Design and development of a distributed planar pressure sensor utilising electrical impedance tomography.

BOOTH, M. J.

Design And Development Of A Distributed Planar Pressure Sensor Utilising Electrical Impedance Tomography

M. J. Booth BEng (Hons)

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

January 2000

Dedications

This work is dedicated to my wife Davina for her patience and understanding during my time at University. Also to my Mother Joan Booth and Davina's parents Ray and Margaret Barnard for their support.

Acknowledgements

I would like to express my sincere thanks to my supervisor Dr. Ivan Basarab-Horwath for his support and assistance throughout the course of this research. I am grateful for his suggestions and constructive criticisms of my work and for the original concept on which the project was formed.

I would also like to record my gratitude to all the staff at Sheffield Hallam University who have given me invaluable support during my time as both a postgraduate and undergraduate. For my colleagues in room 2302 School of Engineering Neil, Joe, Hamid, Fuad, Ruzairi, Tony, Alan, Stuart and Jan, thank you for your valuable discussions and suggestions.

Abstract

This thesis describes an investigation into the use of electrical impedance tomography used in conjunction with a flexible conductive sensor for the measurement of distributed pressure. The main application areas are for the constant monitoring of the pressure distribution between a patient and their support surface i.e. beds and wheel chairs, in order to reduce the formation of pressure sores and tactile sensing for robotics. A number of systems have been developed for the monitoring of patients but non-have proved suitable for constant monitoring and these are reviewed. A review of the tactile sensor techniques used in robotic grippers is presented and when the area to be monitored is relatively small (1-2 cm^2) the techniques already under development can provide the resolution required. However no technique exists to measure distributed pressure over a large area.

A review of both the hardware and reconstruction algorithms used in electrical impedance tomography is presented and the design of the hardware and software developed for the investigation into the sensor design is detailed. As the sensor is such that electrodes are not limited to the periphery both an experimental and computer simulated comparison of three different electrode configurations is described. The three-electrode arrangements investigated are with the electrodes placed at the periphery of both a circular and square boundary, and with electrodes evenly distributed across a square area. The results from the comparisons show that the new distributed electrode arrangement performs significantly better than when the electrodes are confined to the periphery. It also shows that the geometry of the boundary when using peripheral electrodes can also effect the performance of an EIT system.

The initially investigated sensor design was based on a conductive polymer sheet and a number of samples were characterised in term of their V/I characteristics and their creep and resistance change due to applied pressure. Only one of the sample tested had a response worth investigating further but the material could not be obtained for larger area tests. Therefore an alternative sensor design was investigated. This novel sensor consisted of a conductive fluid retained beneath a flexible rubber membrane.

From electrical impedance tomography images obtained from the experimental evaluation of the new sensor design it is shown that the system can image the pressure distribution across its surface. In addition, the analysis of the unprocessed data from the new sensor shows the system to have a well-defined response with a wide applied pressure range and the construction of the sensor is such that its response could be tailored to the range of pressure to be measured.

Contents

1. Introduction

1.1 An overview of distributed pressure measurement

Distributed pressure sensing is concerned with the measurement of the pressure distribution over a given sensing area and a number of techniques have been developed (see Chapter 4 for a full review). The measurement of distributed pressure has a number of applications both in industry and medicine e.g. intelligent robot gripping (Ghani N., 1988 and Nicholls H.R., 1992), monitoring of pressure sores in bed ridden patients (Webster J.G., 1989, Knight R.A. and Lipczynski R.T., 1990) and foot pressure measurement (Lord M., Reynold D.D. and Hughes J.R., 1986). Figure 1.1.1 shows the typical components of a distributed pressure sensing system.

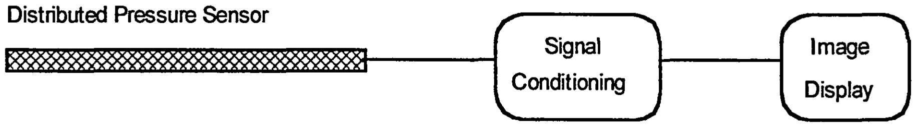


Figure 1.1.1 Block diagram of distributed pressure sensing system

1.2 An overview of electrical impedance tomography

Electrical impedance tomography (EIT) is used to determine the spatial distribution of electrical impedance within either a two or three-dimensional object. This spatial distribution of electrical impedance is traditionally calculated from measurements made at the boundary of the object (Barber D.C. and Brown B.H., 1984). The basic method used in electrical impedance tomography involves placing an array of electrodes on the boundary of the object. A constant ac current is injected between two of these electrodes and the potential difference between the remaining non-current carrying electrodes is measured. The above procedure is repeated for a number of current injection patterns, i.e. constant current injected between different pairs of electrodes. The data from the measurements is then used to reconstruct an image of the impedance distribution within the object. The reconstruction of the impedance distribution is performed by solving Laplace's equations in either two or three dimensions (Webster J.G., 1990).

The reconstructed images of the electrical impedance distribution are used to infer information about other parameters within the area being interrogated, e.g. concentration profiles within stirred mixing vessels (Lyon G.M. and Oakley J.P., 1992), in vivo images of the human body (Barber D. C. and Brown B.H., 1984) or the pressure distribution across a resistive pressure sensitive material or sensing element (Lacy P. and Basarab-Horwath I., 1993).

1.3 The aims and objectives of the work undertaken

The main aim of the work reported in this thesis is an investigation into the feasibility of using electrical impedance tomography in the measurement of distributed pressure applied to a conductive sensor element. Previous work by (Knight R.A. and Lipczynski R.T., 1990 and Knight R.A., 1991) has investigated the use of EIT for interface pressure measurements. However, this work has been limited to simulations of the sensor with no practical sensor element being examined. A major undertaking of the work reported here is the development of a practical sensor element. Previous work in EIT has been in its use as a medical imaging technique (Barber D. C. and Brown B.H.,

1984) and for imaging industrial processes (Xie C.G., 1993). The majority of the work in both these fields of use has assumed a circular geometry with the electrodes placed at the boundary of the object. The boundary geometry when measuring distributed pressure is usually square or rectangular (Knight R.A. and Lipczynski R.T., 1990 and Knight R.A., 1991) therefore different electrode configurations for such a boundary shape have been studied. The aims of the work are to:

I. Study the fundamental principles of electrical impedance tomography and developed suitable hardware and software for a simple EIT system.

II. Evaluate the use of conductive elastomers and rubbers for use as the sensor element.

III. Investigate alternative electrode configurations for electrical impedance tomography which may be applicable to this situation.

IV. Characterise the feasibility and performance of a distributed pressure sensor element which utilises electrical impedance tomography.

V. Propose suggestions for future work.

1.4 Organisation of the thesis

The contents of each Chapter are outlined below:

Chapter 1: A general overview of both distributed pressure measurement and electrical impedance tomography is presented.

Chapter 2: Presents the theoretical background of EIT and looks at both the data acquisition systems and reconstruction algorithms that have been developed previously.

Chapter 3: A detailed description of the EIT system developed for the work is presented. Also presented is the reconstruction algorithm used for the images presented in this thesis.

Chapter 4: Introduces the area of distributed pressure measurement and presents the sensor system initially proposed.

Chapter 5: Presents a study of different electrode configurations, which could be used for the sensor systems described in Chapters 4 and 6.

Chapter 6: Reports the work carried out on a novel sensor design that utilises EIT.

Chapter 7: This chapter presents the conclusions from the work and makes suggestions for future work.

2. Review of electrical impedance tomography

2.1 Introduction

This chapter describes the theoretical background to electrical impedance tomography and then introduces the measurement strategies that have been employed in EIT. The different data acquisition systems that have been constructed to perform the collection of EIT data are also presented along with the reconstruction algorithms that are used in electrical impedance tomography.

2.2 Theoretical Background

The electric field within a region where the conductivity is both inhomogeneous and anisotropic is governed by Poisson's equation

$$\nabla \cdot (\sigma \nabla V) = f \qquad (2.2.1)$$

With the region having boundary conditions

$$\mathrm{V} = \underline{\mathrm{Vo}} \qquad \text{on } \delta\mathrm{A} \qquad (2.2.2)$$

$$\sigma(\delta\mathrm{V}/\delta\mathrm{n}) = \underline{\mathrm{Jo}} \qquad \text{on } \delta A \qquad (2.2.3)$$

Where σ is the conductivity distribution within the region

V is the voltage distribution within the region

and f is the current sources within the region

∇ is $\underline{i}\frac{\delta}{\delta x}+\underline{j}\frac{\delta}{\delta y}+\underline{k}\frac{\delta}{\delta z}$ the Poisson operator

Vo is the measured differential voltage profile at the boundary

and Jo is the current density at the boundary

In EIT it is assumed that any current sources within the body are negligible, consequently $f = 0$ and equation (2.2.1) becomes

$$\nabla \cdot (\sigma \nabla V) = 0 \qquad (2.2.4)$$

If the region's conductivity is both homogeneous and isotropic then equation (2.2.4) further reduces to Laplace's equation

$$\nabla^2 V = 0 \qquad (2.2.5)$$

Where $\nabla^2 = \frac{\delta^2}{\delta x^2}+\frac{\delta^2}{\delta y^2}+\frac{\delta^2}{\delta z^2}$, the Laplace operator.

Solving equation (2.2.4), given the region's conductivity distribution *σ(x, y)* and boundary conditions Vo and Jo, to determine the internal voltages and current densities is referred to as the forward problem. This is shown diagrammatically in Figure 2.2.1.

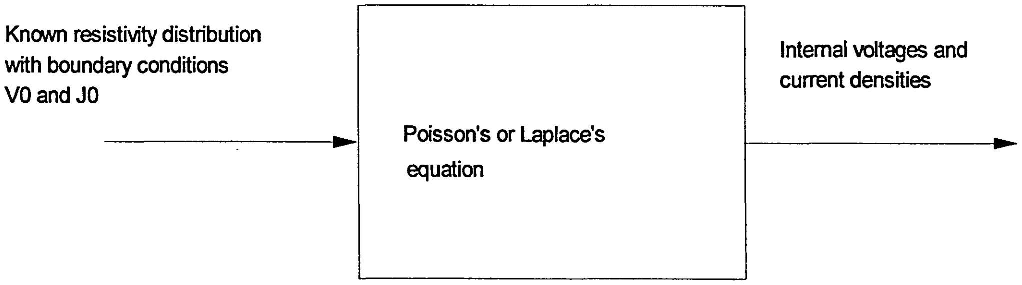


Figure 2.2.1 The Forward Problem

If the voltage and current densities are known then the problem is to find the conductivity distribution within the region. This is known as the inverse problem and is shown diagrammatically in Figure 2.2.2. Solving the inverse problem numerically usually requires iterative solutions of the forward problem. However many of the EIT algorithms do not use an iterative approach as is seen in section 2.4.

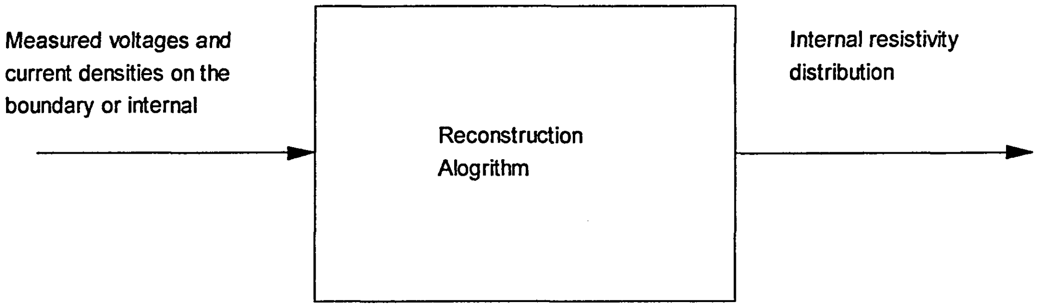


Figure 2.2.2 The Inverse Problem

2.3 EIT Measurement Strategies

Two basic measurement methods can be adopted for obtaining EIT data. Either apply a constant A.C. voltage and measure the current or apply a constant A.C. current and measure the voltage (Webster J.G., 1990). The constant voltage method has been used by W. Sansen et al, 1992, and they report that useful data has been obtained from the system. The voltage-drive methods hardware is not as complicated as that required for the constant current system. However a system based on a constant current source gives better accuracy and also produces better results with an unknown, varying contact impedance (Webster J.G., 1990). For this reason, a constant current/voltage measurement system has been adopted for the hardware developed in this research (see Chapter 3).

Whichever measurement method is utilised for the collection of EIT data, an electrode drive configuration also needs to be applied. The electrode drive configuration defines which electrodes the constant current (or voltage) is applied to and which electrodes the voltage (current) measurements are recorded from. Sections 2.3.1 to 2.3.5 of this chapter give details of the different drive configurations, which have been applied in EIT.

2.3.1 Adjacent (Neighbouring) Electrode Drive Configuration

The adjacent electrode drive configuration applies a constant A.C. current to two adjacent electrodes and measures the differential voltages developed at all other successive non-current carrying pairs of adjacent electrodes. Figure 2.3.1.1. shows this arrangement with 16 equally space electrodes. The current is shown injected between electrodes 1-2 and the resulting differential voltages would be recorded between 3-4, 4-5, 5-6...15-16. The differential voltages between the current carrying electrodes are not recorded, as an error is inherent in this measurement. The error is due to the contact impedance between the current carrying electrodes and the object to which they are attached (Brown B.H and Seagar A.D., 1985). The current source is than applied to the next successive pair of electrodes (2-3, 3-4...16-1) and the differential voltages between the remaining adjacent electrodes are measured. For the 16 electrode system shown in Figure 2.3.1.1 this gives 16 different current drive configurations (know as projections) with 13 differential voltage readings per projection.

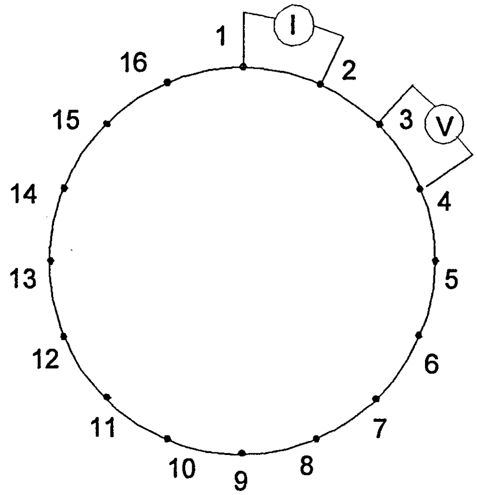


Figure 2.3.1.1 16 Adjacent Electrode Drive Configuration with Constant Current Applied between Electrodes 1 and 2. Voltage Measured between Electrodes 3 and 4.

2.3.2 Polar Electrode Drive Configuration

The polar electrode drive configuration applies a constant A.C. current through diametrically opposite electrodes. The voltage measurements are then recorded in one of two ways. Hua P., Webster J.G. and Tompkins W.J., 1987, (and described by Webster J.G., 1990) used a system where one of the electrodes next to the current injecting electrode was taken as a reference point. The voltages at the other non-current carrying electrodes are recorded. This electrode drive configuration gives 8 projections with 13 voltage measurements per projection. Alternatively, the differential voltages developed between adjacent electrodes can be measured. This gives 12 voltage readings per protection (Avis N. J. and Barber D.C., 1994) Figure 2.3.2.1 shows this strategy.

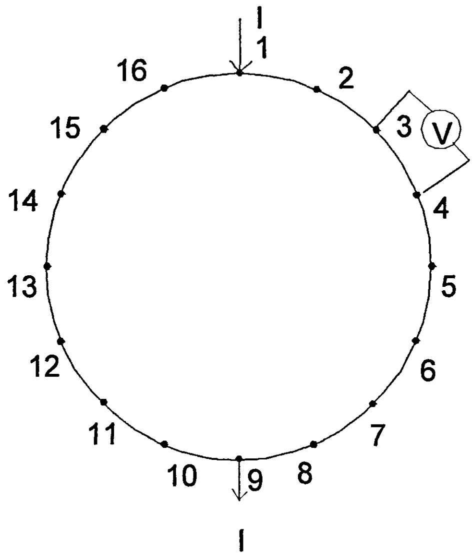


Figure 2.3.2.1 Polar Electrode Drive Configuration with Constant Current Applied between Electrodes 1 and 9. Voltage Measured between Electrodes 3 and 4.

2.3.3 Multireference Electrode Drive Configuration

The multireference electrode drive configuration used by Hua P. et al, 1987 and described by Webster J.G., 1990 uses one electrode as a reference (ground) and current is injected simultaneously between the reference and all other electrodes. Voltage measurements are then recorded between the reference electrode and all the other electrodes. Figure 2.3.3.1 shows this strategy applied to a 16-electrode system, which would give 16 projections with 15 voltage readings per projection giving a total of 240 readings. In Figure 2.3.3.1 electrode 9 is shown as the reference (ground) and therefore current is simultaneously injected between the reference and electrodes 1 to 8 and electrodes 10 to 16.

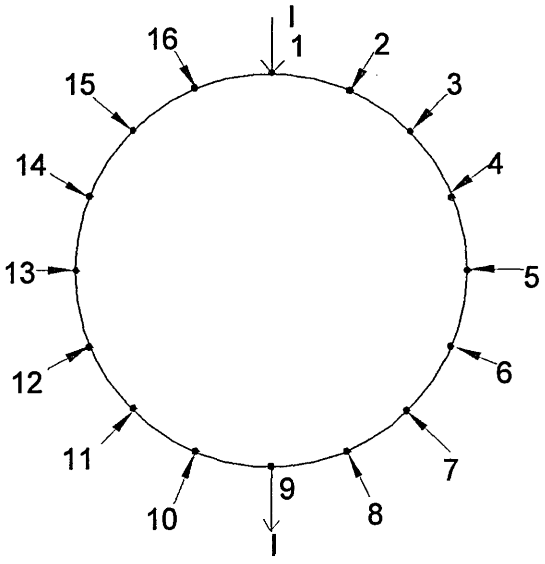


Figure 2.3.3.1 Multireference Electrode Drive Configuration. Electrode 9 used as Reference, Current Injected and Voltage Measured With Respect To This Electrode on All Other Electrodes.

2.3.4 Adaptive Electrode Drive Configuration

The adaptive electrode drive configuration proposed by Gisser D.G., Isaacson D. and Newell J.C., 1987 is similar to the multireference strategy. Current is injected between all electrodes simultaneously and voltages are measured with reference to a single grounded electrode. The current magnitude on each electrode is set between -5mA and +5mA. The magnitude of the current is set to give the optimal current distribution for the conductivity distribution being imaged. For a homogeneous conductivity distribution Gisser D.G., Isaacson D. and Newell J.C., 1987 showed that, to give a uniform current distribution within the interrogated area the current magnitude should follow a cosinusodal function with angular displacement along the circular boundary of the interrogated area.

2.3.5 Linear Electrode Drive Configuration

All the electrode drive configurations detailed above have been for imaging an area, which is bounded by the electrodes. Applications exist especially in geological applications (Webster J.G., 1990) where the electrodes cannot be placed around the boundary. It is for these situations that the linear array has been developed. A number of electrode drive configurations have been developed for linear electrode arrays including a modification of the adjacent electrode drive configuration (Powell H.M. Barber D.C. and Freeston I.L., 1985). The Wenner measurement strategy is commonly used in geophysical subsurface surveying and a detailed description is given by Griffiths D.H. and Barker R.D., 1993.

2.4 EIT Data Acquisition System

One of the earliest EIT data acquisition system was that of Henderson R.P. and Webster J.G., 1978. The system used voltage drive/current measurement and was used in a EIT system designed for imaging the thorax. The system used one large electrode connected to a voltage source that was placed on the chest of the subject. A 12×12 array of electrodes (144 in total) were place on the back of the subject and current measurements recorded. Kim.Y. And Woo H.W., 1987 also developed a voltage drive system for imaging the thorax. It consisted of 192 electrodes arranged on a belt which the subject wears around the chest. The function of each electrode is controlled by a computer. Each electrode could be set to apply a voltage to the subject, or take a current measurement or be logically disconnected from the system. The level of the voltage applied to the electrode was controlled by a digital to analogue converter. Current measurements were recorded using a current to voltage converter connected to the electrode, which was in turn connected to the host computer via an analogue to digital converter.

The "Sheffield Data Collection System" (Brown B.H. and Seagar A.D., 1987) was specifically designed for medical imaging and uses the adjacent electrode drive configuration (see Section 2.3.1). The system uses a single constant current source to inject a 51kHz 5mA peak sinewave into a pair of adjacent electrodes chosen from a set of 16. The differential voltages are recorded by a single differential amplifier which is switched between successive adjacent pairs of non-current carrying electrodes in turn using a multiplexer. The amplified signals are demodulated using a phase sensitive detector and 12bit ADC converts the detector output which is read by the controlling computer. The system acquires a new data set every 79ms.

Hutchison J.M.S. and Kulkarni V., 1995 describe a system based on the "Sheffield Data Collection System". It differs from the Sheffield system in that a fast ADC (8μs) is used to sample the amplified sinewave voltages directly. Sampling the voltage directly allows for the removal of the hardware demodulator. The demodulation of the acquired waveforms into the in-phase and quadrature components is carried out in software. The system samples each voltage for a period of 128μs, which with the frequency used (31.25kHz) gives 4 samples per complete cycle of applied current. The system acquires a complete data set in 62.5ms.

Daniels A.R., 1996, designed an EIT system to be used in conjunction with an optical tomography system for monitoring multi-component flow(s). It is similar to the Sheffield systems in that it has a single current source, which is multiplexed between electrode pairs. This system uses multiple amplifiers connected between all adjacent electrode pairs in order to increase data collection rates as was suggested by Brown B.H. and Seagar A.D, 1987. The amplified differential A.C. signals were converted to an equivalent r.m.s. voltage using a r.m.s. to dc converter. The system was designed for 16 electrodes and there were therefore 16 differential amplifier/r.m.s to dc converter channels. The frequency used was 8kHz and the current source gave a constant current output of 5mA; a complete data set could be acquired in 48ms.

Newell J.C., Gisser D.G. and Isaacson D., 1988 designed a 32 electrode system which utilised the adaptive measurement strategy. The system consisted of multiple current sources. The output magnitude of each current source's A.C. output was controlled using a 12 bit DAC (giving 2^{12} = 1024 levels) and the voltage measured using a precision voltmeter. The output from the precision voltmeter is converted to a digital signal and fed to the controlling computers I/O ports.

Wang W., Brown B.H. and Barber D.C., 1995 developed a multi-frequency system working in the frequency 9.6kHz to 1.2MHz at eight discrete steps. The system produces dynamic images using the lower frequency as the reference. The current magnitude of the system is set to 2mA pk-pk.. Also Griffiths H and Zhang Z, 1989 produced a system for collecting EIT data using 2 separate frequencies (40.96kHz and 81.92kHz) that also imaged the changes between the measured data sets.

Simpson J.C., Tozer R.C. and Freeston I.L., 1996, present a fully parallel EIT system. The system uses multi-frequency current sources to drive 3mApk at 15 different frequency simultaneously between adjacent drive electrodes. The current sources operate over a frequency range of 20-27 kHz each separated by 500Hz. The system records voltages from separate voltage measuring electrodes. The electrodes are configured as a linear array with 17 voltage measuring electrodes. The voltage on each electrode is buffered and subtracted to form 16 potential differences between adjacent voltage measuring electrode pairs. The total time required to measure all voltages required to produce an image is 2ms.

A system that uses a single channel current source and a single channel voltage measurement circuit is classed as a serial system, whereas a system which uses multiple current sources or voltage measurement circuits is classed as a parallel system. The design of the individual elements that make up an EIT data acquisition system can be considered a separate area of development. The aim of the work presented in this thesis was to make significant contribution to the application of EIT. A more detailed description on designing EIT data acquisition systems is documented in Webster J.G., 1990 and Wang M., Dickin F.J. and Beck M.S., 1992.

2.5 Reconstruction Algorithms

This section describes the three most common reconstruction algorithms used in EIT. The function of the reconstruction algorithms is to take the measurements recorded by the EIT data acquisition system (see chapter 2.4) and produce an image of the impedance distribution i.e. solve the inverse problem of Figure 2.2.2.

This is however not a complete review of all the reconstruction algorithms developed for EIT; Such a detailed review is not relevant to the design of a distributed planar pressure sensor.

The reconstruction algorithms used in EIT will produce one of two possible image types:

(1) Algorithms which produce static images i.e. produce actual conductivity or resistance values for each pixel in the reconstructed image from a single set of voltage/current measurements (Nordan M.J., 1995).

(2) Algorithms that produce image of changes in resistance between a reference data set and the last recorded data set.

2.5.1 Backprojection between Equipotential Lines

This is a single pass reconstruction algorithm that uses the principle that the lines of equipotential are traced between the voltage measuring electrodes and the source electrodes (electrodes 2,3,4,5,6 and 7 in Figure 2.5.1.1) any change in impedance in the cross-section between the electrodes (i.e. area A in Figure 2.5.1.1) produces a change in voltage at the electrodes (Barber D.C., Brown B.H. and Freeston I.L, 1983). Figure 2.5.1.1 shown the general principle.

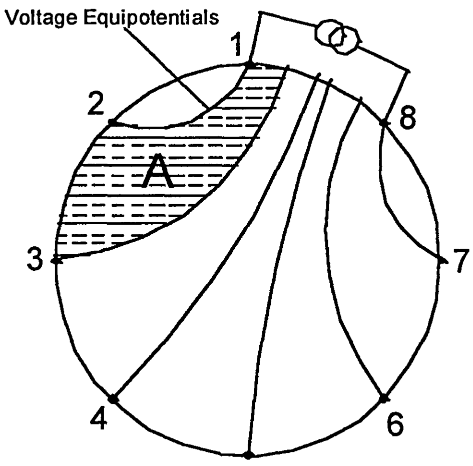


Figure 2.5.1.1 Voltage Equipotentials developed in a circular object when constant current is injected between boundary electrodes 1 and 8

Barber D.C., Brown B.H. and Freeston I.L, 1983 made the assumption that the resistivity of the entire region was initially uniform and that the equipotentials remain a constant shape. The steps in the backprojection between equipotential lines reconstruction algorithm are:

1. Assume an initial uniform resistivity (ρ_o).

2. Calculate the differential voltages that would appear at the electrodes when a constant current is passed between two of the electrodes for the assumed uniform resistivity.

3. Measure differential electrode voltages on the actual object whose resistivity distribution is to being imaged (i.e. voltage between electrodes).

4. Amend the resistivity of the region between the equipotentials ending at the electrodes by the ratio of measured to calculated differential voltages (equation 2.5.1.1).

$$\rho = \rho_0 \times \frac{Vm}{Vc} \qquad (2.5.1.1)$$

5. Repeat steps 3 and 4 for all differential electrode voltages for given current source excluding the electrodes connected to the constant current source (electrodes 1 and 8 in Figure 2.5.1.1).

