°C$	3	4		2.7	4		MHz
e_n	Equivalent Input Noise Voltage	$T_A = 25°C$, $R_S = 100\Omega$, $f = 1$ kHz		25			25		$nV/\sqrt{Hz}$
i_n	Equivalent Input Noise Current	$T_A = 25°C$, $f = 1$ kHz		0.01			0.01		$pA/\sqrt{Hz}$