6. Move the current source to each set of electrodes in turn (i.e. 8 and 6, to 2 and 1) and repeat steps 2 to 5.

7. Sum together all the images for each current source location to obtain final image.

Barber D.C. and Brown B.H., 1987 describes a one step implementation of the above method and a variation of this has been used to image change in resistivity between one data set and another by Daniels A.R., 1996.

2.5.2 Single Pass Backprojection By Sensitivity Coefficients

The backprojection by sensitivity coefficients reconstruction algorithm is reported by Kotre C.J., 1989 which is based on work by Tarassenko L. and Rolfe P., 1984 and produces dynamic images. Kotre's one pass algorithm is shown to be stable and not suffer from the stability and convergence problems that can be critical in iterative algorithms (Kotre C.J., 1993).

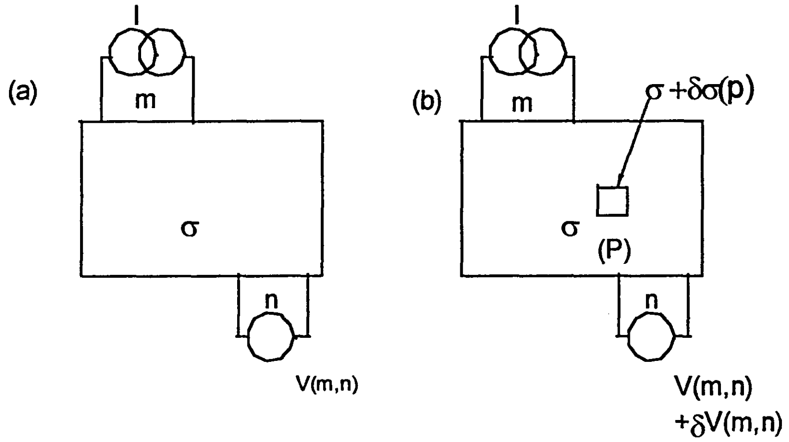


Figure 2.5.2.1 (a) An Object Having Uniform Conductivity (b) An Object Where An Area (P) Representing A Single Pixel In The Reconstructed Image Has Changed

Figure 2.5.2.1(a) shows an object with uniform conductivity; a constant current I is injected between the mth electrode pair and the voltage $V_{(m,\ n)}$ developed between the nth electrode pair is measured. Figure 2.5.2.1(b) shows the same situation as Figure 2.5.1(a) except that a small area P which represents one pixel in the reconstructed image has a conductivity of σ+δσ(p). The voltage measured between the nth electrode pair is now V(m,n)+δV(m,n). If the change in conductivity δσ within the area P is small, the current density can be assumed the same as for the uniform case. Therefore, the change in measured voltage is proportional to the change in conductivity equation 2.5.2.1

$$\delta V(m,n) \propto \delta\sigma(P) \tag{2.5.2.1}$$

If the constant of proportionality between δV(m,n) and δσ(p) is defined by a sensitivity coefficient S(p,m,n,) then

$$S_{(p,m,n)} = \frac{\delta V_{(m,n)}}{\delta\sigma_{(p)}} \tag{2.5.2.2}$$

The sensitivity coefficient for the area defined by P can be calculated as the inner product of the field produced by constant current flow in the source electrode pair (the mth pair of electrodes) and the field produced by a unit current flow in the voltage sensing electrodes (the nth pair) as shown in equation 2.5.2.3 (Kotre C.J., 1989)

$$S_{(p,m,n)} = \int_a \nabla\phi_m \cdot \nabla\phi_n \cdot da \tag{2.5.2.3}$$

where ϕ_m and ϕ_n are the potential fields and the area of integration is over the area (P). Kotre's single pass reconstruction algorithm used the case of a slight change δσ to one pixel in a homogeneous conductivity in calculating the potential fields.

The sensitivity coefficients can be estimated from experimental measurements Yu Z.Z., Peyton A.T., Beck M.S., Conway W.F. and Xu L.A., 1993, see Chapter 3.4 for details. The value of the pixels in the reconstructed image can be related to the sensitivity coefficients by equation 2.5.2.4 (Kotre C.J., 1989).

$$P_{(p)} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} S_{(p,m,n)} \times \left(\frac{\delta V_{(m,n)}}{V_{(m,n)}}\right)}{\sum_{m=1}^{M}\sum_{n=1}^{N} S_{(p,m,n)}} \tag{2.5.2.4}$$

Kotre C.J., 1995, 1996 extended the above reconstruction algorithm to image in three dimensions using an array of orthogonal linear electrodes. Images from simulations representing sub-surface planar networks of pipes, vessel or walls are presented in this work. Also presented are images of the difference between inspiration and full expiration of the human lungs.

2.5.3 Modified Newton-Raphson Reconstruction Algorithm

The Modified Newton-Raphson (MNR) reconstruction algorithm produces static images and solves the inverse EIT problem iteratively. It is one of the most theoretically orientated reconstruction algorithms (Yorkey T.J., 1986). In the MNR reconstruction algorithm, finite element methods replace the distributed Laplace equations with a set of linear algebraic equations. These linear equations are used to calculate the voltages that would appear on the EIT electrodes for a resistivity distribution ρ. The MNR algorithm iteratively updates the resistivity distribution to reduce the error between the measured and calculated voltages. The measure of error between the measured and calculated voltages is known as the objective function φ and is define in equation 2.5.3.1

$$\varphi = \frac{1}{2}\left[V_{cal} - V_{meas}\right]^{T} \times \left[V_{cal} - V_{meas}\right] \tag{2.5.3.1}$$

where V_{cal} is the vector of calculated voltages for a resistivity distribution ρ, V_{meas} is the vector of measured voltages and T means the transpose. The minimum value of φ

occurs when its derivative with respect to resistance change is zero. Thus, from equation 2.5.3.1

$$\varphi(\rho)' = \left[V_{cal_{(\rho)}}'\right]^T \times \left[V_{cal} - V_{meas}\right] = 0 \qquad (2.5.3.2)$$

where $[V_{cal}'{}_{(\rho)}]_{i,j} = \frac{\partial V_{cal\,i}}{\partial \rho_j}$ and is known as the Jacobian matrix

The Taylor series expansion of equation 2.5.3.2 about a point ρ^k (keeping the linear terms) gives equation 2.5.3.3

$$\varphi'\left(\rho^{k+1}\right) \approx \varphi'\left(\rho^k\right) + \varphi''\left(\rho^k\right)\Delta\rho^k = 0 \qquad (2.5.3.3)$$

where $\rho^{k+1} = \rho^k + \Delta\rho^k$.

The second order derivative of φ (φ") in equation 2.5.3.3 is called the Hessian matrix ***H*** and is defined in equation 2.5.3.4

$$H = \varphi''\left(\rho^k\right) = \left[V_{cal}'\right]^T \times \left[V_{cal}'\right] + \left[V_{cal}''\right]^T \times \left\{I \otimes \left[V_{cal} - V_{meas}\right]\right\} \qquad (2.5.3.4)$$

where ⊗ is the Kronecker matrix product. The second derivative appearing in the RHS of equation 2.5.3.4 is very small and can be ignored. Equations (2.5.3.4) and (2.5.3.2) are substituted into equation (2.5.3.3) and the result set to zero. The resulting equation is re-arranged to give the update for the resistivity distribution as shown in equation (2.5.3.5).

$$\Delta\rho^k = -\left\{\left[V_{cal}'\right]^T \times \left[V_{cal}'\right]\right\}^{-1} \times \left[V_{cal}'\right]^T \times \left[V_{cal} - V_{meas}\right] \qquad (2.5.3.5)$$

The MNR algorithm iterates until the error φ reduces to a minimum using the update equation (2.5.3.5). At each iterative step the resistivity distribution is updated by equation (2.5.3.6) shown below.

$$\rho^{k+1} = \rho^{k} + \Delta\rho^{k} \qquad (2.5.3.6)$$

The basic steps in the MNR reconstruction algorithm are thus:

1. Guess an initial resistivity distribution ρ.
2. Calculate the boundary voltages using FEM techniques
3. Calculate the object function φ (equation (2.5.3.1)).
4. If φ is greater than the desired reconstruction error determine $\Delta\rho^{k}$ using equation (2.5.3.5), update the resistivity distribution using equation (2.5.3.6) and return to step 2.
5. If φ is less than the desired error, stop the algorithm and use the last estimate of the resistivity distribution as the reconstructed image.

The above description of the MNR algorithm is taken for Webster J.G, 1990 and Nordin M.J., 1995. Weng Wah Loh and Fraser Dickin, 1996, describe improvements to the MNR algorithm. Their improved MNR algorithm implements the algorithm using sparse matrix methods and avoids redundant computations. These reduce the overall computational complexity of the original algorithm of Yorkey by a factor of five.

2.6 Summary of Chapter

In this chapter the theoretical background of EIT has been presented. Also presented are the EIT data acquisition systems and reconstruction algorithms commonly used in EIT. The review of EIT data acquisition systems shows that the majority of systems are designed around current drive and voltage measurement with only a small number using voltage drive and current measurement. A system can be classed as either a serial system or a parallel system. The serial systems having single channel drive and measurement circuits that are switched to the appropriate electrodes (as defined in the measurement strategy employed) and the parallel systems have either parallel drive or measurement circuitry.

The reconstruction algorithms can be classified in two ways; those that produce static images (images giving true conductivity values i.e. modified Newton-Raphson) and those that produce dynamic images(images between one data set and another i.e. backprojection between equipotential line). The reconstruction algorithms which produce static images are more computationally intensive than the algorithms that produce dynamic images. Due to the computational overhead of the static image reconstruction algorithms, the dynamic algorithms are more commonly used.

3. Electrical Impedance Tomography System Design

3.1 Introduction

In this Chapter, the design of the EIT system hardware and software is presented along with the details of the reconstruction algorithm used in the work. The chapter is divided into three main sections; Sections 3.2 and 3.3 detail the EIT systems hardware and software design respectively while Section 3.4 details the reconstruction algorithm used.

3.2 Hardware design

A serial EIT data acquisition system has been chosen as the design criteria for the prototype system (see Chapter 2.3 for a review of serial and parallel EIT systems) as it provided the simplest solution. The serial current source has a frequency range of approximately 5.7kHz to 36.4kHz in steps of 5kHz and a current magnitude adjustable between 1mA and 4mA. The frequency is controlled by the EIT system software (see Section 3.3 below) while the current magnitude is adjusted manually (via a trim pot.). A higher value for the upper limit on the frequency range (1MHz) would make the system more versatile for the purposes of researching the fundamental physics of EIT and the sensor element; however, this was not possible due to the cost of such a unit. The amplifier and demodulator section uses a two stage programmable differential amplifier with the gain adjusted by the controlling software to always give the maximum possible output. This is done in order that the SNR for any given reading is optimised. The demodulator is a RMS to DC converter with pre-filtering at frequencies below 1kHz in order to remove mains and any other low frequency interference. Figure 3.2.1 shows the block diagram of the EIT system developed.

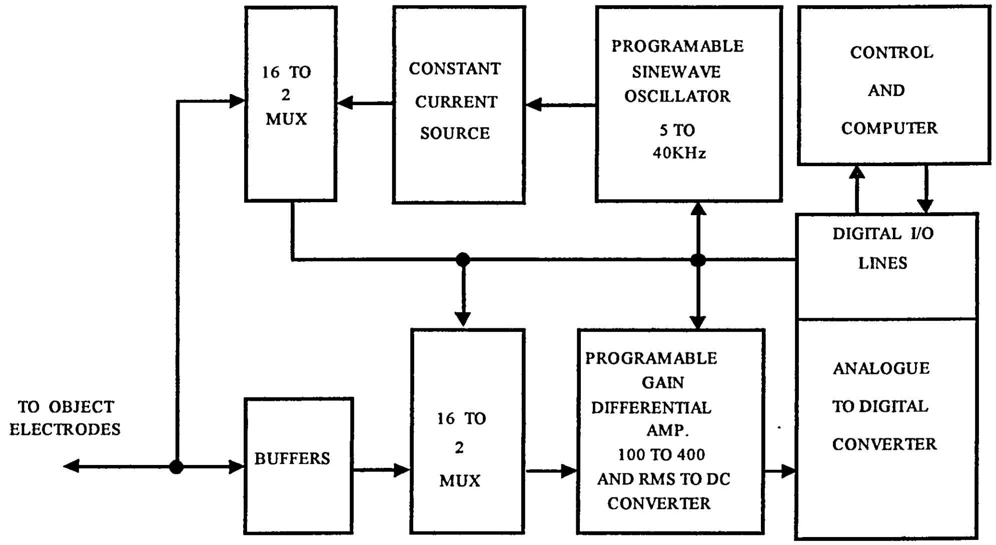


Figure 3.2.1 Block Diagram of Serial EIT System.

Both the digital I/O lines and the analogue to digital converter are part of a PC interface card (ADC-42) by Blue Chip Technology and a 33MHz 486 based PC is used as the controlling computer. All other blocks form part of the designed EIT data acquisition system.

3.2.1 Sinewave oscillator

For the sinewave oscillator a monolithic integrated circuit ICL8038 by Harris Semiconductors (data sheet file number 2864) capable of producing sinewaves with high linearity (0.1%) and low distortion (0.5%) is used. The ICL8038 requires minimum external components as shown in Figure 3.2.1.1.

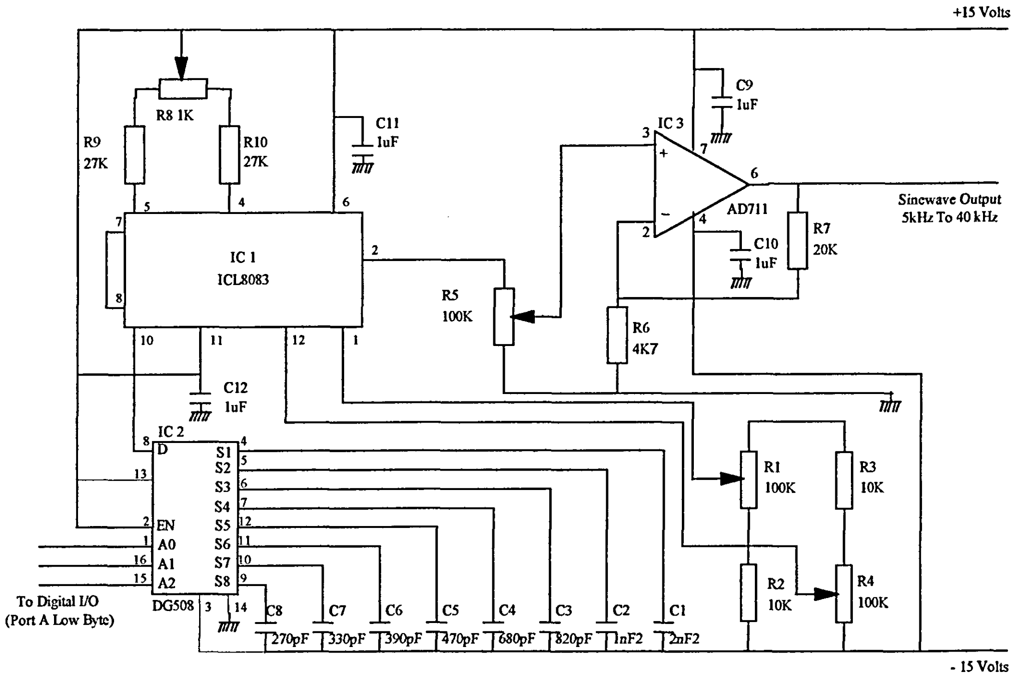


Figure 3.2.1.1 Variable Frequency Sinewave Oscillator.

The frequency of oscillation is selected by the analogue multiplexer IC2 switching it's input to the appropriate external capacitor (C1 to C8). Capacitors C1 to C8 and resistors R9 and R10 set the frequency of oscillation according to equation 3.2.1.1

$$F_0 = \frac{0.33}{R_O C_O} \qquad (3.2.1.1)$$

where Fo = frequency of oscillation, Ro is the value of both R9 and R10, and Co is C1 to C8

There are constraints on the values of Ro as it sets the charging current for the external timing capacitor Cn (n = 1 to 8). Optimum performance is obtained with charge currents between 10uA to 1mA. The magnitude of the charging current due to Ro can be calculated from equation 3.2.1.2.

$$I = \frac{0.22(V^+ - V^-)}{R_O} \qquad (3.2.1.2)$$

where I = charging current, V^+ = positive supply voltage and V^- = negative supply voltage.

The values of R9 and R10 are set at 27kΩ giving a charging current magnitude of 0.24mA, which is within the value for optimum performance. Using equation 3.2.1.1, the values of Co for the required frequencies (5KHz to 40kHz) were calculated. Table 3.2.1.1 lists the value of Co for a given frequency along with the nearest preferred value used, the frequency of oscillation measured using a Hameg HM205-3 oscilloscope and the percentage difference between the desired and measured frequencies. The actual calculated values for Co could have been realised by a combination of fixed value and trimming capacitors. However, only a range of frequencies between 5kHz and 40 kHz is required. Therefore, the frequencies obtained using the nearest preferred values for Co were deemed sufficient for the EIT system.

Required Frequency (kHz)	**Co Calculated Value**	**Co Nearest Preferred Value**	**Measured Frequency (kHz)**	**Δ% Error**
5	2.44nF	2nF2	5.7	14.0
10	1.22nF	1nF2	10.0	0.0
15	814.81pF	820pF	14.7	-2.0
20	611.11pF	680pF	17.5	-12.5
25	488.89pF	470pF	25.0	0.0
30	407.41pF	390pF	28.6	-4.7
35	349.21pF	330pF	33.3	-4.9
40	305.56pF	270pF	36.4	-9.0

Table 3.2.1.1 Co and Oscillator Frequency

From Table 3.2.1.1, it can be seen that the highest percentage difference (error) between the required and actual frequencies is at 5kHz and is 14%. However there is a -8.33% difference in the value of Co for this frequency. In addition, the resistors and capacitor used have component tolerances of ±5% and ±10% respectively and therefore the actual frequency obtained is within expected limits given the components used. Table 3.2.1.2 lists the calculated values for Fo using the nearest preferred values for Co and also the maximum and minimum frequencies given the tolerances for Ro and Co. It is assumed that the on-resistance of the multiplexer IC2 is zero.

Calculate Frequency (Fo) Form Co Preferred Value (kHz)	Minimum Fo due to Co and Ro component Tolerances (kHz)	Maximum Fo due to Co and Ro component Tolerances (kHz)
5.55	4.81	6.50
10.19	8.82	11.91
14.91	12.90	17.43
17.97	15.56	21.02
26.00	22.51	30.41
31.34	27.13	36.65
37.04	32.06	43.32
45.27	39.19	52.94

Table 3.2.1.2 Oscillator Frequency Variations Due To Component Tolerances

Form Table 3.2.1.2 it can be seen that the values of the measured frequencies given in Table 3.3.1.1 are within the range expected except for the value for Co = 270pF. In the case of Co = 270pF the lower limit is at 39.19kHz whereas the value measured is at 36.4kHz. The difference is considered to be due to the multiplexer's channel-on resistance and capacitance. No further experiments to determine the exact causes was undertaken, as the range of frequencies obtained was considered sufficient for the EIT system being developed.

Resistor R1, R2 R3 and R4 are used to reduce the sinewave distortion and the configuration shown is reported by the manufacture's data sheet (Harris Semiconductors data sheet file number 2864) to give a sinewave distortion of close to 0.5%. Circuit components IC3, R5, R6 and R7 provides buffering, gain and amplitude adjustment. It is the adjustment of the oscillator's output, which controls the magnitude of the current from the voltage controlled current source (see below).

3.2.2 Voltage Controlled Current Source

Two voltage controlled current sources (VCCS) were investigated for use in the EIT system. The Howland one op-amp current source (Horowitz and Hill, 1987) as shown in Figure 3.2.2.1 and a three op-amp current source based on the design by Newell C .J., et al, 1988 as shown in Figure 3.2.2.2. (Page 33)

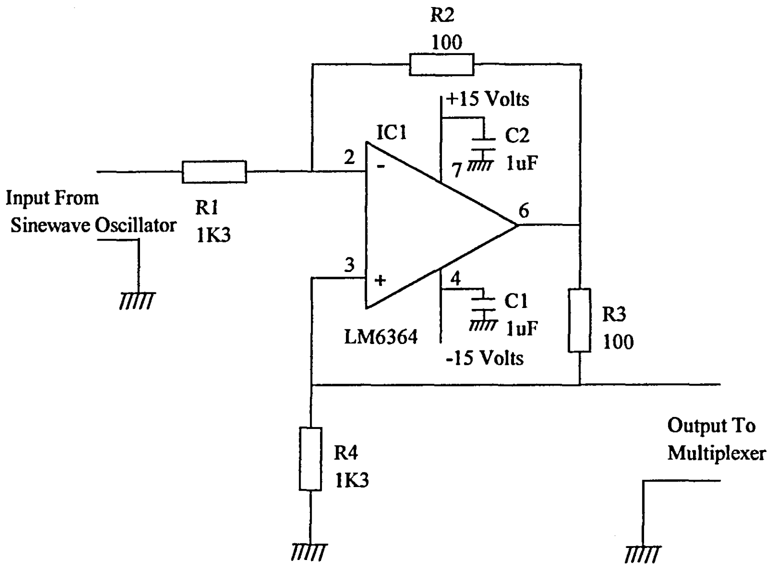


Figure 3.2.2.1 Howland 1 Op-amp Voltage Controlled Current Source (VCC1)

Assuming an ideal op-amp and matched resistors the load current of the Howland voltage controlled current source shown in Figure 3.2.2.1 is given by equation 3.2.2.1(Webster J.G., 1990)

$$I_L = \frac{-V_{in}R_2R_4}{R_1R_3R_4 + Z_L(R_1R_3 + R_2R_4)} \tag{3.2.2.1}$$

where I_L is the load current magnitude, Z_L is the load impedance and *Vin* is the VCCS input voltage. For a high output impedance and therefore good performance, R_1R_3 must equal R_2R_4 (Webster J.G., 1990) and equation 3.2.2.1 reduces to equation 3.2.2.2

$$I_L = \frac{-V_{in}}{R_4} \tag{3.2.2.2}$$

A modification to the Howland circuit (Nowicki D. J. and Webster J. G., 1989) incorporates compensation for both the op-amp gain and phase errors by adding a small variable resistor (50Ω) in series with R4. More importantly, it also incorporates capacitive compensation for the capacitance of the coaxial cables connecting the EIT system hardware to the EIT electrodes. This is achieved by adding a capacitor in parallel with the serial combination of R4 and the 50Ω variable resistor. The design is reported as giving 12bit accuracy at 50kHz for a load impedance of between 39Ω to 1039Ω.

However, the compensation system needs to be adjusted to the frequency being used and to the cables connecting the EIT hardware to its electrodes. As the current source of Nowicki D. J. and Webster J. G., 1989 needed to be adjusted to the frequency and cables being used. It was not a design that could be used with a system which is to operate over a range of frequencies.

Assuming the feedback path to be ideal and again matching resistor values the load current of the VCCS shown in Figure 3.2.2.2 is given by equation 3.2.2.3

$$IL = \frac{-(Vin + Va) - VL}{Z4} \qquad (3.2.2.3)$$

where Vin is the VSSC input voltage, *VL* is the load voltage, *Va* is the feedback path voltage and Z4 is the impedance of the resistor/capacitor combination R4 and C1.

The circuit of Figure 3.2.2.2 is designed such that *Va* is equal to -*VL*. If this is achieved then equation 3.2.2.3 reduces to equation 3.2.2.4

$$IL = \frac{-Vin}{Z4} \qquad (3.2.2.4)$$

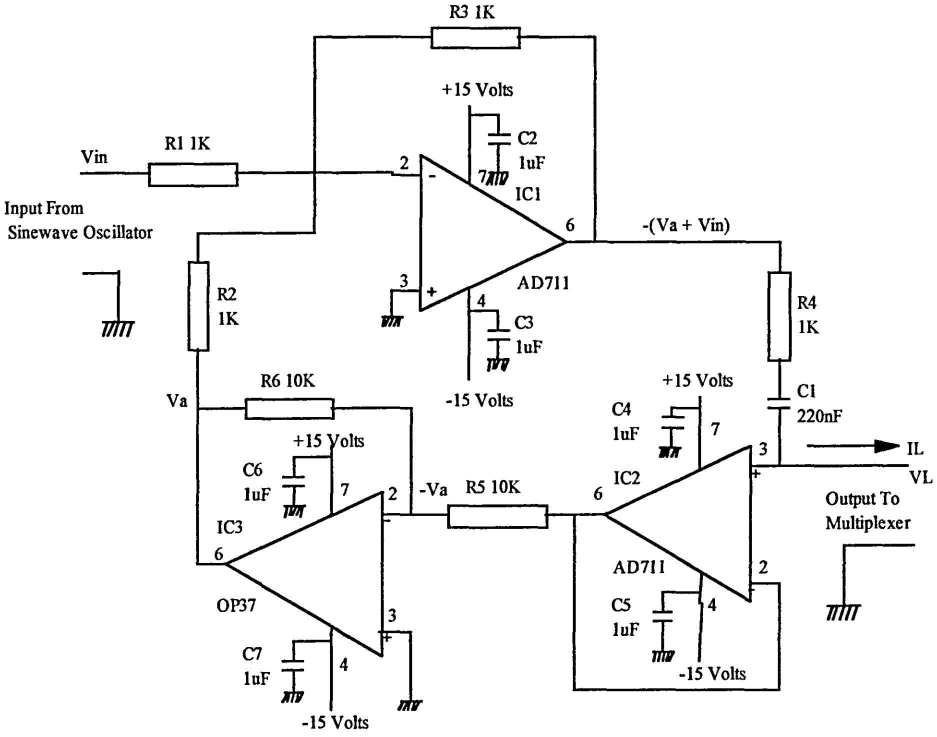


Figure 3.2.2.2 Voltage controlled current source (2)

The multiplexing and driven shield circuitry affect the performance of an EIT system's current source i.e. the on-channel resistance and capacitance of the multiplexers are in series with any load. The driven shield reduces the capacitance of the coaxial cables (assuming a 1 metre coaxial cable it's effective capacitance is reduced from 100pF to 1 pF; Webster J.G, 1990). Therefore testing the performance of the current sources was done with these circuit elements in place as shown in Figure 3.2.2.3

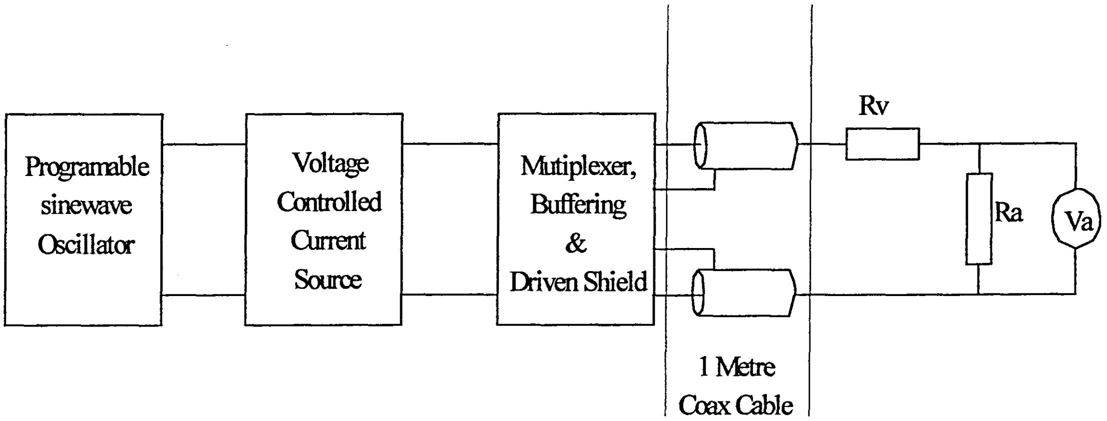

Programable sinewave Oscillator
Voltage Controlled Current Source
Mutiplexer, Buffering & Driven Shield
Rv
Ra
Va
1 Metre Coax Cable

Figure 5.2.2.3 Diagram of Apparatus for Current Source Performance Tests

In order to determine the range of loads for which the EIT system would operate correctly a circular tank 100mm in diameter and 40mm high was filled with saline solution with a conductivity of 100mS to a height of 20 mm. The tank has sixteen equally spaced electrodes on the periphery, which run the full height of the tank. Circular insulating and conducting anomalies with a diameter of 86mm (86% of the tank diameter) and full tank height were placed in the tank. The insulating anomalies are made from turned wood sprayed with paint and the conducting anomalies are made from mild steel. The above anomalies are used to give the maximum contrast between the conductivity of the saline solution and the anomaly conductivity. This will therefore give the maximum conductivity variation seen between two of the electrodes.

Therefore, the maximum conductivity change with which the current source has to content with.

The resistance between two adjacent electrodes was measured using a Hewlett Packard 4284A impedance analyser over a range of frequencies (20Hz to 1MHz). The impedance measured between the adjacent electrodes with the insulating and conducting anomalies present over the range of frequencies used is show in Table 3.2.2.1. along with the percentage differences between the two readings.

Frequency (Hz)	Impedance with Insulating Object (Ohms)	Impedance with Conducting Object (Ohms)	Percentage Difference
20	436.00	400.01	9.00
200	180.17	158.03	14.01
2k	83.06	44.25	87.71
5k	71.14	36.44	95.23
10k	66.26	32.10	106.47
15k	64.63	30.74	110.25
20k	63.86	30.11	112.09
25k	63.40	29.76	113.04
30k	63.11	29.53	113.71
40k	62.75	29.27	114.38
50k	62.54	29.12	114.77
100k	62.16	28.84	115.53
500k	61.78	28.59	116.09
1M	61.44	28.50	115.59

Table 3.2.2.1 Impedance Verses Frequency with Insulating and Conducting Anomalies

From Table 3.2.2.1 it can be seen that with the conducting anomaly the impedance between the two electrodes with frequency reduces more than with the insulating anomaly.

Initially the two voltage controlled current sources were tested at 5.7kHz and 36.4kHz which are the upper and lower limit to the range of frequencies generated by the oscillator stage of the design. A resistor having a value of 33Ω was used for Ra and a variable resistor 0Ω to 470Ω used for Rv. This variation in resistance tests the current source over a much larger resistance variation than that determined by Table 3.2.2.1. The actual variation in resistance was from 34.4Ω to 554.8Ω which was measured using a FLUKE 8010A multimeter. The current magnitude was set to approximately 3mA and the voltage across Ra measured using a Fluke 8050A multimeter, while Rv was varied; the results are presented in Table 3.2.2.2.

	V across Ra @ Load = 33.4Ω	**V across Ra @ load =554.8Ω**	**I_L @ load = 33.4Ω**	**I_L @ load = 554.8Ω**	**%ΔI_L**
VCCS 1 @ 5.7kz	101.742mV	101.782mV	3.0462mA	3.0474mA	0.0394
VCCS 2 @ 5.7kHz	101.830mV	101.842mV	3.0488mA	3.0492mA	0.0131
VCCS 1 @ 36.4kHz0	94.806mV	94.977mV	2.8385mA	2.8436mA	0.1797
VCCS2 @ 36.4kHz	102.600mV	102.660mV	3.0719mA	3.0736mA	0.0553

Table 3.2.2.2 Current Source Load Variation Tests

From Table 3.2.2.2 it can be seen that the performance of VCCS 2 (3 op-amp design) due to the resistance variations is approximately three time that of VCCS 1 (Howland 1 op-amp design). In addition, the change in current magnitude between the upper and lower frequencies with a constant load resistance of 33.4Ω are 7.317% for VCCS 1 and 0.75% for VCCS 2. Therefore VCCS 2 was chosen for the EIT system although its performance with the load variation used (33.4Ω to 554.8Ω) is only 0.0553% at 36.4kHz. However, the load variation used is 17 times higher than the highest expected load variation as given in Table 3.2.2.1. Therefore, the current source will give 12-bit accuracy (0.02%) over the load variations for which it is used.

The performance of VCCS 2 with a load variation of 33.4Ω to 554.8Ω over a frequency range of 5.7kHz to 36.4kHz and with the current magnitude set to 3mA is shown in Table 3.2.2.3. From Table 3.2.2.3, it can be seen that at 10kHz, 17.5kHz, 28.6kHz and 36.4kHz 12-bit accuracy is not achieved for the load variation used. The load variation is however 17 times higher then that expected. 12-bit accuracy will be achieved for the load variations for which the system is to be used.

Frequency kHz	V Ra @ load = 33.4Ω	V Ra @ load = 554.8Ω	I_L @ load = 33.4Ω	I_L @ load = 554.8Ω	%ΔI_L
5.7	101.831mV	101.843mV	3.0488mA	3.0492mA	0.013
10.0	103.089mV	103.125mV	3.0865mA	3.0876mA	0.035
14.7	102.380mV	102.403mV	3.0653mA	3.0660mA	0.022
17.5	102.708mV	102.744mV	3.0751mA	3.0762mA	0.034
25.0	102.642mV	102.653mV	3.0731mA	3.0734mA	0.011
28.6	102.680mV	102.727mV	3.0743mA	3.0757mA	0.044
33.3	102.778mV	102.801mV	3.0772mA	3.0779mA	0.021
36.4	102.601mV	102.660mV	3.0719mA	3.0737mA	0.055

Table 3.2.2.3 VCCS 2 Frequency and Load Performance Test Results

3.2.3 Buffering and multiplexing

In order to switch both the single current source and the measurement circuitry to the required EIT electrodes a multiplexer system is used as shown in Figure 3.2.3.1. Four HI506 16 channel CMOS multiplexers are used in two banks of two. The first bank switches the current source to the electrodes while the second bank switches the differential voltages developed between the electrodes to the measurement circuitry.

Each electrode is connected to the measurement circuitry multiplexer via a unity gain buffer. The buffer output is also fedback to the screen of the cable connecting the electrode to the EIT electronics via a second unity gain buffer. Connecting the screen to the buffer output makes the potential of the screen equal to the potential of the inner core; virtually reducing the capacitance between the inner signal core and the outer screen of the cable.

The digital I/O lines from the PC data acquisition card control which pair of electrodes are connected to the current source and which pair are connected to the measurement circuitry. One eight-bit digital I/O port is required for each bank of multiplexers.

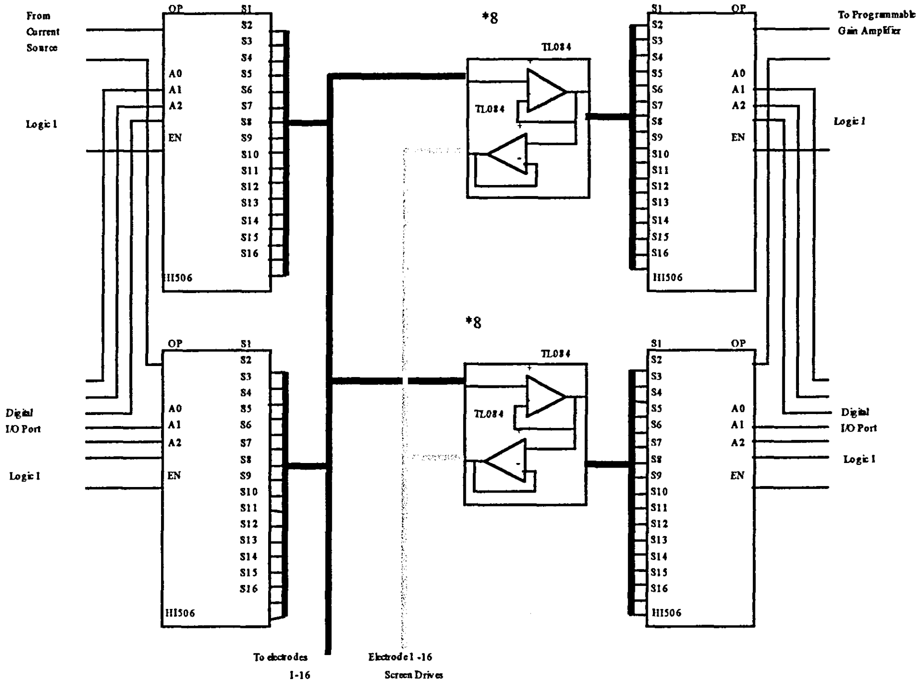


Figure 3.2.3.1 Multiplexing and Buffering

3.2.4 Differential Programmable gain amplifier

The differential programmable gain amplifier (DPGA) is a two-stage circuit design as shown in Figure 3.2.4.1. The first stage provides an ideal differential gain of 10 to the signals from the buffer/multiplexer stage of the EIT system (see section 3.2.3 above). The second stage is the PGA with ideal gains of 10, 20, 30 and 40, which are provided by an inverting amplifier (OP21). The gains are selected via the multiplexer (AD7590) switching in the appropriate resistance (R4, R5, R6 and R7). The overall gain of the DPGA is then simply the gain of the differential stage times the gain of the PGA stage alone as given by equation 3.2.4.1.

$$G = \left(\frac{2R1}{R2} + 1\right)\left(-\frac{R8}{Ra + Rm}\right) \qquad (3.2.4.1)$$

Where Ra = R4, R5, R6, or R7, and Rm = Multiplexer Channel Resistance (90Ω max.).

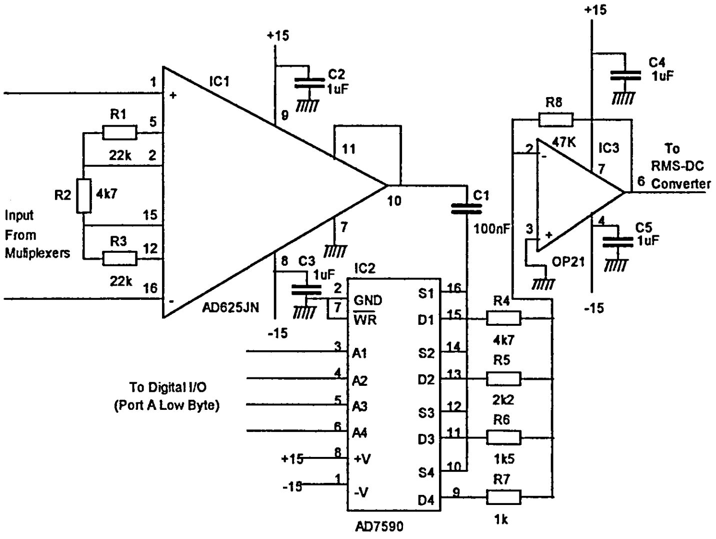


Figure 3.2.4.1 Programmable Gain Amplifier

Table 3.2.4.1 shows the actual gains achieved at a frequency of 40kHz and the maximum and minimum gains possible due to resistor tolerances (5%).

Desired Gain	Minimum Gain	Maximum Gain	Actual Gain at 40kHz
100	84	123	106
200	176	257	216
300	254	370	309
400	371	538	423

Table 3.2.4.1 Programmable Gain Amplifier Gain Performance

From Table 3.2.4.1 it can be seen that the actual gains achieved are all higher than those required but are within the range expected due to resistor tolerances. The software controlling the system allows for these changes by dividing the ADC output by the amplifiers current gain setting.

3.2.5 RMS. to D.C. converter

The A.C. signal from the differential amplifier is then converted to its RMS value using Analogue Devices AD536A RMS to DC converter. The AD536A converts the RMS level of the ac input signal to an equivalent DC output with a maximum error of 0.5%. This error is greater than 12 bit (0.02%) however as each measurement has the same conversion error the effects of this can be ignored. (Webster J.G., 1990). The circuit diagram of the RMS to DC converter is shown in Figure 3.2.5.1

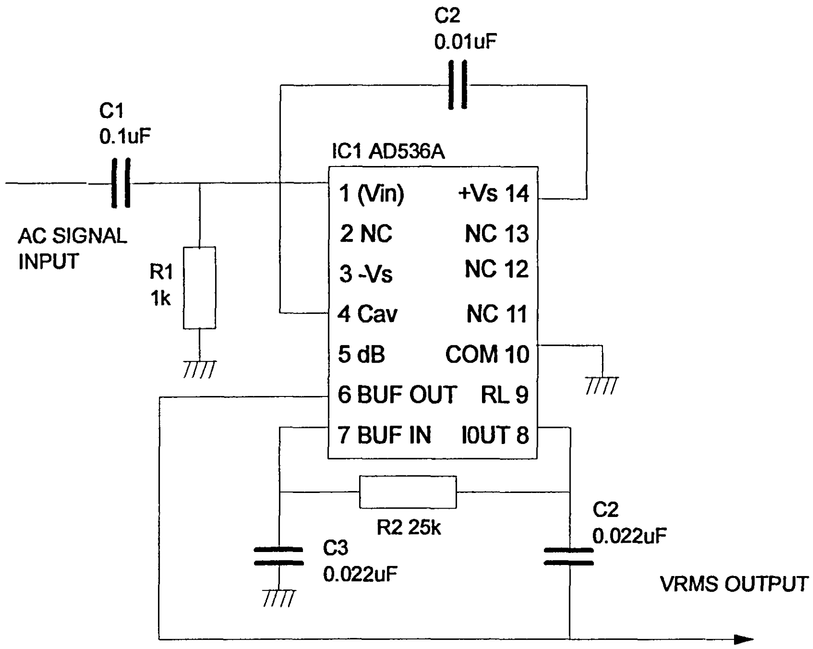


Figure 3.2.5.1 RMS To DC Converter

The capacitor-resistor combination C1 and R1 forms a simple first order high pass filter with a cut-off frequency of 1.6kHz to remove any DC level from the previous stage and also filters out any noise below 1.6kHz. Components C2, C3, R2 and the internal buffer of the AD536A form a 2 pole Sallen-Key output filter which reduces the conversion (averaging) error. With the component values used in the design and the AD536A design tables this design gives an expected percentage error of 0.1% ± 0.02% due to component tolerances at a frequency of approximately 5kHz and a settling time of 3.6ms.

As the frequency of the signal increases, the error reduces and therefore the system was tested at its lowest frequency of 5kHz. A signal of 5kHz with amplitude of 10V pk-pk was applied to the system and the output measured. This was compared with the RMS value measured using a true RMS meter. The difference between the two readings was 0.14%. This percentage error is 17% higher than that predicted by the design tables the difference between the expected error and that measured is most certainly due to stray capacitance within the circuit (the system was constructed on VEROBOARD).

The bandwidth of the AD536A increases with input voltage and is 300 kHz at Vin = 0.1Vrms and 2MHz at Vin => 1Vrms therefore the bandwidth of the RMS to DC converter design is well above the maximum frequency used (36.4kHz).

3.2.6 Measurement Circuitry Frequency Response

The complete measurement system was tested by applying a 15mV pk-pk A.C. signal to the system and recording the output voltages given by the system when the frequency is changed from 10Hz to 1MHz. The frequency response obtained from the system is shown in Figure 3.2.6.1

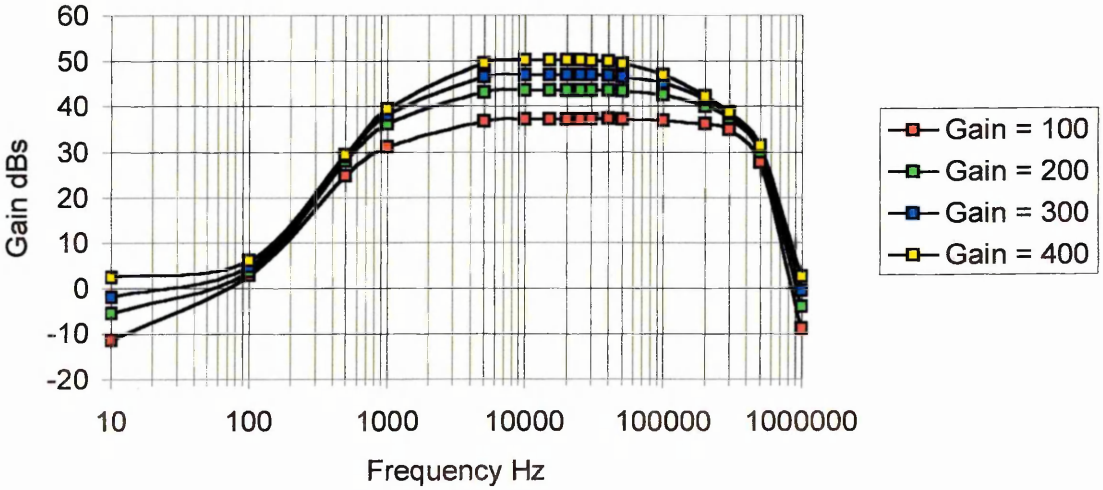


Figure 3.2.6.1 Frequency Response Of EIT System

3.3 Software Design

The software for the system is coded in Borland C and is a DOS based program. The software controls all aspects of the EIT system (excluding the magnitude of the constant current source). This section describes the functionality of the software and not the program structure (a full listing of the software is given in Appendix B). On running the software, the main menu screen as shown in figure 3.3.1 is seen. All the available options for the control of the system are shown on the right hand side of the screen and a description of each follows.

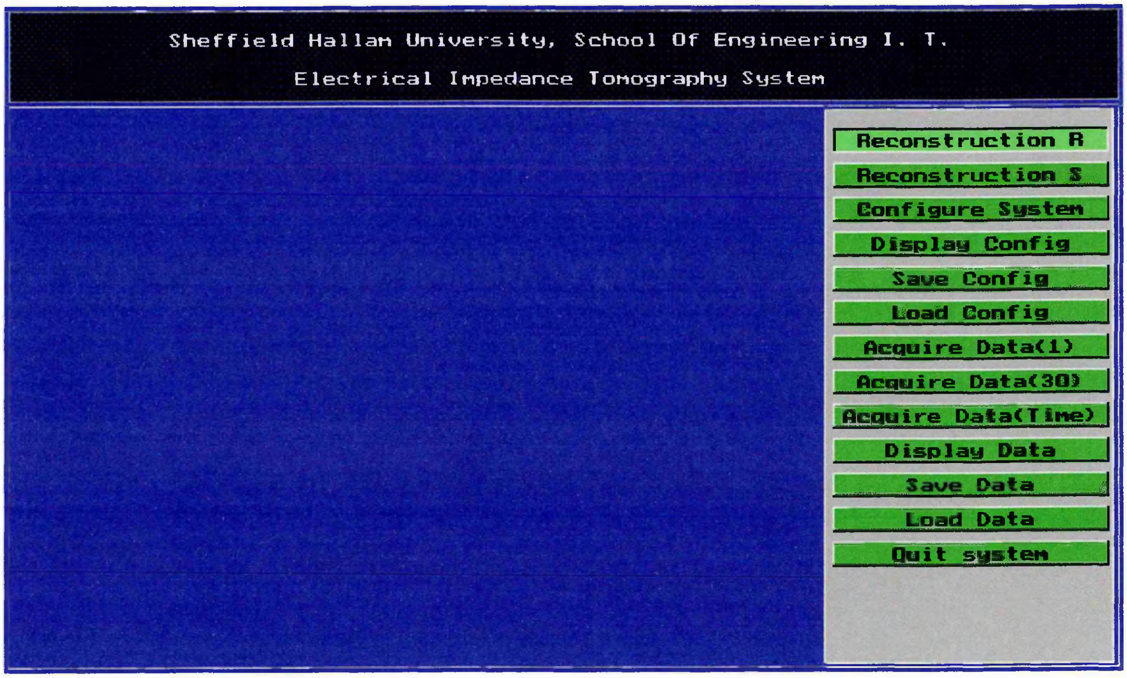


Figure 3.3.1 Main Menu Screen of Tomography Software

Reconstruction R: Enters menu to reconstruct image for a round phantom

Reconstruction S: Enters menu to reconstruct image for a square phantom

Configure System: Opens up a window to configure the frequency to be used and for each projection which electrodes are used for the current source and which pairs of electrodes the differential voltages are measured between.

Display Config: Opens up a window that shows the current configuration of the system (see Figure 3.3.2)

Save Config: Saves current configuration to disk.

Load Config: loads current configuration from disk.

Acquire Data (1): Acquires one complete set of voltage readings for each projection.

Acquire Data (30): Acquires an average of 30 complete sets of voltage readings for each projection.

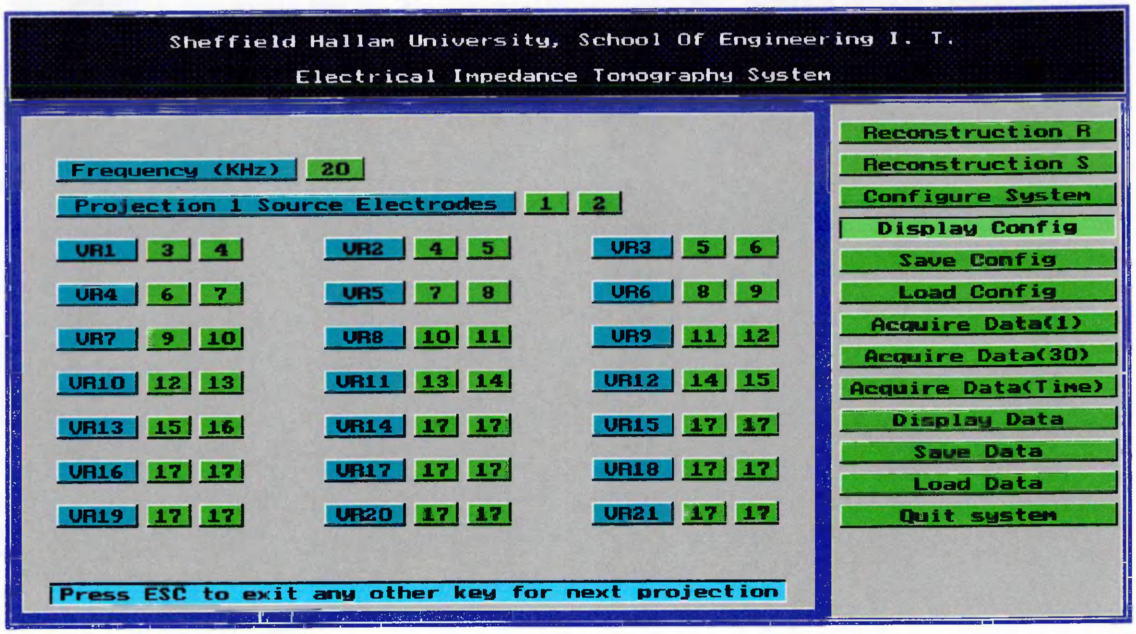
Acquire Data (Time): Allows the operator to set the system to acquire 1 to 60 complete sets of voltage readings with a time period of 1 to 60 seconds between each set.

Display Data: Displays the voltage readings as a bar graph on a projection bases (see Figure 3.3.3)

Save Data: Saves voltage readings to disk (file saved is in standard text format).

Load Data: Loads voltage reading from disk

Quit System: Shuts down programme and return to DOS.



Sheffield Hallam University, School Of Engineering I. T.
Electrical Impedance Tomography System

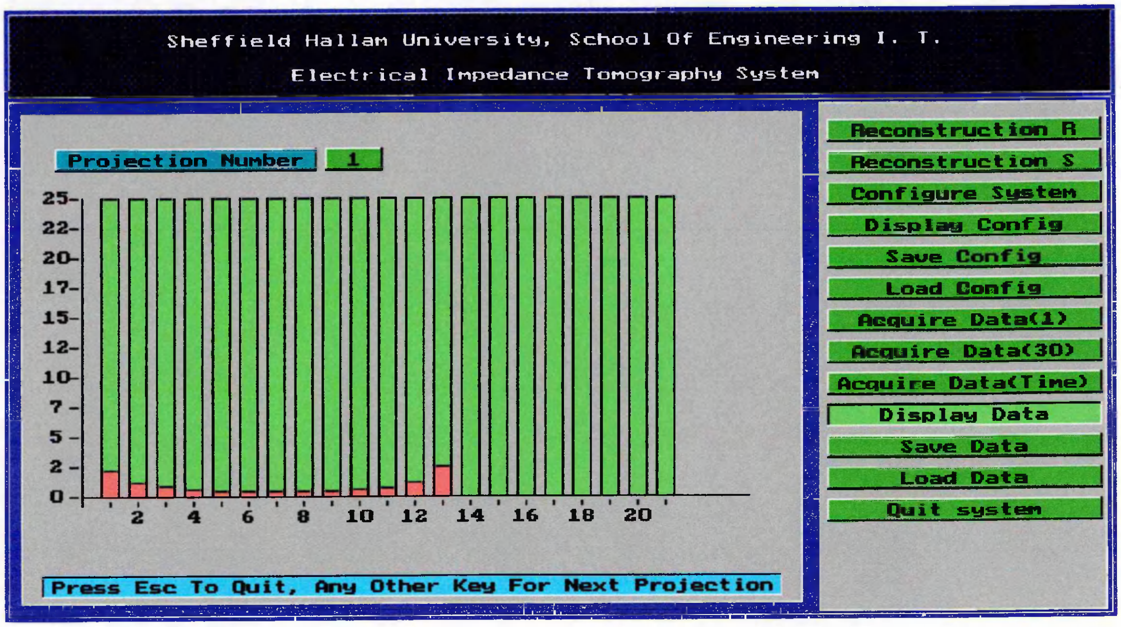
Frequency (KHz) 20

Projection 1 Source Electrodes 1 2

VR1	3	4	VR2	4	5	VR3	5	6
VR4	6	7	VR5	7	8	VR6	8	9
VR7	9	10	VR8	10	11	VR9	11	12
VR10	12	13	VR11	13	14	VR12	14	15
VR13	15	16	VR14	17	17	VR15	17	17
VR16	17	17	VR17	17	17	VR18	17	17
VR19	17	17	VR20	17	17	VR21	17	17

Press ESC to exit any other key for next projection

Figure 3.3.2 Display Config. Window Showing
Adjacent Electrode Drive Configuration for Circular Phantom (Projection 1)



Sheffield Hallam University, School Of Engineering I. T.
Electrical Impedance Tomography System

Projection Number 1

Press Esc To Quit, Any Other Key For Next Projection

Figure 3.3.3 Display Data Window Showing
Voltage Profile From Circular Phantom Using Adjacent Electrode Drive Configuration

On either selecting the “Reconstruction R” or Reconstruction “S” options from the main menu screen the menu options on the right hand side of the screen changes as shown in Figure 3.3.4. A description of all available menu options follows.

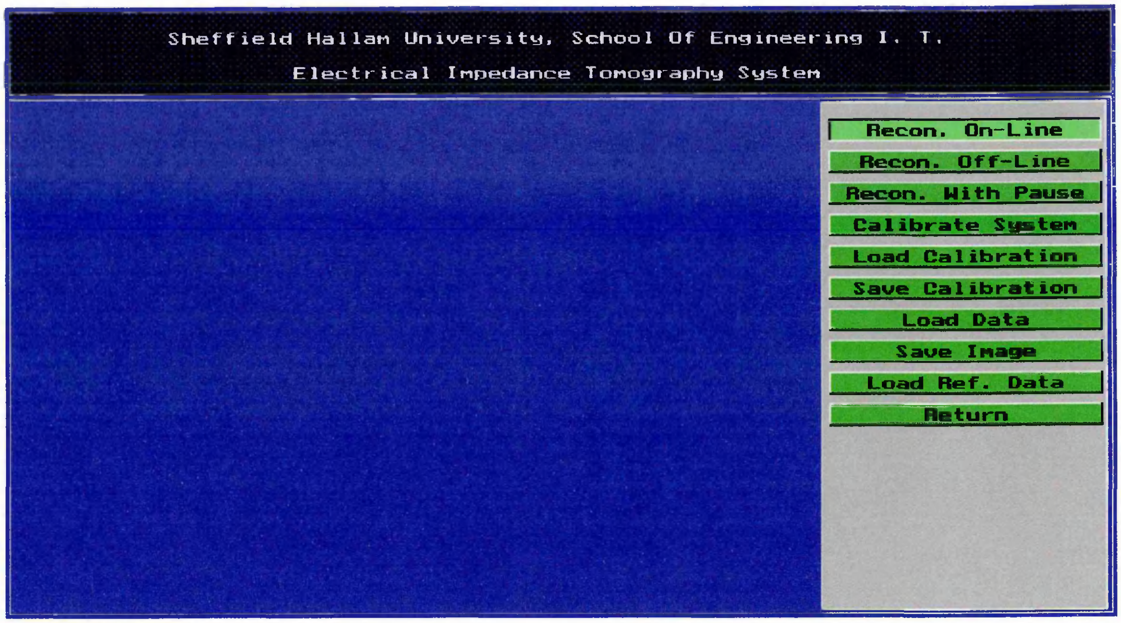


Figure 3.3.4 Menu Displayed
for Reconstruction R or Reconstruction S Main Menu Options

Recon. On-Line: Records a set of voltage readings using the EIT hardware and reconstructs an image using the reconstruction algorithm described in Chapter 3.4. The image reconstructed has either a circular or square boundary depending on which menu was selected (Reconstruction R = circular boundary and Reconstruction S = square boundary).

Recon. Off-Line: Reconstructs an image from data loaded from disk (see Figure 3.3.5).

Recon. With Pause: Records a set of voltage readings using the EIT hardware and reconstructs an image. Then pauses until the operator press a key to continue. The system then records another set of voltage readings and again reconstructs an image. This continues until the "ESC" key is pressed.

Calibrate System: This menu option allows the operator to experimentally determine the sensitivity coefficients used in the reconstruction algorithm (see Figure 3.3.6).

Load Calibration: Loads sensitivity coefficients from disk.

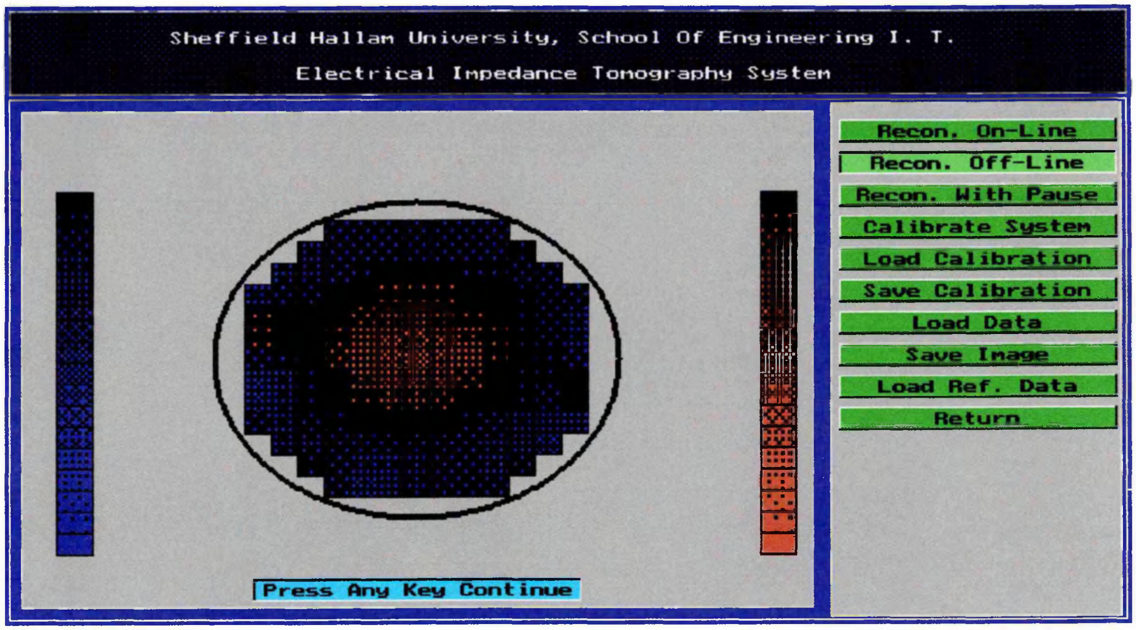
Save Calibration: Save sensitivity coefficients to disk.

Load Data: Loads a set of voltage readings from disk. This is used when reconstructing an image off-line.

Save Image: Saves the last reconstructed image to disk. The file name is Rimage.dat for an image with a circular boundary and Simage.dat for the square boundary.

Load Ref. Data: Loads a set of voltage readings that is used as the reference set (usually a set of voltage readings from a phantom with homogenous conductivity.

Return: Return the system to the main menu screen.

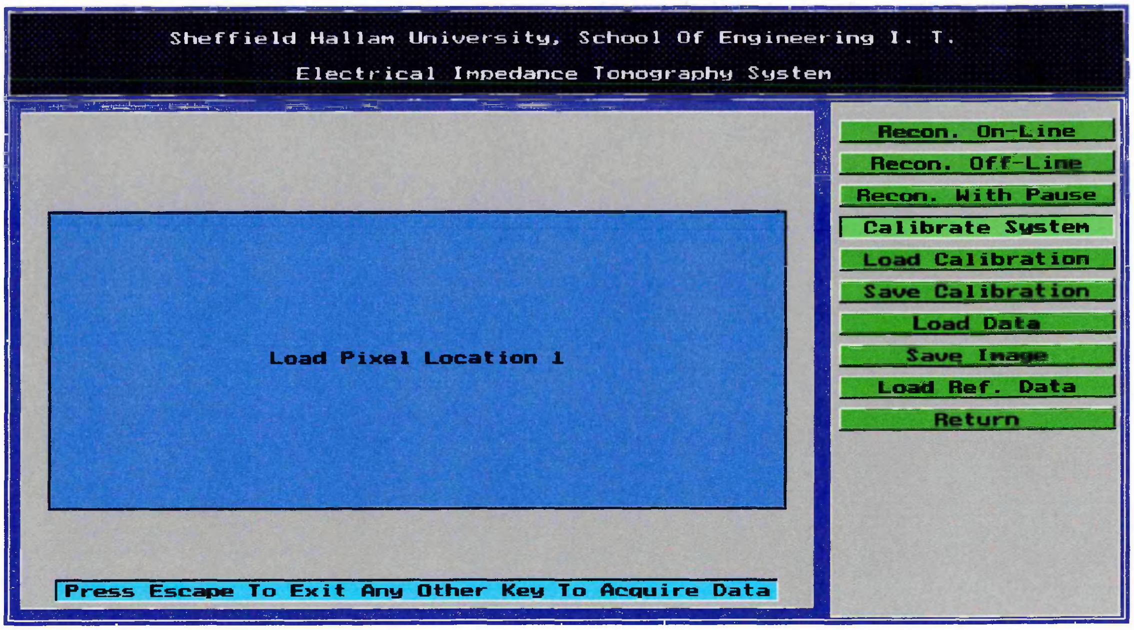


Sheffield Hallam University, School Of Engineering I. T.

Electrical Impedance Tomography System

Press Any Key Continue

Figure 3.3.5 Reconstructed Image
from Cicular Phantom using Adjacent Electrode Drive Configuration.



Sheffield Hallam University, School Of Engineering I. T.

Electrical Impedance Tomography System

Load Pixel Location 1

Press Escape To Exit Any Other Key To Acquire Data

Figure 3.3.6 Calibrate System Window

3.4 Reconstruction Algorithm

A reconstruction algorithm based on sensitivity coefficients (Kotre C.J., 1989) was implemented for the images presented in this thesis. However the sensitivity coefficients are not determined numerically (Kotre C.J., 1993) but are determined experimentally (Yu Z.Z., Peyton A.T., Beck M.S. Conway W.F. and Xu L.A., 1993). Determining the sensitivity coefficients experimentally allows for changes in object geometry and changes in the excitation and measurement strategies used.

The area to be imaged is divided into a number of squares, each square representing one pixel in the reconstructed image. To initially test the reconstruction algorithm (and further test the EIT hardware) a circular tank of 100mm diameter and 50mm height was used as the interrogated area. A set of sixteen equally spaced stainless steel electrodes are positioned on the periphery of the tank as shown in Figure 3.4.1 to a height of 25mm. The tank is filled with a saline solution that has a conductivity of 10 mS cm^{-1} to a depth of 25mm. The tank was divided into 145 pixels as shown in Figure 3.4.2. The adjacent electrode excitation and measurement strategy was used giving 16 projections (M) with 13 differential voltage readings (N) per projection.

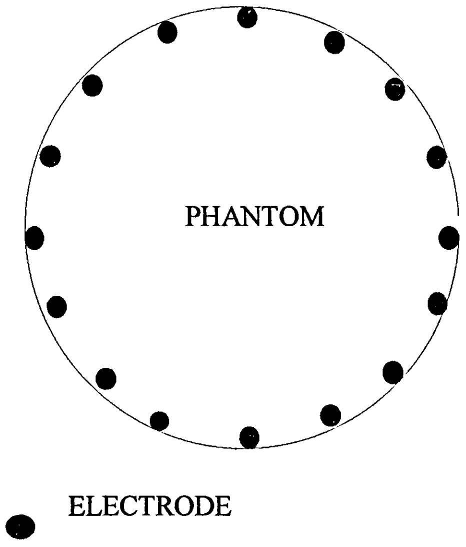


Figure 3.4.1 Circular Tank Electrode Configuration

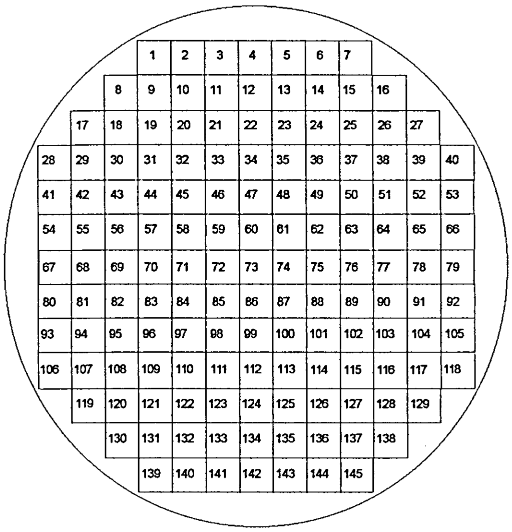


Figure 3.4.2 Circular Tank Pixel Locations

A set of boundary voltages were recorded with only the saline solution present. This set of boundary voltages were taken as the homogenous data set ($V_{H(m.n)}$). An insulating (or conducting) rod was inserted into each pixel location in turn and a set of boundary voltages for that pixel location were recorded ($V_{p\,(m,n)}$). The sensitivity coefficients were then defined from these boundary voltages as:

$$S_{(p,m,n)} = V_{p(m,n)} - V_{H(m,n)} \qquad (3.4.1)$$

where p is the pixel number, m is the projection number (there are a total of M projection) and n refers to the nth element of the measured voltage data set for a given projection (there are N differential voltage readings for each projection). The sensitivity matrix ($S_{(p,m,n)}$) therefore describes the system's response to an input at each pixel location. The reconstruction algorithm is defined as:

$$P_{(p)} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} S_{(p,m,n)} \times ((V_{c(m,n)} - V_{H(m,n)})/V_{H(m,n)})}{\sum_{m=1}^{M}\sum_{n=1}^{N} S_{(p,m,n)}} \qquad (3.4.2)$$

where P is the pixel value and $V_{c(m,n)}$ is the boundary voltages for the conductivity distribution being imaged. A square perspex rod (anomaly) 1.96cm^2 is placed in the tank and centred at pixel locations 73, 75 and 77; the images produced by the reconstruction algorithm are shown in Figures 3.4.3 to 3.4.5. The images reported in the thesis have been generated by importing the raw pixel values for the reconstruction algorithm into MATLAB. MATLAB is used to normalise the images by mapping the minimum pixel value to 0 and the maximum pixel value to 100. These normalised images are then displayed using MATLAB. The algorithm use to perform the normalisation is given in equation 3.4.3

$$Pn_{(m,n)} = \frac{P(p)_{(m,n)} - P\min}{(P(p)_{(m,n)} - P\min)\max} \qquad (3.4.3)$$

where: $Pn_{(m,n)}$ is the matrix of normalised pixel values

$P(p)_{(m,n)}$ is the matrix of original pixel value

*P*min is the minimum pixel in $P(p)_{(m,n)}$

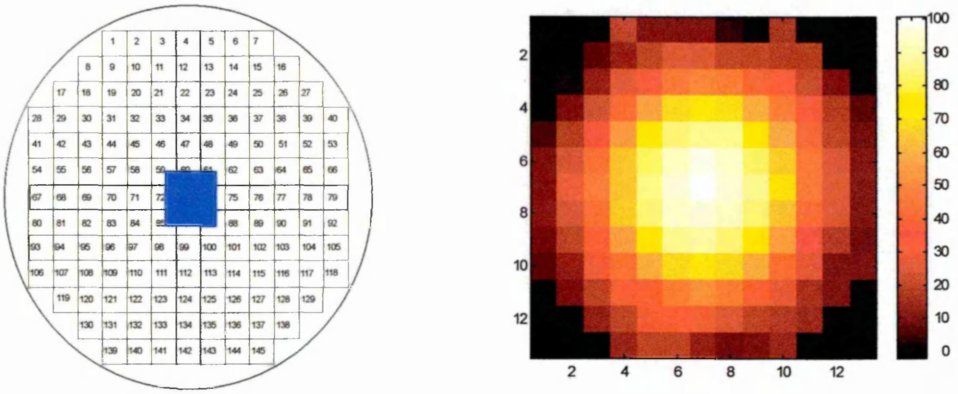


Figure 3.4.3 Reconstructed Image with Anomaly Centred at Pixel Location 73

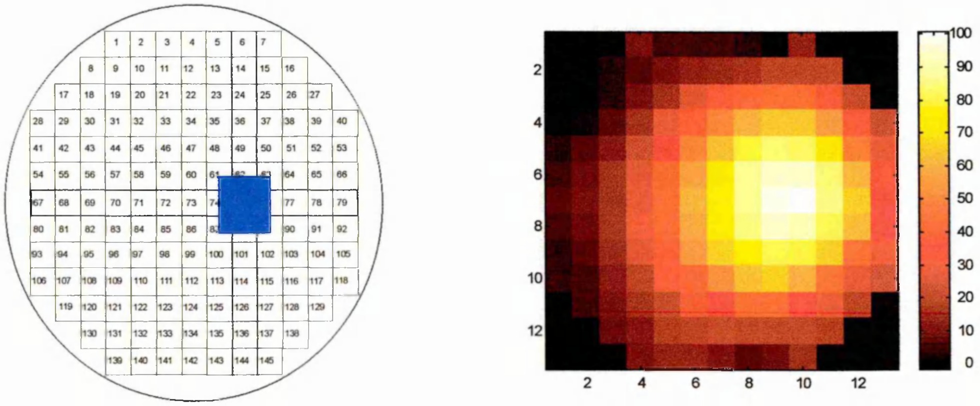


Figure 3.4.4 Reconstructed Image with Anomaly Centred at Pixel Location 75

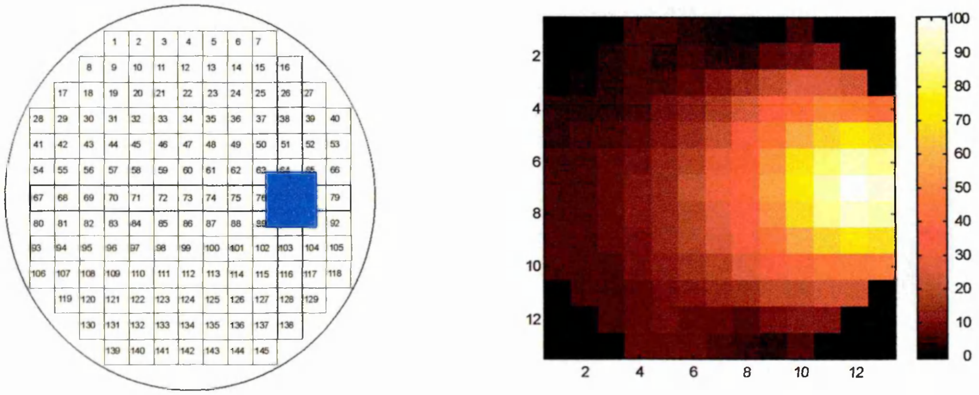


Figure 3.4.5 Reconstructed Image with Anomaly Centred at Pixel Location 77

From Figures 3.4.3 to 3.4.5, it can be seen that the reconstruction algorithm is working correctly. There is a significant amount of blurring in the images and the reconstructed position of anomalies towards the edge of the tank do not fully correlate with the actual anomalies position. Both the above effects are known limitations of the reconstruction algorithm (Kotre C.J., 1993)

The above algorithm was tested on a square tank in order to verify that the algorithm will work with different object geometries. The tank's dimensions were 150mm by 150mm with a height of 30mm. A set of sixteen equally spaced stainless steel electrodes were placed at the periphery of the tank as shown in Figure 3.4.6 to a height of 25mm. The tank was filled with a saline solution with a conductivity of 10 mS cm^{-1} to a depth of 25mm. The tank was divided into 225 pixels as detailed in Figure3.4.7 and the adjacent electrode drive configuration was used. The same procedure used to obtain the sensitivity coefficients for the circular tank was also used to obtain the sensitivity coefficients for the square tank.

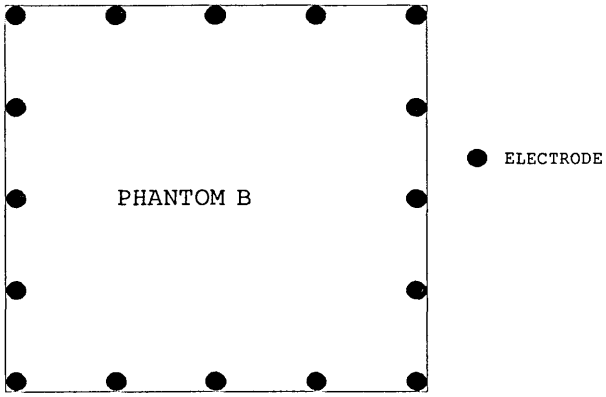
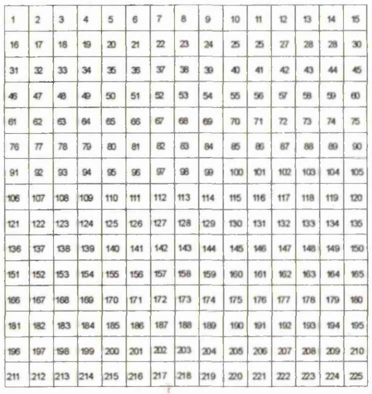


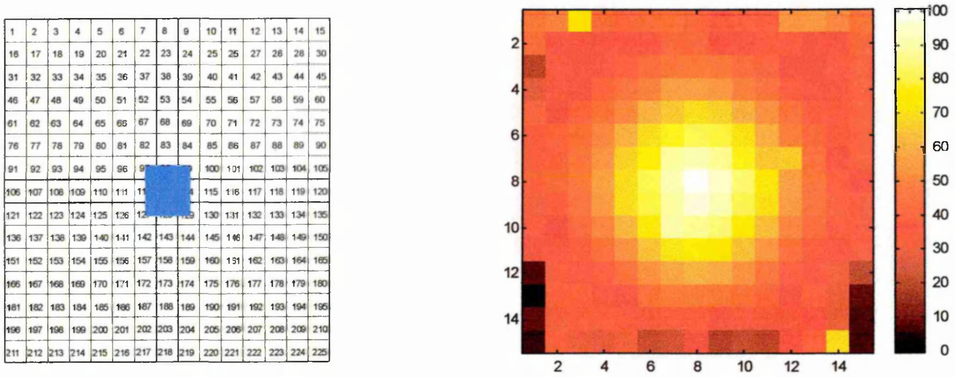
Figure 3.4.6 Square Tank Electrode Configuration



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	25	27	28	28	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165
166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225

Figure 3.4.7 Square Tank Pixel Locations and Number

A square 1.96cm^2 perspex rod was used as an insulating anomaly and is placed in the tank at pixel locations 113, 115, 117 and 119. The images produced by the reconstruction algorithm are shown in Figures 3.4.8 to 3.4.11. All the images have been scaled between 0 and 100% of their minimum and maximum pixel value (as per equation 3.4.2).



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	25	27	28	28	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	9		9	100	101	102	103	104	105
106	107	108	109	110	111	11		4	115	116	117	118	119	120
121	122	123	124	125	126	12		29	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165
166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225

Figure 3.4.8 Reconstructed Image with Anomaly Centred at Pixel Location 113

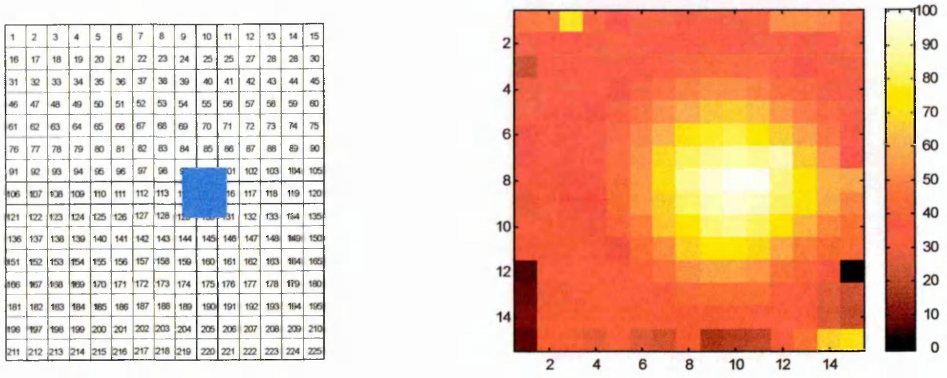


Figure 3.4.9 Reconstructed Image with Anomaly Centred at Pixel Location 115

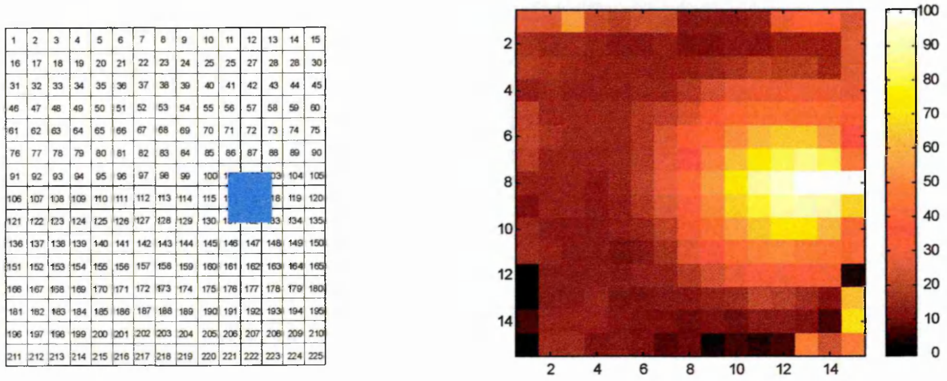


Figure 3.4.10 Reconstructed Image with Anomaly Centred at Pixel Location 117

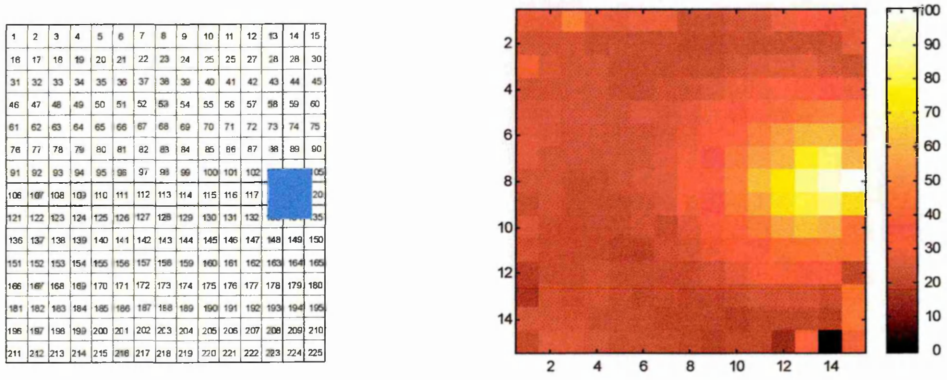


Figure 3.4.11 Reconstructed Image with Anomaly Centred at Pixel Location 119

From Figure 3.4.8 to 3.4.11 it is shown that the reconstruction algorithm does image different object geometries and is therefore suitable for investigating and comparing different boundary geometry's and electrode drive configurations.

A distributed electrode drive configuration was investigated which was developed from the work of Lyon G.M. and Oakley J.P., 1992. They showed that placing electrodes on the central stirrers of a mixing vessel improved the sensitivity in the central area. Therefore, the hypotheses for using the distributed electrode drive configuration was that placing electrodes throughout the area would further increase the sensitivity. The electrode arrangement is shown in Figure 3.4.12 and has significant improvements over the standard periphery arrangement of electrodes as detailed in Chapter 5. The reconstruction algorithm outlined above was used to reconstruct images from a square tank using the distributed electrode configuration shown in Figure 3.4.12.

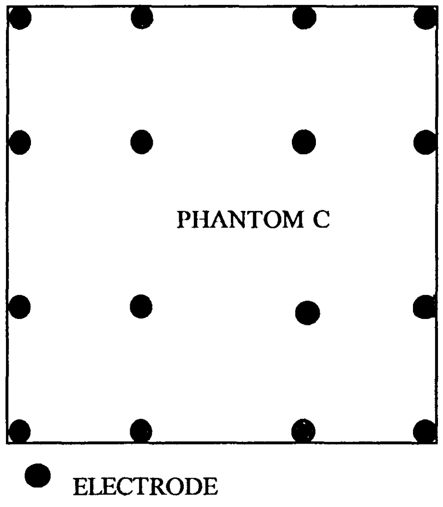


Figure 3.4.12 Distributed Electrode Configuration

The tank was 150mm by 150mm with a height of 30mm. Stainless steel electrodes were used for each electrode shown in Figure 3.4.12. The tank was filled with a saline solution with a conductivity of 10 mS cm^{-1} to a depth of 25mm. The tank is divided into 225 pixels as detailed in Figure 3.4.7.

A square perspex rod (anomaly) $1.96cm^2$ was placed in the tank and centred at pixel locations 113, 115, 117 and 119 the images produced by the reconstruction algorithm are shown in Figures 3.4.13 to 3.4.16. All the images have been scaled between 0 and 100% of their minimum and maximum pixel value (as per equation 3.4.3).

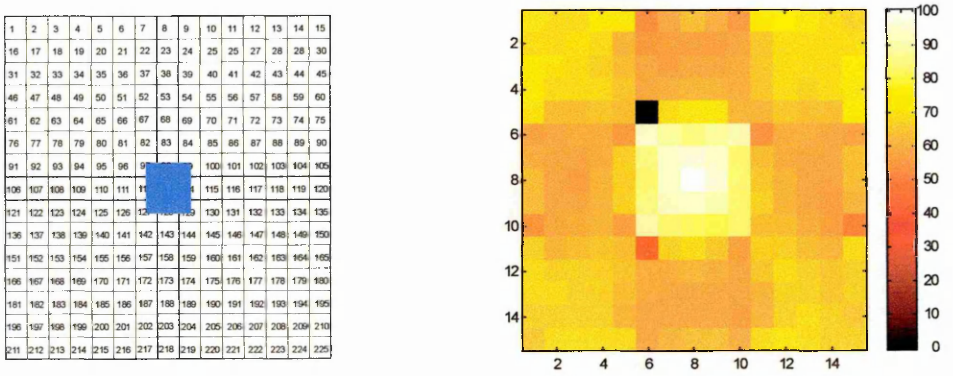


Figure 3.4.13 Reconstructed Image with Anomaly Centred at Pixel Location 113

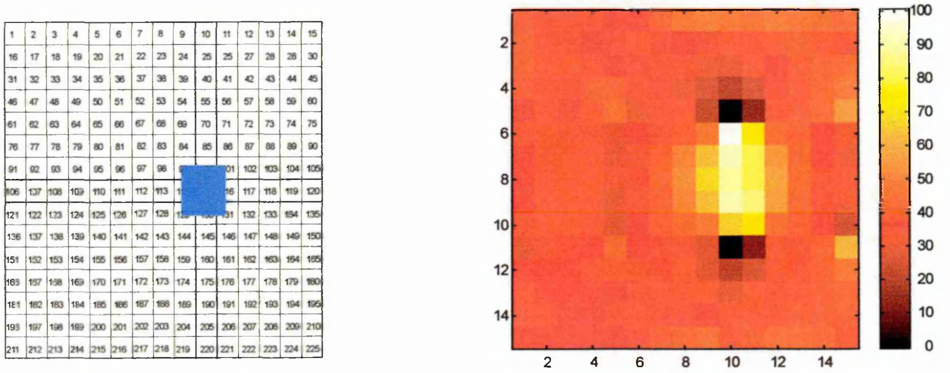


Figure 3.4.14 Reconstructed Image with Anomaly Centred at Pixel Location 115

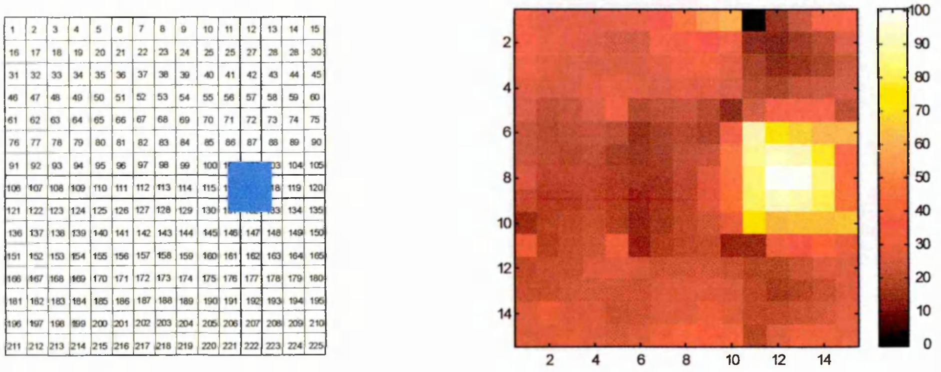


Figure 3.4.15 Reconstructed Image with Anomaly Centred at Pixel Location 117

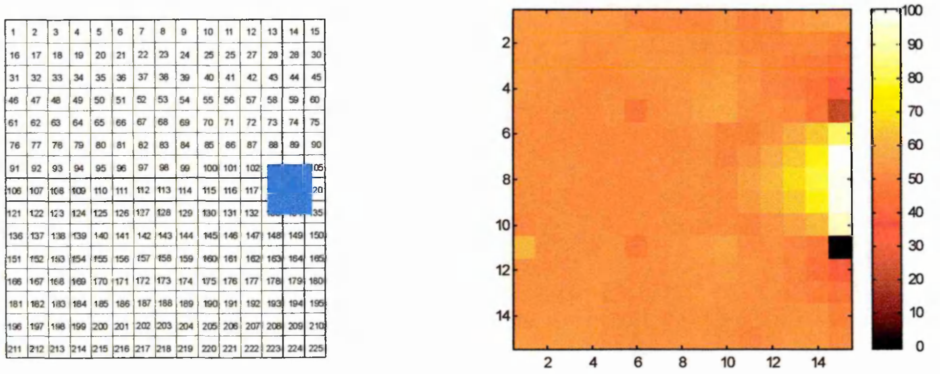


Figure 3.4.16 Reconstructed Image with Anomaly Centred at Pixel Location 119

From Figure 3.4.13 to 3.4.16, it can be seen that the reconstruction algorithm reconstructs images when a different electrode configuration is used. However, the images from the distributed electrode configuration have a number of artefacts as well as the image of the anomaly present. These image artefacts are due to the higher sensitivity of the distributed electrode configuration as shown in Chapter 5. The increase in sensitivity of the distributed arrangement leads to an increase in sensitivity of the reconstruction algorithm to the soft field effect. The increase in sensitivity of the distributed electrode arrangement can be seen in Figures 3.4.17 and 3.4.18. These show the sensitivity profiles of the square tank with the peripheral electrodes and the distributed electrodes respectively (the images shown are scaled between 0 and 100 per of their minimum and maximum values.

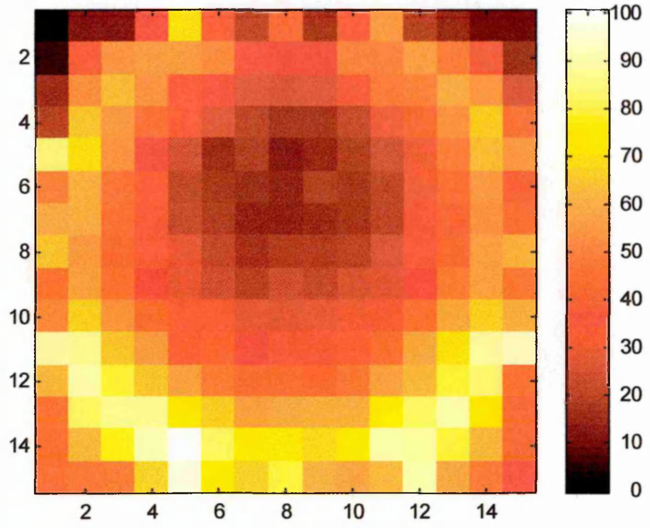


Figure 3.4.17 Sensitivity Profile of Square Tank Peripheral Electrode Configuration.

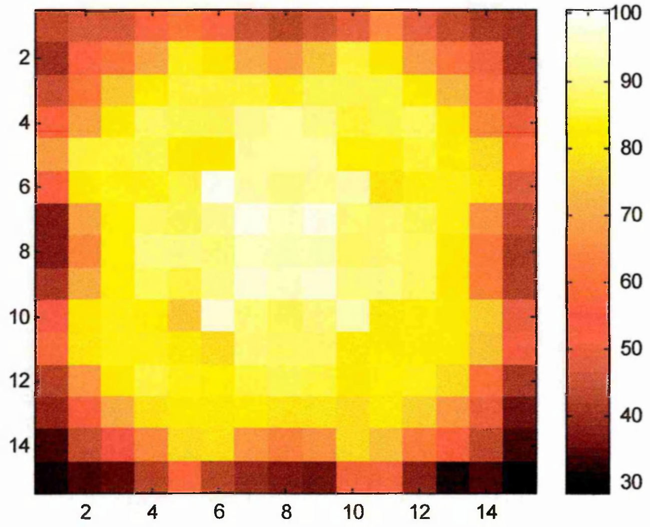


Figure 3.4.18 Sensitivity Profile of Square Tank Distributed Electrode Configuration.

From the sensitivity profiles of Figures 3.4.17 and 3.4.18, it can be seen that the square tank with the distributed electrode configuration is more sensitive at the centre of the tank than at the edge. Whereas the square tank with the peripheral electrode configuration is more sensitive at the edge of the tank than at the centre. In addition, the maximum sensitivity value for the peripheral electrode configuration is 4533 whereas the maximum sensitivity for the distributed electrode arrangement is 9735. This is an increase of 114.76%. A more detailed comparison of the above electrode configurations is given in Chapter 5.

4. Distributed pressure measurement

4.1 Introduction

This chapter presents a review of distributed pressure measurement and also presented is the initial sensor design proposed. The theoretical relationship between changes in the sensor's conductivity distribution and applied pressure is presented in Section 4.3.2. and Section 4.3.3 presents the investigation into suitable materials for the sensor.

4.2 Review of distributed pressure measurement

The review of distributed pressure measurement is divided into two Sections. Section 4.2.1 reviews distributed pressure measurement as developed for interface pressure measurement in medical applications and Section 4.2.2 reviews tactile sensor arrays that have been use to image force/pressure distribution in other applications.

4.2.1 Interface pressure measurement

The measurement of distributed pressure has an important role in medical application for monitoring the pressure exerted at the interface between a body and its supporting medium Knight R.A., 1991. The main application areas being in the prevention of pressure sores, by being able to constantly monitor those at the highest risk (Knight R.A., 1991), the design of seat cushions for wheel chairs (Mooney V., Einbund M.J., Rogers J.E. and Stauffer E.S., 1971) and the design of foot wear to reduce the occurrence of ulceration's commonly suffered by diabetics (Warren-Forward M.J, Goodall R.M., and Pratt D.J., 1992). Many Interface pressure measurement techniques have developed and a review of these between 1965 and 1989 is presented by Knight R.A, 1991 the main findings of which follow.

The early work of Lindan O., Greenway R.M., and Piazza J.M., 1965 studied the pressure distribution of seated and lying subjects and the pressure distribution measurements made were used in studies into pressure sore formation. The system of Lindan et al, 1965 consisted of a approximately 1000 padded headed nails (depending on the resolution that was required) each nail was threaded through a calibrated spring and placed through drilled holes in a plywood sheet. These padded headed nails provided a bed, on which the subject either laid or sat. The pressure exerted by the subject caused the springs to compress which therefore caused the nails to protrude through the bottom of the plywood sheet. The protrusion of each nail below the plywood sheet was measured and these measurements used to determine the pressure distribution exerted on the bed. Although the system of Lindan et al, 1965 did provided pressure distribution information, the system had a number of limitations. The time required to collect all the measurements was between 1 and 2 hours duration in which time the subject had to remain still. This made it inappropriate for constant monitoring of patients and as the system was the support surface it could not be used to measure the pressure distribution between the subject and a real support surface.

A simple technical method for obtaining distributed pressure profiles for a comparative studies of seating design for airlines was researched by Swearingen J.J., Wheelright D.C. and Garner J.D., 1962. The distributed pressure profiles were obtained by placing an absorbent paper over inked corduroy cloth. A similar study was also performed by FrisinaW. And Lehneis H.R., 1970.

From Knight's review, it is clear that the measuring technique based on a pneumatic cell pressure sensor has been used by a number of researchers for comparative studies of patient support surfaces. The pneumatic cell sensor consists of a rubber bladder with electrical contacts on opposite faces the bladder is inflated until the electrical contacts are broken. A sphygmomanometer was used to inflate the bladder and the reading on the sphygmomanometer when the electrical contacts were broken taken as the highest pressure exerted on the bladder. The researchers made the point that the maximum pressures measured did not give a true representation of the maximum pressure locally exerted by the patient on the support medium. This was reported as being due the sensor itself conforming and redistributing the pressure.

Garber S.L., Krouskop T.A. and Carter R.E., 1978 developed a system that produced pressure distribution profiles using 144 pneumatic sensor in a 12 by 12 array. The array of sensor was connected to a 12 by 12 array of LED's with each LED representing one of the individual pneumatic sensors in the sensor array. The pressure distribution profile was obtained by first inflating all the sensors in the array until all the LED's were off (i.e. the pneumatic sensors inflated until the electrical contact is broken). The pressure in the sensor is then reduced in fixed decrements and the array of leds monitored. As with the other pneumatic sensors this type of array should not be assumed precise but is useful for comparative studies of support media.

A barographic measurement technique was used in a study by Patil K.M. and Babu T.S., 1986 to map the pressure distribution under the leprotic foot. The pressure distribution profiles were obtained by illuminating a thick glass sheet with a fluorescent light from it edges. A thin, opaque, plastic sheet was placed on the top surface of the glass on which the patient stood. The higher the pressure exerted by the patient on the plastic sheet the greater the contact pressure between the plastic sheet and the glass. The increase in contact pressure between the plastic sheet and the glass caused breakdown of the total internal reflection within the glass. The image formed by the breakdown of the total internal reflection was viewed from the bottom surface using a microdesitometer. As the light intensity was related to the applied pressure, the scanned image was used to determine the pressure distribution. This system did provided information, which led to improvement in footwear design however, the system is not practical for constant patient monitoring.

Also from the review by Knight it is clear that a larger number of the techniques developed for foot pressure measurement have been based on arrays of individual pressure sensors each sensor giving a pressure reading. The readings from each of the individual sensors being used to obtain a pressure distribution profile.

Soames R.W, 1985 used an array of strain gauges to measure peak pressures and temporal parameters of gait. Whereas Tappin J.W., Pollard J. and Beckett E.J. 1980 used semiconductor devices containing, magneto-resistors to analyse shear forces acting upon the foot during walking. The magneto-resistors generate output voltages that are proportional to the horizontal shear forces acting upon them. Two main reasons why these techniques have not been more widely applied are the lack of information regarding the effects of the support interface. Also as the size of the area increases or a higher resolution is need so the number off discrete sensors has to be increased and hence the number of connections to them.

Hennig E.M., Cavangh P.R., Albert H.T. and Macmillan N.H., 1982 used an array of 500 piezoelectric transducers. A piezoelectric device generates a surface charge proportional to the stress applied to its surface. Although the system did produce impressive results when monitoring the footfall during walking, a disadvantage of this and similar systems is that not only did the device exhibit a piezoelectric effect but also a pyroelectric effect. A pyroelectric device generates a surface charge due to changes in temperature. As a patient in contact with the sensor would cause changes in temperature, its use for constant patient monitoring is limited. Another limiting factor is the fact that a piezoelectric device can only measure dynamic changes in pressure and can not measure static pressures.

Maalej N., Webster J.G., Tompkins W.J. and Wertsch, J.J., 1989, Webster J.G., 1989, and Pax Jr. R.A., Webster J.G. and Radwin R.G., 1989 describe how arrays of discrete pressure sensors available from Interlink Electronics could be used for measuring interface pressure. The discrete sensors consisted of a conductive polymer film between two layers of Mylar film. The bottom layer of Mylar film had a conductive pattern printed on to the surface and at a microscopic level, the conductive polymer has an uneven surface. As the pressure applied to the sensor is increased the area of the conductive polymer in contact with the conductive pattern on the Mylar film is also increased and so the resistance of the sensor decreases. The sensor arrays made from the above sensors were thin and flexible put had the disadvantages that they exhibited hysteresis and their performance degraded over time.

More recently Warren-Forward M. J., Goodall R. M. and Pratt D. J., 1992 presented a system to improve the design of footwear for diabetics. The system consisted of four coils embedded within a rubber insole. One coil is used for excitation and is embedded in one surface while the three remaining coils used for sensing are embedded in the opposite surface. The excitation coil is driven by a 40mA 20kHz signal and the voltages induced in the sensing coils due to flux linkage measured. The measured voltages are processed to determine the displacement of the excitation coil in relation to the sensing coil with accuracy's of 5% being achievable. Warren-Forward M. J., Goodall R. M. and Pratt D. J., 1992 intend to further the work by building eight of the above sensors into an insole and also to process the voltages further in order determine applied forces. The above system appears ideal for measuring triaxial forces between a patients foot and their footwear. For larger surface areas i.e. between a patient and a bed, the number of individual sensor required and their associated connections may make a system based on discrete sensors impractical. The sensor base on EIT reported in this thesis could overcome this problem.

4.2.2 Tactile sensor array

The main area of application for tactile sensors arrays is in the field of robotic gripper control. The tactile array gives a robotic gripper an image of the object it is holding. The image obtained by the array can be analysed during a handling or assembly process for the purpose of shape recognition, orientation, slip detection and the monitoring of force patterns. (Robertson B. E. and Walkden A. J., 1985)

A tactile sensor array is made up of a matrix of separate tactile sensors (discrete sensor devices). The data collected from each of the tactile sensors in the array can be combined to describe the characteristics of the object in contact with the array. Typical characteristics that can be described are: (Groover M.P., Weiss M., Nagel R.N. and Odrey N.G., 1986)

1. The presence of an object.
2. The objects contact area, shape and location.
3. The object's pressure (force) and pressure distribution.

Tactile sensors can be divided into two classes: touch sensors and force sensors. Touch sensors provide a binary signal output indicating the presence or absence of an object. The touch sensor can give characteristic 1 and 2 from the above list. Force sensors indicate not only the presence or absence of the object but also the magnitude and direction of the force the object is exerting on the sensor (Groover M.P., Weiss M., Nagel R.N. and Odrey N.G., 1986). A number of tactile sensors have been developed where an applied force alters the optical, resistive, capacitive, inductive, piezoelectric or acoustic characteristics of the tactile (Kolesar Jr E., Reston R.R., Ford D.G. and Fitch Jr C., 1992).

Tekada S., 1974 and Page C. J., Pugh A., and Heginbotham W.B., 1976 developed some of the early types of sensor which were simple mechanical displacement force sensors. The systems had limited resolution and the construction meant they were not suitable for integration into robotic grippers. A tactile sensor base on magnetoelastic was developed by Checinski S. S. and Agrawal A. K., 1984 which measured forces in the 1-100gm range, had low hysteresis and an overload capability of 100gm.

Sommer G., Leibbrandt S. L. and Stojanoff C. G., 1994 describe a tactile sensor that utilises the deformation of a holographic membrane. The forces applied to the membrane cause it to deform and these deformations changed the diffraction efficiency of the holographic membrane. The changes in the diffraction efficiency result in an intensity coded signal that is measured precisely by a CCD. The sensitivity of the device is 0.01N with a range of 10N.

Many researchers in recent years have developed tactile sensors using piezoelectric materials (Regtien P .P. L., 1989). One resent system is that of Kolesar Jr E., Reston R.R., Ford D.G. and Fitch Jr C., 1992. The sensor is a silicon-integrated circuit coupled to a 0.025mm thick piezoelectric polyvinylidene fluoride (PVDF) film. The integrated circuit consists of an array of 25 sensing electrodes arranged in a 5 x 5 matrix onto which the PVDF film was attached. Each electrode is 0.6mm x 0.6mm in area and they are separated by 0.6mm. The sensor is reported to have a linear response over loads spanning 0 to 60 gmf and a response bandwidth of 33Hz. Two concerns expressed by Kolesar Jr E.   et al were the pyroelectric effect experienced by the piezoelectric film and the ability to resolve the direction of the forces applied. They suggested that the piezoelectric effect could be compensated for by either coating the PVDF film with a compliant film with a low thermal conductivity value or include a microthermistor in the array. However even with the above concerns from their experiments they suggested that resolving 1mm features was achievable.

Kolesar Jr E. S. and Dyson C. S., 1995 further developed the above system by increasing the number of electrodes to 64 configured as an 8 x 8 matrix. They also reduced the size of the electrode to 0.4 x 0.4mm and reduced the spacing between each electrode to 0.3mm and the PVDF film thickness was increased to 0.04mm. The response of their sensor was linear over 0.8-135gmf with a response bandwidth of 25Hz. The sensor is shown to recognise the silhouette of various shapes and a resolution in the order of 0.7mm is reported.

A capacitive sensor based on differential capacitive measurement is presented by Chu Z, Sarro P.M. and Middelhoek S., 1996. The sensor is fabricated using IC processing and micromachining technology. The experimental devices developed have a sensitivity of 0.13pF/g to normal forces and 0.32pF/g to shear forces with a force range of 0 to 1g. The devices have a response bandwidth of 162Hz and a spatial resolution of 2.2mm.

From the published literature it is clear that in recent years the majority of the research into tactile sensing has been in reducing the size of the individual sensing devices and forming them into arrays. This has been achieved by either IC fabrication processes or micromachining fabrication. The main aim being to produce a sensor with a resolution similar to the human finger (i.e. less than 1mm). The systems based on piezoelectric tactiles and capacitive arrays achieved these level of resolution.

4.3 Initial Sensor Design

4.3.1 Proposed system

The two main components of the proposed system are the distributed pressure sensor and electrical impedance tomography system. The operation of such a system is thus: a load applied to the distributed pressure sensor causes changes in resistance in the area to which the load is applied and these changes in resistance are mapped by the EIT system. One design proposed for the distributed pressure sensor consists of a flexible conductive sheet with the electrodes of the EIT system connected to the boundary of the sheet.

4.3.2 Theoretical relationship between conductivity changes in a planar sensor element and the applied pressure.

This analysis is similar to that described by Knight R. A., 1991. Consider a conductive flexible sheet exhibiting homogenous and isotropic conductivity σ with modulus of compression E (Young's modulus of elasticity) and uniform thickness z as shown in Figure 4.3.2.1, where

$$\sigma = (GL)/A \tag{4.3.2.1}$$

$$E = S/\varepsilon \tag{4.3.2.2}$$

where *G* is the conductance of the material, *A* is the cross-sectional area of the sheet , *L* is its length, *S* is the stress and ε is the strain.

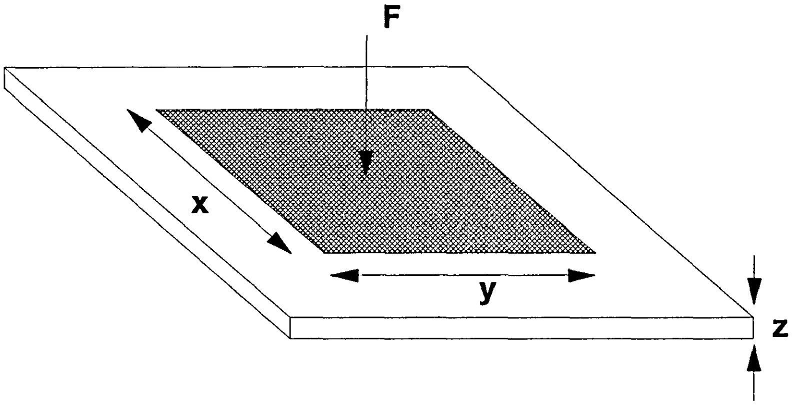


Figure 4.3.2.1 Conductive sheet with area (x, y) subject to an applied force F

Now consider an area of the sheet subjected to an applied force F, with the area to which the force is applied having dimensions x and y as shown in Figure 4.3.2.1. It is assumed that the edges of the sheet are constrained and only dimensional changes in the thickness of the sheet (z) are affected by the applied force. The area to which the force is applied is subjected to a stress S and therefore a strain ε which are related to the applied force by

$$S = F/(xy) \tag{4.3.2.3}$$

and

$$\varepsilon = \delta z/z \tag{4.3.2.4}$$

where δz is the change in thickness z due to the applied force.

The modulus of compression of the sheet in terms of its dimensions and the force to which it is subjected is determined by substituting equations (4.3.2.3) and (4.3.2.4) into equation (4.3.2.2)

$$E = (Fz)/(xy\delta z) \tag{4.3.2.5}$$

From equation 4.3.2.1

$$\sigma = (Gy)/(xz) \tag{4.3.2.6}$$

which gives

$$G = (\sigma xz)/y \tag{4.3.2.7}$$

This represents the conductance of the area under consideration before the force has been applied. Following the application of the force

$$G' = [\sigma x(z - \delta z)]/y \tag{4.3.2.8}$$

where *G'* is the conductance of the area when subjected to the force *F*. The change in conductance for the area subjected to the force can now be determined by subtracting equation (4.3.2.8) from (4.3.2.7)

$$\begin{aligned} \delta G &= G - G' \\ &= (\sigma x \delta z)/y \end{aligned} \tag{4.3.2.9}$$

By rearranging equation 4.3.2.5 to find δz and substituting into equation 4.3.2.9 along with equation 4.3.2.6 for σ, the change in conductance δG of the area of the applied force can be expressed in terms of the magnitude of applied force and the dimensions of the area subjected to this force

$$\begin{aligned} \delta G &= (\{(Gy)/(xz)\}xFz)/(xy^2 E) \\ &= (GF)/(xyE) \end{aligned} \tag{4.3.2.10}$$

Assuming that the relationship between pressure and applied force is a linear relationship, then

$$P = F/A \tag{4.3.2.11}$$

where A is the area of application

Rearranging equation (4.3.2.11) to find F and substituting into equation (4.3.2.10) gives a direct theoretical relationship between the applied pressure and the change in conductance,

$$\delta G = (GP)/E \tag{4.3.2.12}$$

Thus, the small change in conductivity is directly proportional to the applied pressure and inversely proportional to the modulus of compression. The above relationship assumes that the material used has no piezo-resistive effect and that its resistance is stable with temperature.

4.3.3 Material studies

This section describes the characterisation experiments performed on various conducting polymer samples. Table 4.3.3.1 shows the sources of the various samples investigated. Three characterisation experiments were performed on the samples in order to evaluate their suitability as a sensor material for the proposed system. The three tests are voltage/current characterisation, creep characterisation and load characterisation.

Sample No	Source or Manufacture
1	The Gates Rubber Company Limited
2	Maclellan Rubber Limited
3	Xelflex Precision Moulders
4	The Layland & Birmingham Rubber Co. Limited
5	Conductive Foam, RS Components Limited
6 7	Dr. S Kosina, Dept. Chemical Physics, Faculty of Chemical Technology, Slovak Technical University (Polypyrrole samples)

Table 4.3.3.1 Conductive Polymer Samples Source Of Manufacture

It is well known that the contact resistance between the electrodes attached to the polymer and the polymer surface is a problem (Norman R. H., 1970). To reduce this error when characterising the samples the four-point or probe method is used (Mayer J.W and Lau S. S., 1910) as shown in Figure 4.3.3.1. To further reduce the contact resistance effects on the measurements, brass electrodes are used as these are reported to give the lowest contact resistance (Norman R. H., 1970), at least with conducting polymers having resistances up to 10^5 Ωcm.

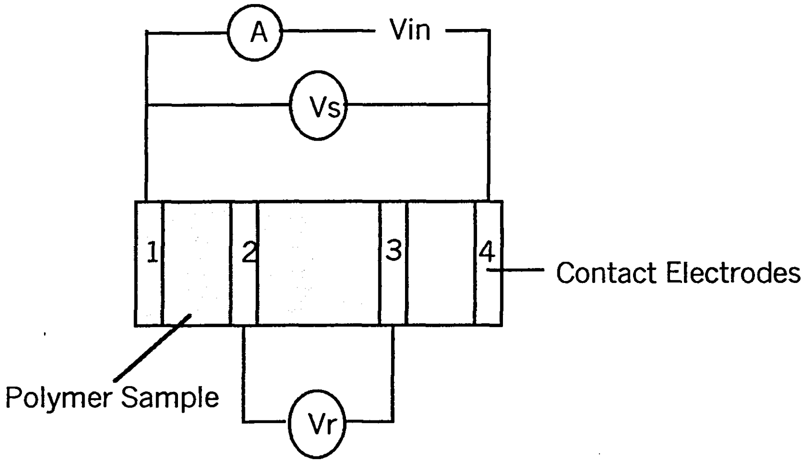


Figure 4.3.3.1 Four-probe method for characterisation of polymer samples

The measurement procedure shown in Figure 4.3.3.1 was used for samples 1 to 5. For sample 6 and 7 due to their size (1cm by 1cm) they had to be mounted onto a mounting plate, this fitted onto the standard test rig, used. This mounting plate is shown in Figure 4.3.3.2.

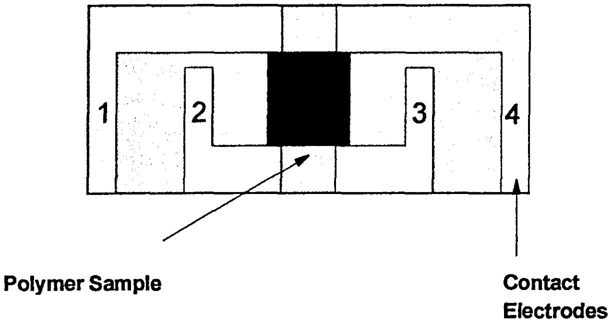


Figure 4.3.3.2 Mounting plate for sample 6 and 7

4.3.3.1 Voltage/Current Characteristics

This experiment was designed to determine the V/I characteristics of the samples. The voltage applied to electrodes 1 and 4 was varied and the resulting voltage (Vr) between electrodes 2 and 3 measured. The current through the samples was then recorded and Vr was plotted against current to give the V/I characteristics for each sample, as shown in Figures 4.3.1.1 to 4.3.1.7. The voltage between electrodes 2 and 3 was measured using a Thurbly 1905a multimeter and current through the sample was measured using a Hameg 8011-3 multimeter. The maximum voltage used with each sample was determined by increasing the applied voltage (Vs) in discrete steps and then allowing the supply current reading to settle for 10 seconds. If after 10 seconds, the current reading had not settled the previous value for Vs was taken as the maximum supply voltage to be applied to the sample. The randomly and continually fluctuating supply current readings above the maximum supply voltage used for each sample are most likely caused by the heating effects within the sample.

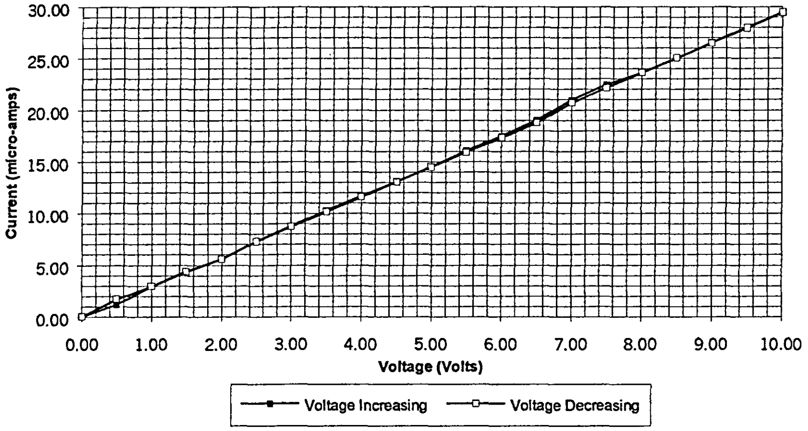


Figure 4.3.3.1.1 Sample 1; V/I characteristics

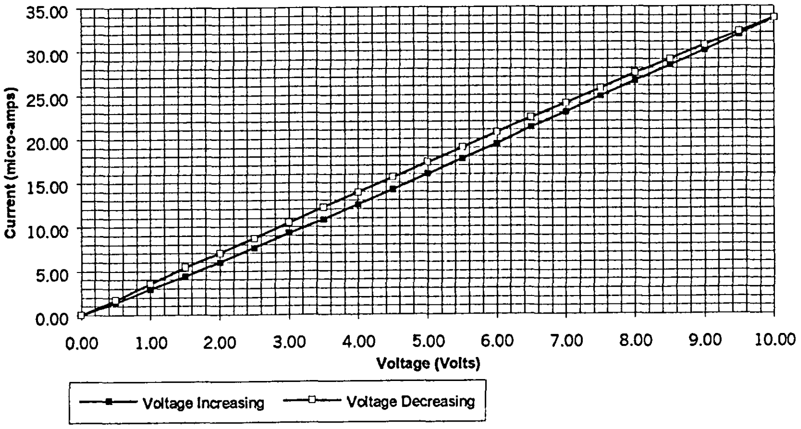


Figure 4.3.3.1.2 Sample 2; V/I characteristics

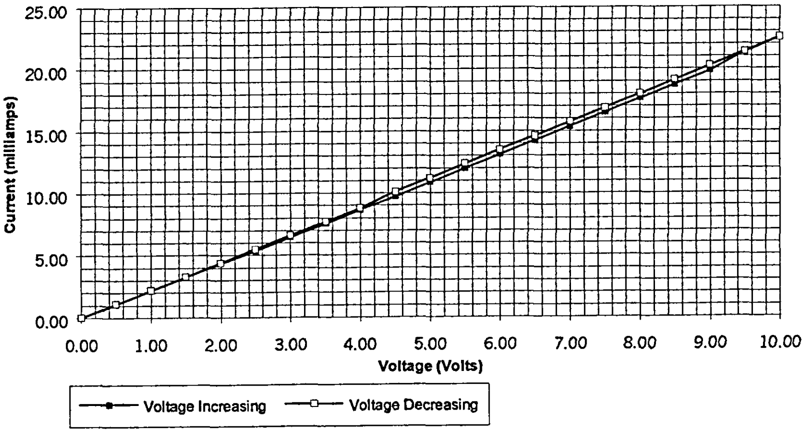


Figure 4.3.3.1.3 Sample 3; V/I characteristics

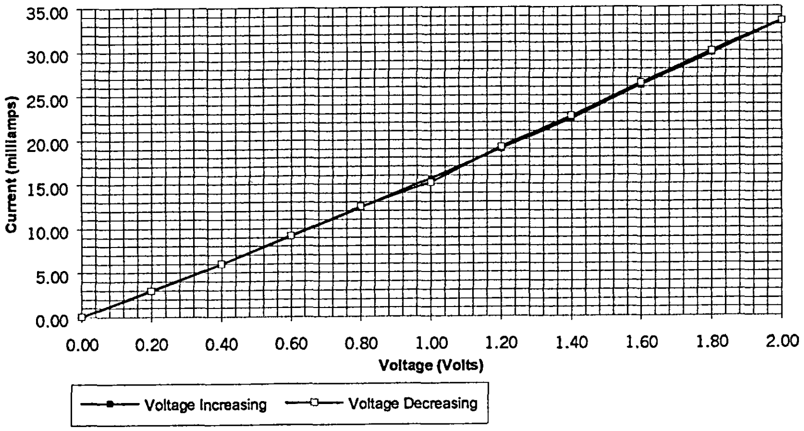


Figure 4.3.3.1.4 Sample 4; V/I characteristics

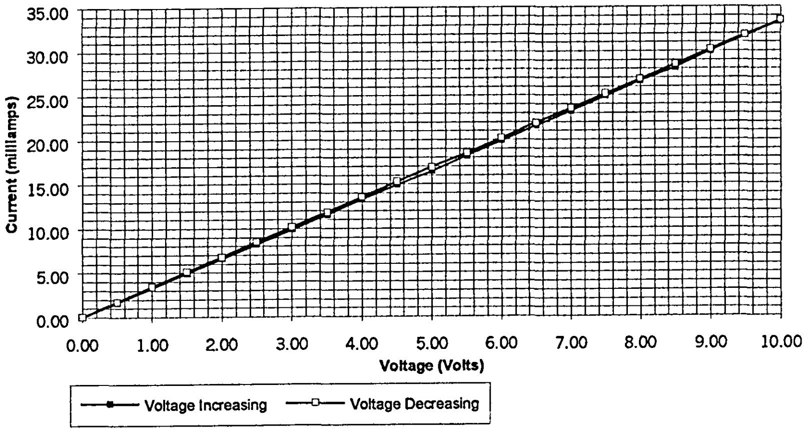


Figure 4.3.3.1.5 Sample 5; V/I characteristics

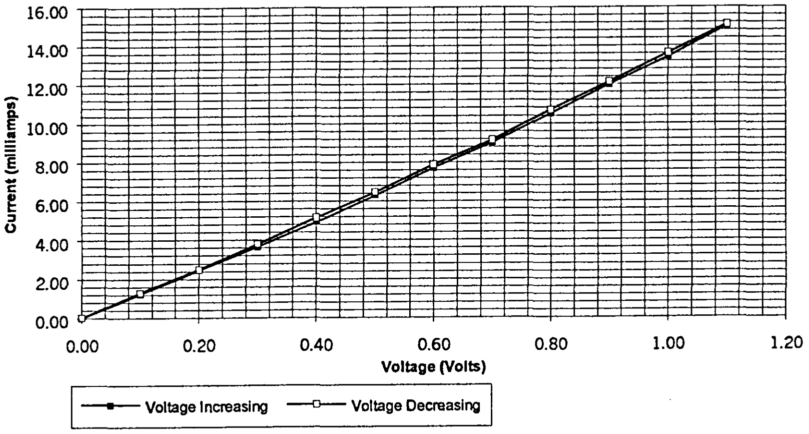


Figure 4.3.3.1.6 Sample 6; V/I characteristics

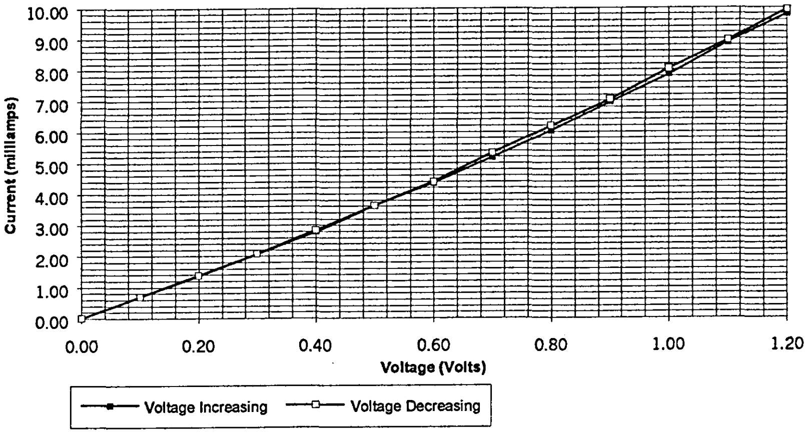


Figure 4.3.3.1.7 Sample 7; V/I characteristics

4.3.3.2 Creep characterisation

This experiment was designed to determine the short-term drift characteristics of the samples. A load of 89 kPa is applied to the area between electrodes 2 and 3 and the voltage applied to the sample (Vin) is kept constant at a level such that the voltage Vr is set at half its maximum value as determined by its V/I characteristics (4.3.1). The current (A) drawn by the samples is measured using a Hameg 8011-3 multimeter and the voltage Vr is measured using a Thurbly 1905a multimeter. Figures 4.3.3.2.1 to 4.3.3.2.7 show the change in the resistance of the samples over a 100 second period.

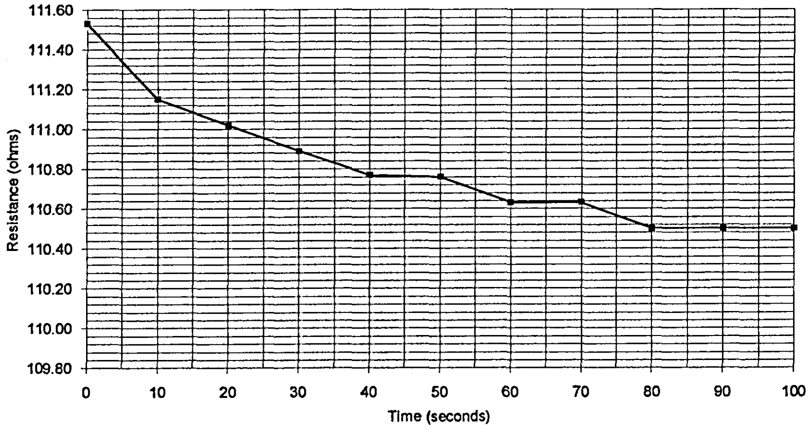


Figure 4.3.3.2.1 Sample 1; Creep Characteristics

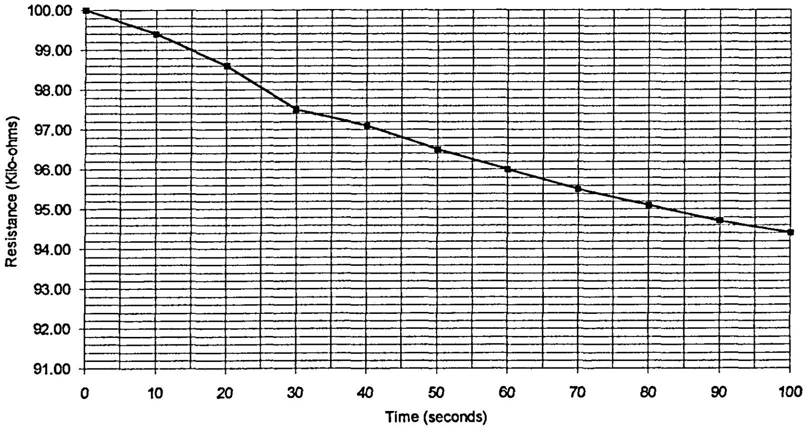


Figure 4.3.3.2.2 Sample 2; Creep Characteristics

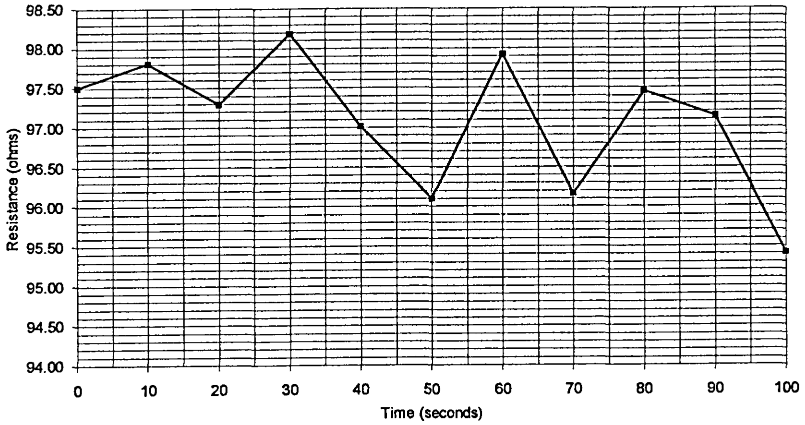


Figure 4.3.3.2.3 Sample 3; Creep Characteristics

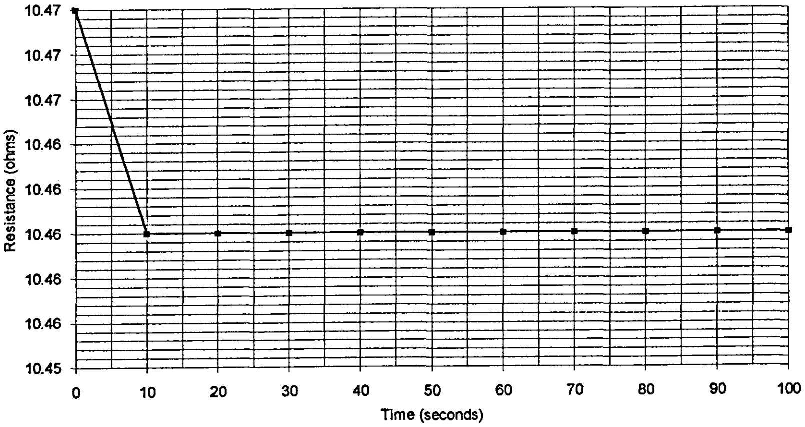


Figure 4.3.3.2.4 Sample 4; Creep Characteristics

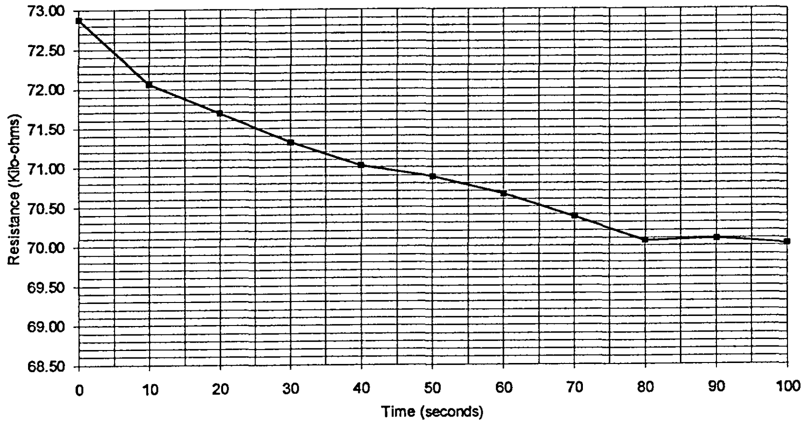


Figure 4.3.3.2.5 Sample 5; Creep Characteristics

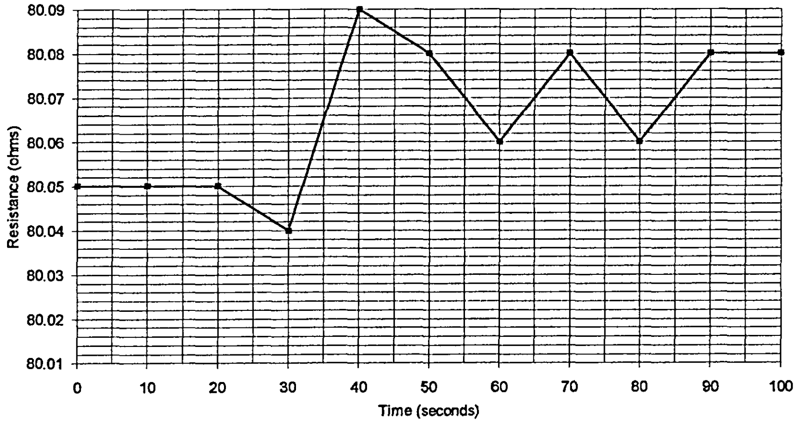


Figure 4.3.3.2.6 Sample 6; Creep Characteristics

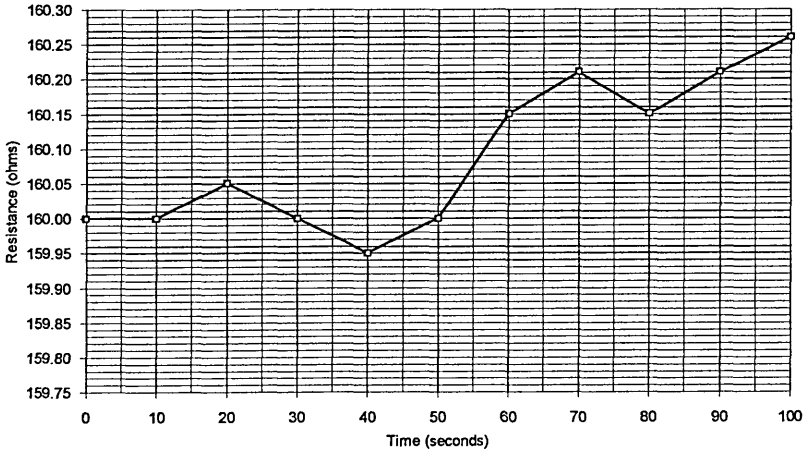


Figure 4.3.3.2.7 Sample 7; Creep Characteristics

4.3.3.3 Load characterisation

The experiment was designed to determine the load characteristics of the samples i.e. the change in the samples resistance for a given applied pressure. The applied voltage (Vin) is kept constant at half the maximum allowed value as determined by the V/I characterisation experiments (4.3.3.1). The voltage (Vr) is measured using a Thurbly 1905a multimeter and current was measured using a Hameg 8011-3 multimeter. The load is applied to an area of 1cm by 1cm between electrodes 2 and 3, except samples 6 and 7 where it's applied over the full area due to the size of these samples. Figures 4.3.3.3.1 to 4.3.3.3.7 show the load characteristics of the samples over the range of loads used.

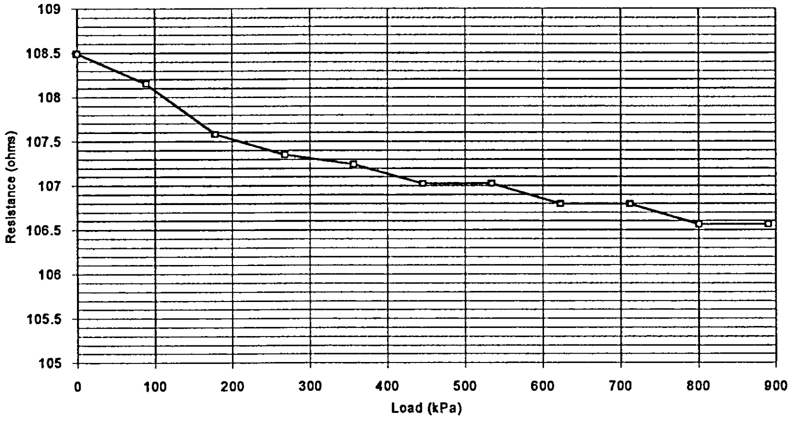


Figure 4.3.3.3.1 Sample 1; Load Characteristics

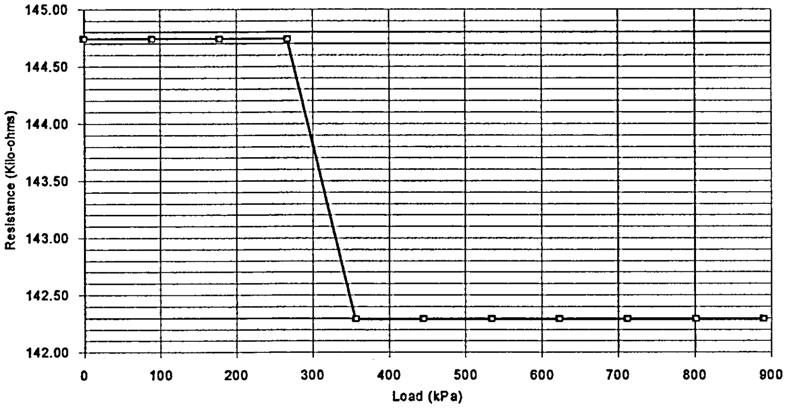


Figure 4.3.3.3.2 Sample 2; Load Characteristics

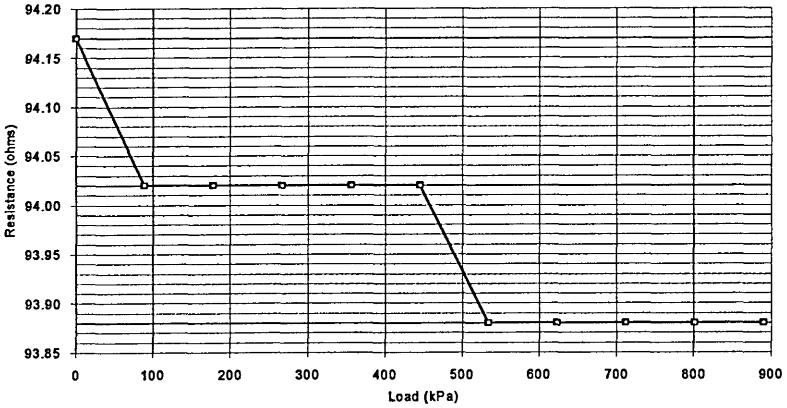


Figure 4.3.3.3.3 Sample 3; Load Characteristics

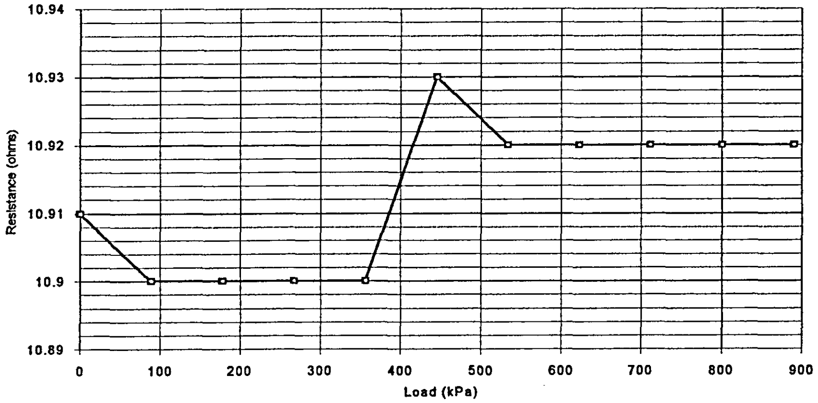


Figure 4.3.3.3.4 Sample 4; Load Characteristics

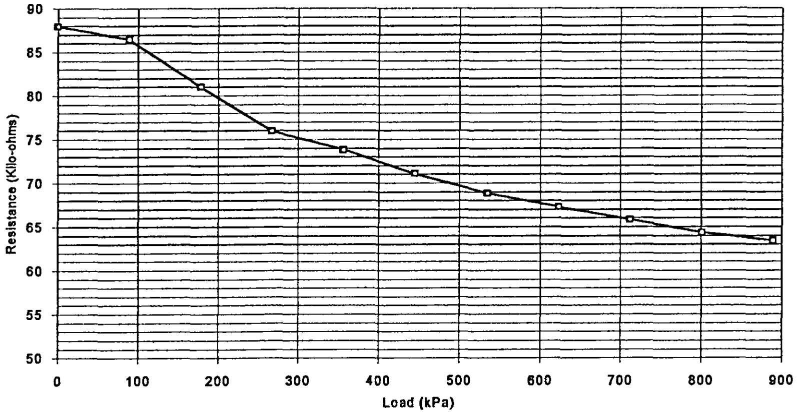


Figure 4.3.3.3.5 Sample 5; Load Characteristics

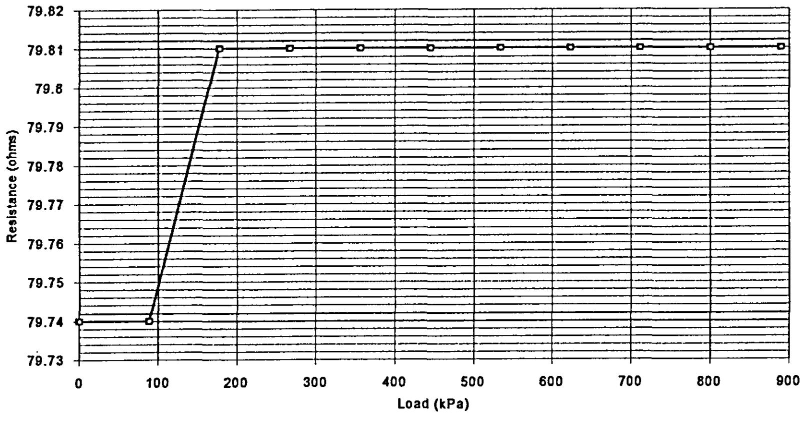


Figure 4.3.3.3.6 Sample 6; Load Characteristics

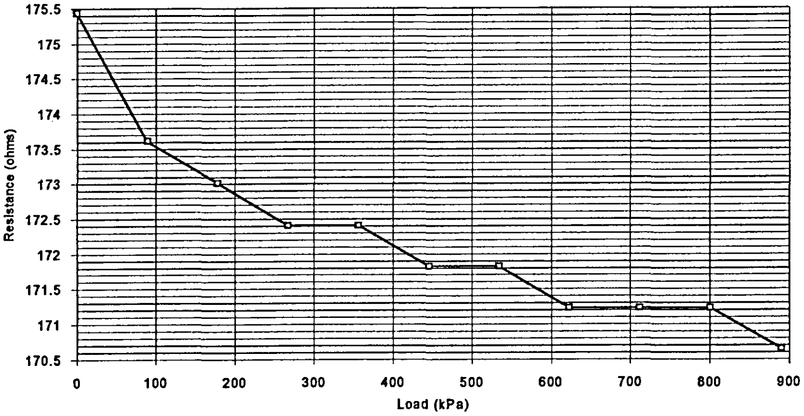


Figure 4.3.3.3.7 Sample 7; Load Characteristics

4.3.3.4 Conclusions

In this chapter the performance of a number of conductive polymer samples have been investigated in terms of their V/I characteristics, creep characteristics and load characteristics. From Figure 4.3.3.1.1 to 4.3.3.1.7, it can be seen that all the samples have reasonable ohmic behaviour i.e. there is little if any non-linearity in the V/I characteristics. However, there is some hysteresis in the V/I characteristics for all the samples between increasing and decreasing the applied voltage with the current being higher when decreasing the applied voltage. Sample 2 has the greatest V/I hysteresis which can be clearly seen in Figure 4.3.3.1.2. The hysteresis present could be due to the temperature changes caused by the power dissipation within the samples due to the applied current (i.e. the temperature coefficient of resistance of the samples) or due to the conduction process within the samples.

Table 4.3.3.4.1 shows the percentage change in resistance between the voltage being increased to the maximum voltage for the sample and then being reduced. The values quoted for percentage change in hysteresis are at half the maximum applied voltage for each sample. This voltage level was chosen as it was also the voltage used in the creep and load characterisation experiments.

Sample Number	**% Change InV/I Characteristics At Half Vs (Hysteresis)**	**% Change In Resistance Due to Creep (0s to 100s)**	**% Change In Resistance Due To Applied Load (min. to max.)**
1	-0.14	-1.04	-1.78
2	8.13	-5.60	-1.69
3	3.24	-2.14	-0.31
4	1.35	-0.10	0.09
5	2.92	-3.91	-27.91
6	2.33	0.04	0.13
7	0.92	0.16	-2.73

Table 4.3.3.4.1 Percentage change in hysteresis, creep and change in applied load for each sample characterised

From Table 4.3.4.1 it can be seen that only samples 1 and 7 have hysteresis of less than 1%. However, if the current density applied to the sample was sufficiently low then the amount of hysteresis present may not prove a problem for an EIT system. That is, the variation in impedance caused by this hysteresis would not be detectable by the EIT system. The more important characteristics of the samples are the percentage impedance variations due to creep (Figures 4.3.3.2.1 to 4.3.3.2.7) and due to the applied load (Figures 4.3.3.3.1 to 4.3.3.7).

Table 4.3.3.4.1 also shows that samples 5 and 7 have variations in impedance due to applied load of above 2% even with such a large load 889 kPa. Although sample 5 has a large variation in impedance due to the applied load (-27.91%), the percentage variations in hysteresis (2.92%) and creep (-3.91%) are also high which makes it an unsuitable material for the proposed sensor. Only sample 7 (polypyrrole) showed some promise toward the sensor material, unfortunately further experiments on this sample were not possible due to supply problems.

Sample 7 was an inherently conductive polymer (Kosina S., Baluch S. and Annus J., 1994) produced by electrochemical polymerisation and an investigation into producing this polymer in-house was undertaken ((Kosina S., Baluch S. and Annus J., 1994 and Naarmann H., 1990). Unfortunately, due to the costs of the chemicals, rheostat and gold/platinum electrodes needed to produce polypyrrole, this option could not be researched further. Thus, no suitable material from a reliable source could be found for the proposed distributed planar pressure sensor. An important point is that samples 1 to 5 are compounds of carbon and polymers/rubbers where as samples 5 and 7 are inherently conductive rubbers. Therefore it may be possible to develop an inherently conductive polymer for the sensor element. There was clearly a need for an alternative sensor design and this is presented in Chapter 6.

5. Electrode Configurations for E.I.T.

5.1 Introduction

There are several applications of EIT where the electrodes need not be placed at the periphery of the object. Applications for alternative electrode schemes exist in areas such as the measurement of distributed pressure, where a conductive pressure sensitive planar sensor element can have electrodes placed behind it (Lacey P. and Basarab-Horwath I., 1993). Also, in process mixing, where the electrodes can be mounted on the centrally placed stirrer(s) (Lyon G.M. and Oakley J.P., 1992). This Chapter details work, which compares three different electrode configurations. The three electrode configurations are shown in Figure 5.1.1. Phantom is the term used to describe the area or volume being interrogated and is a term adopted from medical imaging (see chapter 3).

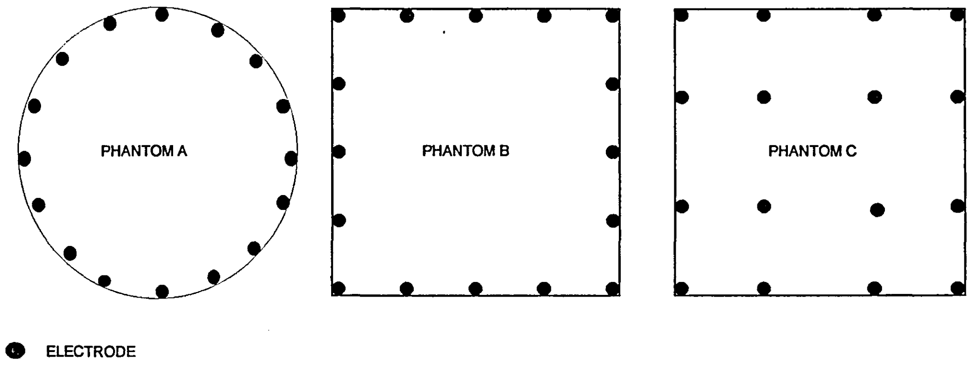


Figure 5.1.1 Electrode configurations,
Phantom A, classical peripheral configuration; Phantom B, peripheral configuration square phantom; Phantom C, distributed configuration square phantom

It is known that the peripheral mounted electrode system has limited resolution towards the centre of the interrogated object (Lyon G.M. and Oakley J.P., 1992). Lyon G. M. and Oakley J. P., 1992 showed that placing electrodes near to the centre of the object improves the resolution within the central area. The classical EIT problem is ill conditioned and it has been shown theoretically that placing electrodes near to the centre reduces this ill conditioning (Fulton W.S. and Lipczynski R.T., 1993). The classical method of dealing with this ill conditioning is by regularisation (Avis N.J. and Barber D.C., 1994), which has the effect of degrading image quality. With distributed electrodes, less regularisation should be needed and hence image quality should be improved throughout the interrogated area. It is known that the adjacent electrode current drivė gives the best spatial resolution, whereas the polar drive (diametrically opposite current injection) provides the best signal-to-noise ratio (Avis N.J. and Barber D.C., 1994).

The three electrode configurations use the adjacent electrode drive configuration. With the distributed electrode configuration however because the electrodes are distributed throughout the area being imaged some in effect function as a polar drive for any given anomaly. The three different electrode configurations examined were:

1. The 'classical' circular arrangement, with the electrodes placed on the periphery of a circular phantom,
2. A square arrangement with the electrodes placed on the periphery of a square phantom.

3. A distributed arrangement where the electrodes are evenly distributed over a square phantom;

With both the peripheral electrode configurations the adjacent electrode pair current injection drive gives 16 projections with 13 differential voltage measurements per projection giving a total of 208 readings. Applying the adjacent current injection drive to the distributed electrode configuration on all electrode pairs both vertically and horizontally gives 24 projections. The differential voltage are also measured between all non-current carrying vertically and horizontally adjacent electrodes, giving a different number of voltage readings for each projection of between 14 and 19 readings. This results in 448 differential voltage readings for the distributed electrode configuration.

5.2 Measure of Performance

Before defining the measures of performance used to compare the performance of the three electrode arrangements the terms used are described:

1. A_r, the normalised area is the ratio of the anomaly area (A_a) to the background area (A_b), that is

$$A_n = A_a / A_b \ , \tag{5.2.1}$$

2. V is the measured boundary voltage with an anomaly present in the phantom
3. V_u is the measured boundary voltage with no anomaly present: that is, a uniform conductivity distribution.

4. σ_n the conductivity contrast which is the ratio of anomaly conductivity (σ_a) to background conductivity (σ_b), that is

$$\sigma_n = \sigma_a / \sigma_b \tag{5.2.2}$$

The measures of performance are defined as:

1. $\delta V/V$, the fractional change in boundary voltage is

$$\delta V / V = \frac{1}{N} \sum_{i=1}^{N} (V - V_u) / V_u \tag{5.2.3}$$

2. Q, the visibility of an anomaly (Seagar A.D., Barber D.C. and Brown B.H., 1987) is

$$Q = \frac{1}{N} \sum_{i=1}^{N} \frac{(V - V_u)}{(V + V_u)} \tag{5.2.4}$$

Where N is the total number of voltage readings for each electrode configuration.

5.3 Experimental comparison

The experimental comparison is performed using the EIT hardware described in Chapter 3.2. The three phantoms are perspex tanks filled to a height of 2cm with a saline solution. The electrodes are just higher than 2cm so that only two-dimensional effects need be considered. To compare the three electrode configurations the size of circular conducting and non-conducting anomalies placed at the centre of each phantom was varied. The anomaly size is limited in the case of the distributed configuration due to the placement of the electrodes.

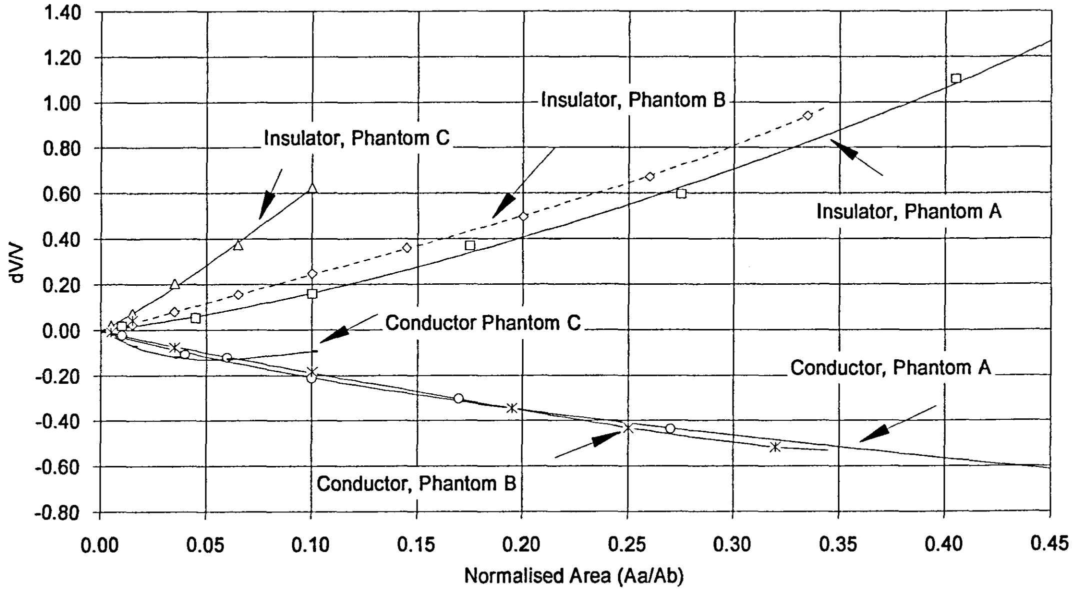


Figure 5.3.1 Fractional Change in Voltage Against Normalised Area.

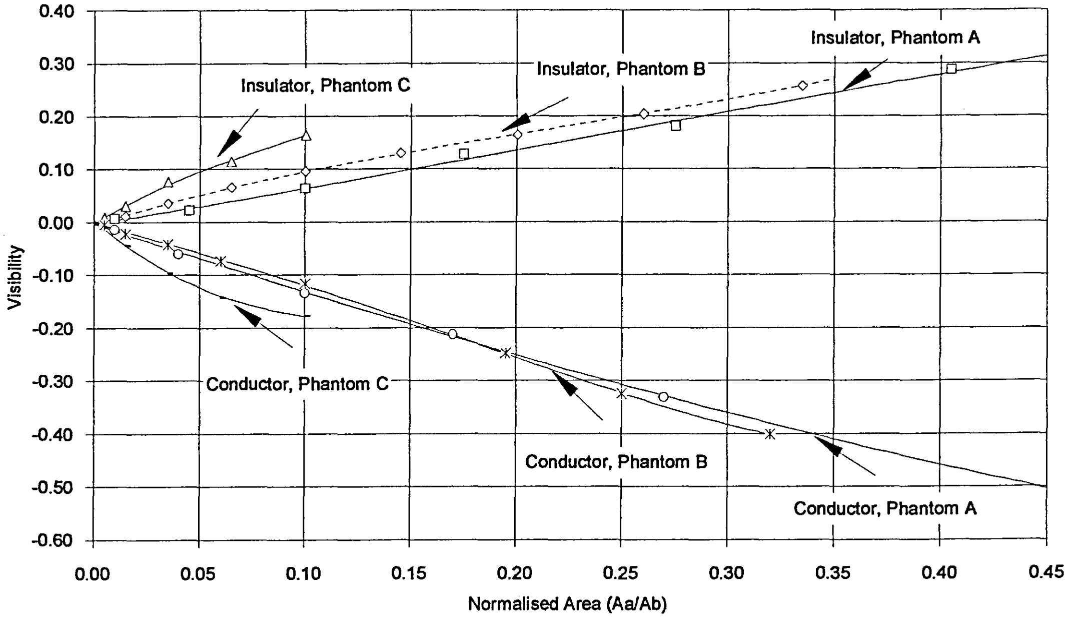


Figure 5.3.2 Visibility Against Normalised Area.

It is shown in Figures 5.3.1 and 5.3.2 that the values for both visibility and fractional change in voltage are higher with Phantom C than either Phantom A or B. It is also shown that for the insulator anomaly Phantom B performs better than Phantom A.

The three phantoms are also compared experimentally using the sensitivity coefficients used in the reconstruction algorithm as described in Chapter 3.4. Figure 5.3.3 shows the cross section through the sensitivity profiles from the left hand side of the phantom to the centre for each of the phantoms.

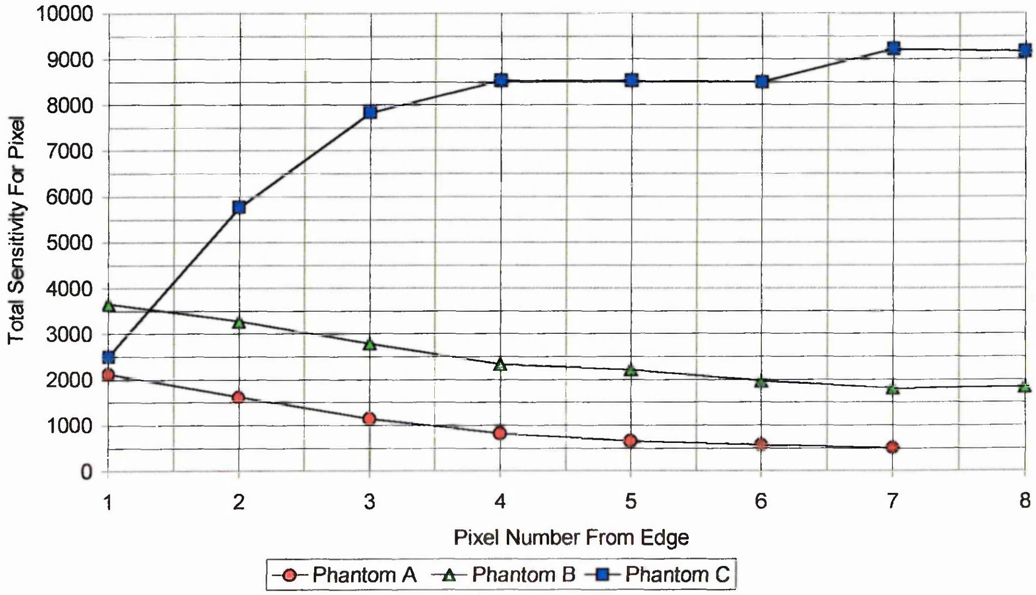


Figure 5.3.3 Cross Section of Sensitivity Profile For Each Phantom For Left hand Side Of Phantom To Centre

It is shown in Figure 5.3.3 that Phantom C has a higher sensitivity value than both Phantoms A and B and in particular has a higher sensitivity at the centre, which is the reverse of both A and B Phantoms. As is observed in both the Visibility and Fractional Change in Voltage Phantom C has a higher sensitivity than Phantom A or B

5.4 Simulation

The simulation is base on a resistor network and was used to build a two dimensional model of the different electrode arrangements described. A number of researchers have used resistor networks in order to model the area to be image by an EIT system. Griffiths H., 1988 simulated the area by building a physical model using a resistor mesh. The resistor analogy was also used by Daniels A. R., 1996, to construct both two and three-dimensional models using HSPICE to analyse EIT for multi component flows. Knight R. A., 1991, used nodal analysis and Finite element methods to compare different reconstruction algorithms used in EIT and the simulation described here is based on this work.

The first step is to divide the phantom area into a number of discrete elements. The circular phantom with peripheral electrodes was divided into 241 elements as shown in Figure 5.4.1. The square phantom with peripheral electrodes was divided into 289 elements as shown in Figure 5.4.2. The square phantom with distributed electrodes was divided into 256 elements as shown in Figure 5.4.3. The square phantoms have a different number of elements so that the number of discrete elements between each electrode is equal for that phantom see Figures 5.4.6 to 5.4.8 for electrode positions.

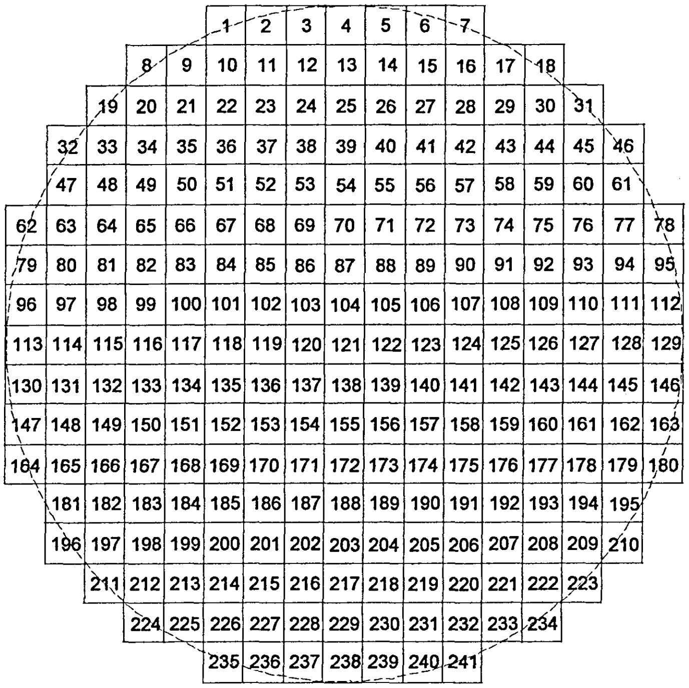


Figure 5.4.1 Circular Phantom Divided into 241 Elements

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	46	45	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102
103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187
188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204
205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221
222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238
239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272
273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289

Figure 5.4.2 Square Phantom Divided into 289 Elements

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Figure 5.4.3 Square Phantom Divided into 256 Elements

The next step is to construct a resistor network for each element. The resistor network used for the simulation is shown in Figure 5.4.4.

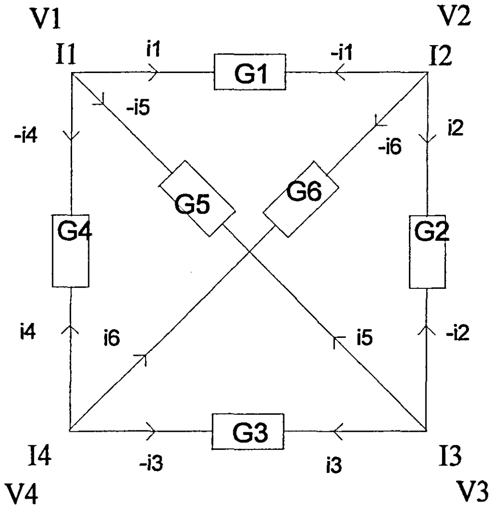


Figure 5.4.4 Resistor network Used for Each Element in the Simulations

Kirchoff's current laws are used to analysis the current flowing into each node at the four corners of the network. The analysis of the current flowing into each node leads to equations 5.4.1 to 5.4.4.

$$i1 - i4 - i5 = I1 \tag{5.4.1}$$

$$i2 - i1 - i6 = I2 \tag{5.4.2}$$

$$i3 - i2 + i5 = I3 \tag{5.4.3}$$

$$i4 - i3 + i6 = I4 \tag{5.4.4}$$

By substituting, the nodal voltages and conductance values of each resistor element in the network into equations 5.4.1 to 5.4.4 equations 5.4.5 to 5.4.8 are obtained.

$$(V1\text{-}V2)G1 - (V4\text{-}V1)G4 - (V3\text{-}V1)G5 = I1 \tag{5.4.5}$$

$$(V2\text{-}V3)G2 - (V1\text{-}V2)G1 - (V4\text{-}V2)G6 = I2 \tag{5.4.6}$$

$$(V3\text{-}V4)G3 - (V2\text{-}V3)G2 + (V3\text{-}V1)G5 = I3 \tag{5.4.7}$$

$$(V4\text{-}V1)G4 - (V3\text{-}V4)G3 + (V4\text{-}V2)G6 = I4 \tag{5.4.8}$$

Equations 5.4.5 to 5.4.8 can be rearrange in order to group together the nodal voltages and these are given in equations 5.4.9 to 5.4.12

$$V1(G1 + G4 + G5) - V2G1 - V3G5\text{ - }V4G4 = I1 \tag{5.4.9}$$

$$\text{-}V1G1 + V2(G2 + G1 + G6) - V3G2\text{ - }V4G6 = I2 \tag{5.4.10}$$

$$\text{-}V1G5 - V2G2 + V3(G3 + G2 + G5) - V4G3 = I3 \tag{5.4.11}$$

$$\text{-}V1G4 - V2G6 - V3G3 + V4(G4 + G3 + G6) = I4 \tag{5.4.12}$$

Equations 5.4.9 to 5.4.12 can now be expressed using matrix notation as shown in equation 5.4.13

$$|G|\cdot\underline{\mathrm{V}}=\underline{\mathrm{I}} \tag{5.4.13}$$

Where

$$|G|=\begin{bmatrix}(\mathrm{G1}+\mathrm{G4}+\mathrm{G5}) & -\mathrm{G1} & -\mathrm{G5} & -\mathrm{G4}\\ -\mathrm{G1} & (\mathrm{G2}+\mathrm{G1}+\mathrm{G6}) & -\mathrm{G2} & -\mathrm{G6}\\ -\mathrm{G5} & -\mathrm{G2} & (\mathrm{G3}+\mathrm{G2}+\mathrm{G5}) & -\mathrm{G3}\\ -\mathrm{G4} & -\mathrm{G6} & -\mathrm{G3} & (\mathrm{G4}+\mathrm{G3}+\mathrm{G6})\end{bmatrix},$$

$$\underline{V}=\begin{bmatrix}\mathrm{V1}\\ \mathrm{V2}\\ \mathrm{V3}\\ \mathrm{V4}\end{bmatrix} \quad \text{And} \quad \underline{\mathrm{I}}=\begin{bmatrix}\mathrm{I1}\\ \mathrm{I2}\\ \mathrm{I3}\\ \mathrm{I4}\end{bmatrix}$$

In the simulation, all the resistor values are set to 1Ω and by substituting these values into equation 5.4.13 |G| becomes G_{ele} as shown below

$$G_{ele}=\begin{bmatrix}3 & -1 & -1 & -1\\ -1 & 3 & -1 & -1\\ -1 & -1 & 3 & -1\\ -1 & -1 & -1 & 3\end{bmatrix}$$

The next step in the simulation process is to map the individual element matrix G_{ele} into a systems matrix G_{sys}. Consider the simple case shown in Figure 5.4.5 where an area has been divided into 4 discrete elements each single element being the resistor network described above in Figure 5.4.4.

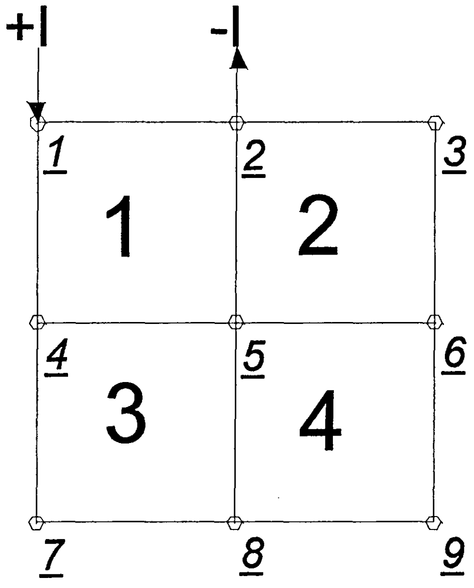


Figure 5.4.5 Simple Resistor Simulation with A Current Of 1 Amp Injected Between Nodes 1 and 2

Each element in the simple case of Figure 5.4.4 is numbered 1 to 4 and all the nodes in the network are numbered globally 1 to 9. The local node numbers of the single resistor element (1 to 4) are mapped to global nodes (1 to 9) which is known as the system topology scheme. For the simple case above, the mapping between local and global nodes for each element is given in table 5.4.1.

Element Number	**Local Node Number**			
	1	**2**	**3**	**4**
1	1	2	5	4
2	2	3	6	5
3	4	5	8	7
4	5	6	9	8
	Global Node Numbers			

Table 5.4.1 Simple Resistor Model System Topology Scheme

Element 1's local nodes are mapped to global nodes1, 2, 5 and 4 the individual elements of the elements matrix G_{elc} are therefore mapped into the system matrix G_{sys} as detailed below:

$$G_{sys(1,1)} = G_{ele(1,1)} = 3, \qquad G_{sys(1,2)} = G_{ele(1,2)} = -1$$

$$G_{sys(1,5)} = G_{ele(1,3)} = -1, \qquad G_{sys(1,4)} = G_{ele(1,4)} = -1$$

$$G_{sys(2,1)} = G_{ele(2,1)} = -1, \qquad G_{sys(2,2)} = G_{ele(2,2)} = 3$$

$$G_{sys(2,5)} = G_{ele(2,3)} = -1, \qquad G_{sys(2,4)} = G_{ele(2,4)} = -1$$

$$G_{sys(5,1)} = G_{ele(3,1)} = -1, \qquad G_{sys(5,2)} = G_{ele(3,2)} = -1$$

$$G_{sys(5,5)} = G_{ele(3,3)} = 3, \qquad G_{sys(5,4)} = G_{ele(3,4)} = -1$$

$$G_{sys(4,1)} = G_{ele(4,1)} = -1, \qquad G_{sys(4,2)} = G_{ele(4,2)} = -1$$

$$G_{sys(4,5)} = G_{ele(4,3)} = -1, \qquad G_{sys(4,4)} = G_{ele(4,4)} = 3$$

Once all the element have been mapped into the system matrix G_{sys} is now as detailed below

$$G_{sys=}\begin{vmatrix} 3 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 6 & -1 & -1 & -2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 6 & -2 & 0 & -1 & 0 & 0 \\ -1 & -2 & -1 & -2 & 12 & -2 & -1 & -1 & -1 \\ 0 & -1 & -1 & 0 & -2 & 6 & 0 & -2 & -1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & -2 & -1 & -1 & 6 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 3 \end{vmatrix}$$

In order to solve equation 5.4.13 for the nodal voltages given the currents at external nodes 1 and 2 (i.e. +1 A injected between global node 1 and 2) equation 5.4.13 is rearrange to give equation 5.4.14

$$|G|^{-1} \cdot \underline{I} = \underline{V} \tag{5.4.14}$$

where $[G]^{-1} = G_{sys}$

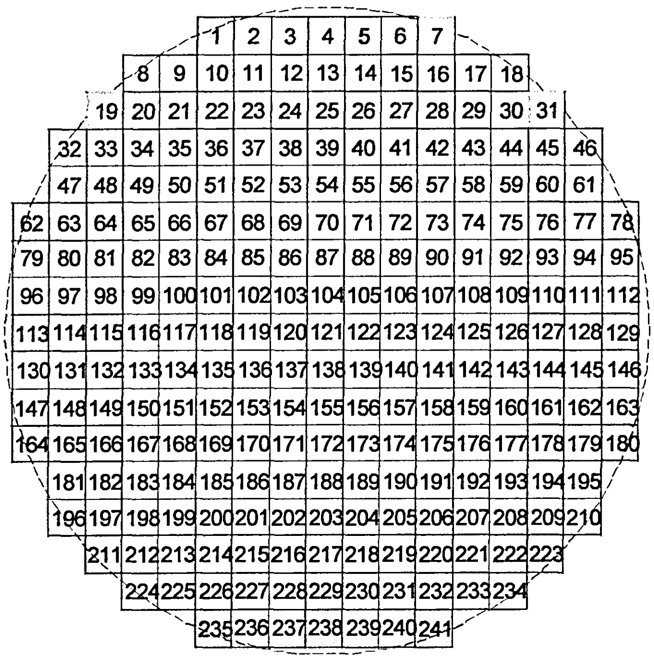
The G_{sys} matrix is however singular, a simple solution to this is to use the method described by Huebner K.H., 1975, to make G_{sys} non-singular. This is achieved by earthing one or more nodes (i.e. applying known boundary conditions). In the simple simulation describe the –I terminal is set to 0Volts (node 2) this is now incorporated into the system matrix G_{sys} by setting all the entries in column 2 and row 2 equal to 0 except at $G_{sys(2,2)}$ which is set to 1 the 2nd element of the I vector must also be set to 0. The other boundary condition that must also be included in equation 5.4.14 is the Value of +I at node 1. This is done by simply setting the i element of the current vector $\underline{I}$ to 1 and setting all others to 0 where i in this case is 1. Equation 5.4.13 can now be solved.

The simulation processes detailed above was implement for the phantoms show in Figures 5.4.1 to 5.4.3 using MATLAB and Appendix B includes the MATLAB "M" files used. The conductivity of individual element can be easily adjusted when the element matrix (G_{ele}) is being mapped into the system matrix. The algorithm used is given below:

```
For i = 1 to number of elements
    For j  = 1 to 4
        x = topology matrix(i,j)
        For k = 1 to 4
            y = topology matrix(i,k)
            Gsys (x,y) = Gsys(x,y) + (Gv(i) *Gele(j,k))
        End for k
    End for j
End for i
```

where Gv is a vector of the conductivity weightings for each element in the simulation.

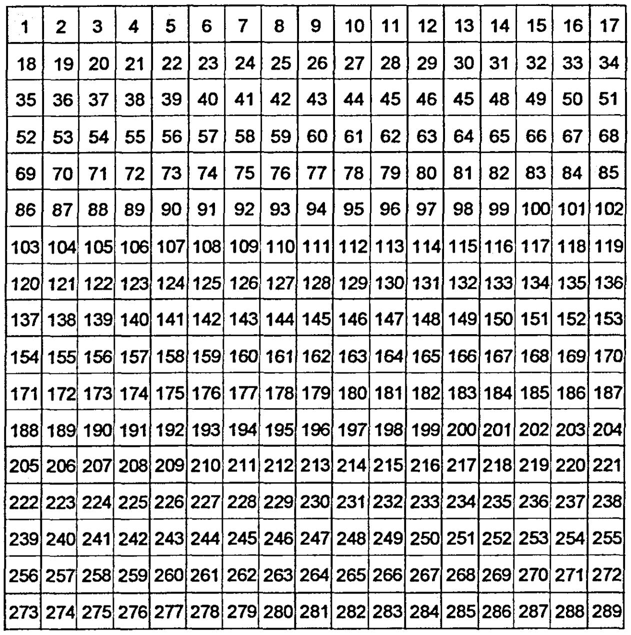
The conductivity of each element in the simulations that represent an electrode is set at 10000 S. the Location of the electrodes for each of the simulations is shown in Figures 5.4.6 to 5.4.8.



					1	2	3	4	5	6	7					
			8	9	10	11	12	13	14	15	16	17	18			
		19	20	21	22	23	24	25	26	27	28	29	30	31		
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	
	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	
62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78
79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146
147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163
164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	
	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	
		211	212	213	214	215	216	217	218	219	220	221	222	223		
			224	225	226	227	228	229	230	231	232	233	234			
					235	236	237	238	239	240	241					

n EIT Electrode

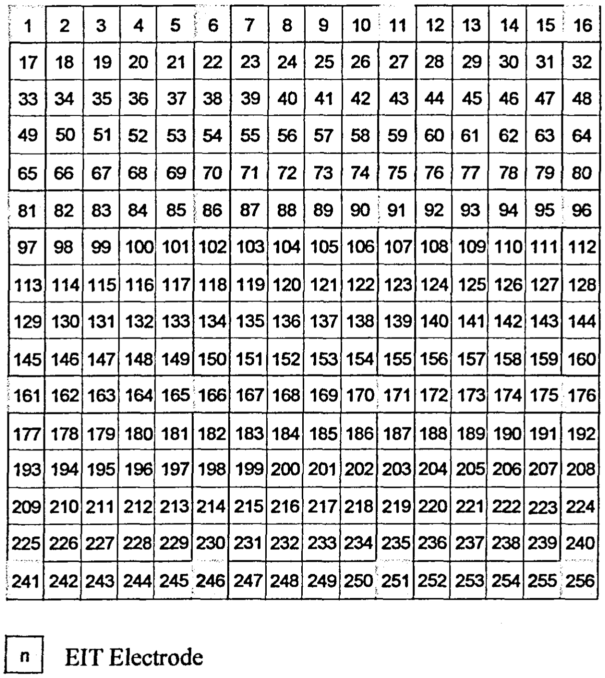
Figure 5.4.6 Simulated Circular Phantom Showing Location of Peripheral Electrodes



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	46	45	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102
103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187
188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204
205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221
222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238
239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272
273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289

n EIT Electrode

Figure 5.4.7 Simulated Square Phantom Showing Location of Peripheral Electrodes



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Figure 5.4.8 Simulated Square Phantom Showing Location of Distributed Electrodes

The adjacent electrode drive configuration is used in each simulation. However, the simulations do not include all possible projections for each of the phantoms. As the information from one projection will be the same as that form another similar projection if the conductivity distribution is homogenous or if the anomaly within the phantom is centrally located e.g. consider the circular phantom the projection when current is injected between electrodes at element 4 and 7 is the same as that for electrodes at 129 and 180. For both the circular and square phantoms, there are 4 projections and for the square phantom with the distributed electrode configuration there are 6 projections.

To verify the simulation the differential voltages recorded for each of the projections from both the simulation and EIT system are correlated and Tables 5.4.2 to 5.4.4 and Figures 5.4.9 to 5.4.11 show the results of the correlation.

	Sim 1	Sim 2	Sim 3	Sim 4
Data 1	**0.995**	0.983	0.973	0.988
Data 2	0.991	**0.979**	0.989	0.994
Data 3	0.987	0.968	**0.981**	0.994
Data 4	0.995	0.980	0.973	**0.989**

Table 5.4.2 Correlation Between Experimental Data (Data 1 –4) and Simulation Data (Sim 1- 4) for Circular Phantom With Homogenous Conductivity Distribution

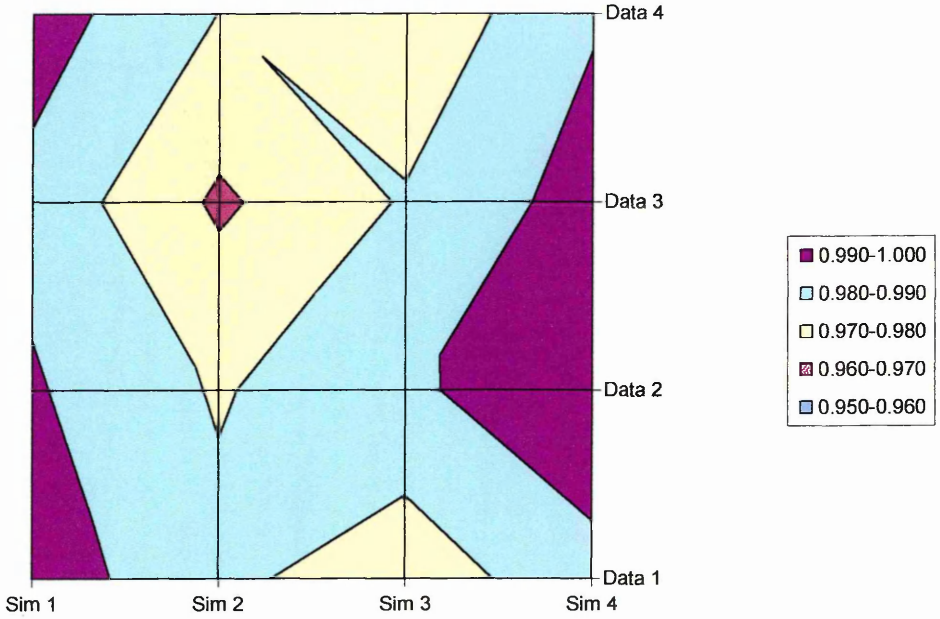


Figure 5.4.9 Correlation Between Experimental Data (Data 1 –4) and Simulation Data (Sim 1- 4) for Circular Phantom With Homogenous Conductivity Distribution

Table 5.4.2 shows that the correlation between simulated data and measured data from the EIT system for the circular phantom is between 0.979 and 0.995. This proves the validity of both the simulation data and the data obtained from the EIT system for a circular phantom.

	Sim 1	Sim 2	Sim 3	Sim 4
Data 1	**0.999**	0.789	0.247	0.492
Data 2	0.819	**0.999**	0.218	0.251
Data 3	0.258	0.242	**0.999**	0.809
Data 4	0.480	0.235	0.798	**1.000**

Table 5.4.3 Correlation Between Experimental Data (Data 1 –4) and Simulation Data (Sim 1 - 4) for Square Phantom With Peripheral Electrodes for Homogenous Conductivity Distribution

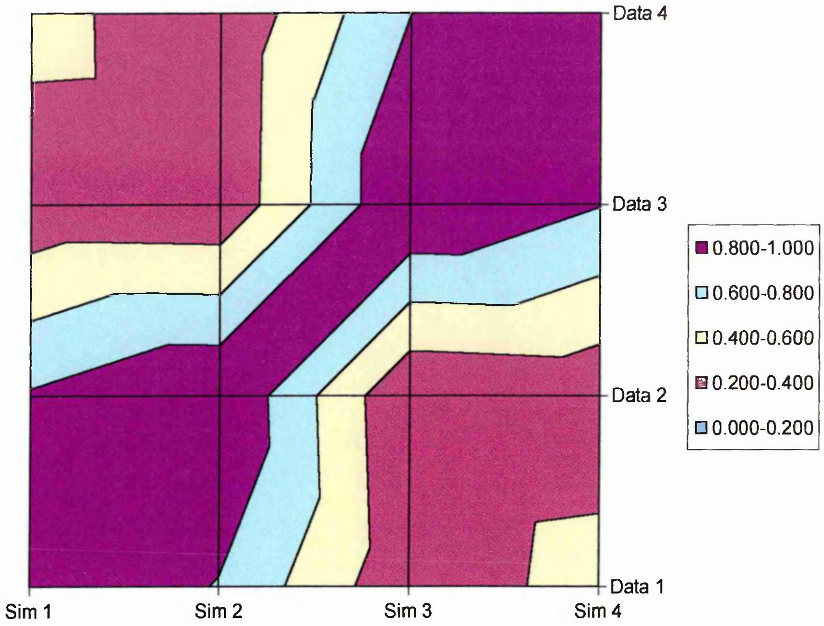


Figure 5.4.10 Correlation Between Experimental Data (Data 1 –4) and Simulation Data (Sim 1 - 4) for Square Phantom With Peripheral Electrodes for Homogenous Conductivity Distribution

Table 5.4.3 shows that the correlation between simulated data and measured data from the EIT system for the square phantom with peripheral electrodes is between 0.999 and 1. This proves the validity of both the simulation data and the data obtained from the EIT system for this phantom.

	Sim 1	Sim 2	Sim 3	Sim 4	Sim5	Sim 6
Data 1	**0.998**	0.640	-0.225	0.276	0.678	0.227
Data 2	0.643	**0.998**	-0.393	0.214	0.531	0.110
Data 3	0.245	-0.362	**0.998**	-0.195	-0.282	0.001
Data 4	0.292	0.198	-0.156	**0.998**	0.487	-0.357
Data 5	0.715	0.594	-0.262	0.439	**0.999**	0.083
Data 6	0.220	0.099	0.059	-0.357	0.048	**0.999**

Table 5.4.4 Correlation Between Experimental Data (Data 1 –6) and Simulation Data (Sim 1 - 6) for Square Phantom With Distributed Electrodes for Homogenous Conductivity Distribution

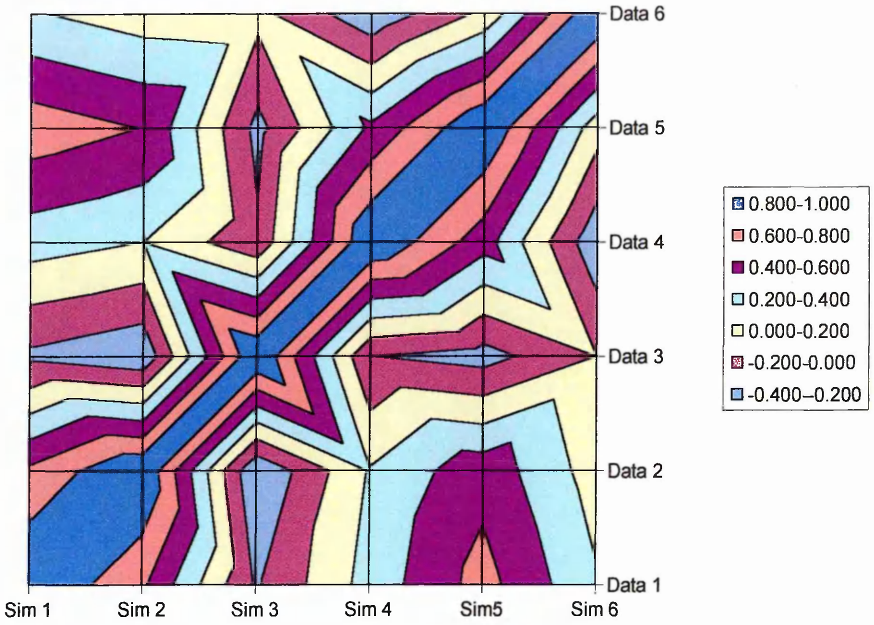


Figure 5.4.11 Correlation Between Experimental Data (Data 1 –6) and Simulation Data (Sim 1 - 6) for Square Phantom With Distributed Electrodes for Homogenous Conductivity Distribution

Table 5.4.4 shows that the correlation between simulated data and measured data from the EIT system for the square phantom with distributed electrodes is between 0.999 and 1. This proves the validity of both the simulation and the data obtained from the EIT system for this phantom.

A series of simulations for each of the phantoms were made that included the presene of anomalies which varied in both size and conductivity. These were placed at the centre of each phantom and the differential voltages for each of the projections recorded. Table 5.4.5 show the normalised area of each of the anomalies for each of the phantoms and the conductivity was varied from 0.001 to 1000 S with the background conductivity remaining at 1 S.

Anomaly Number	Circular Phantom Peripheral Electrodes Aa/Ab	Square Phantom Peripheral Electrodes Aa/Ab	Square Phantom Distributed Electrodes Aa/Ab
1	0.00415	0.00346	0.01563
2	0.03734	0.03114	0.06250
3	0.10373	0.08651	0.14063
4	0.20332	0.16955	0.25000
5	0.33610	0.28028	0.03916
6	0.50207	0.41869	0.56250
7	0.70124	0.58478	0.76563
8	Not Simulated	0.77855	Not Simulated

Table 5.4.5 Simulated Anomaly Numbers Normalised Area For Each Of The Phantoms

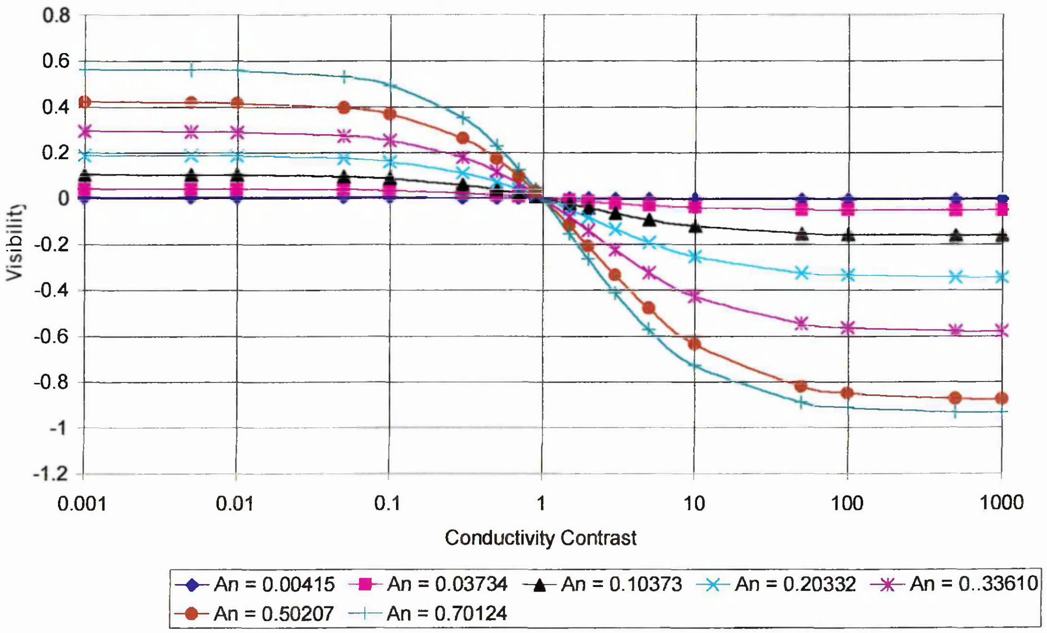


Figure 5.4.12 Visibility against Conductivity Contrast for Centrally Placed Anomalies in Circular Phantom with Peripheral Electrodes (Phantom A)

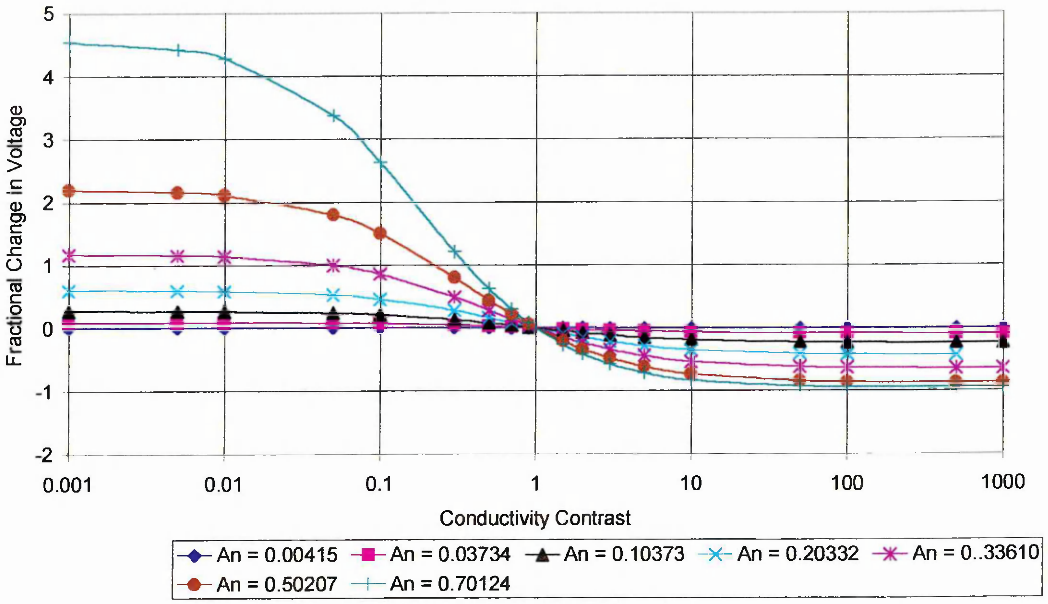


Figure 5.4.13 Fractional Change in Voltage against Conductivity Contrast for Centrally Placed Anomalies in Circular Phantom with Peripheral Electrodes (Phantom A)

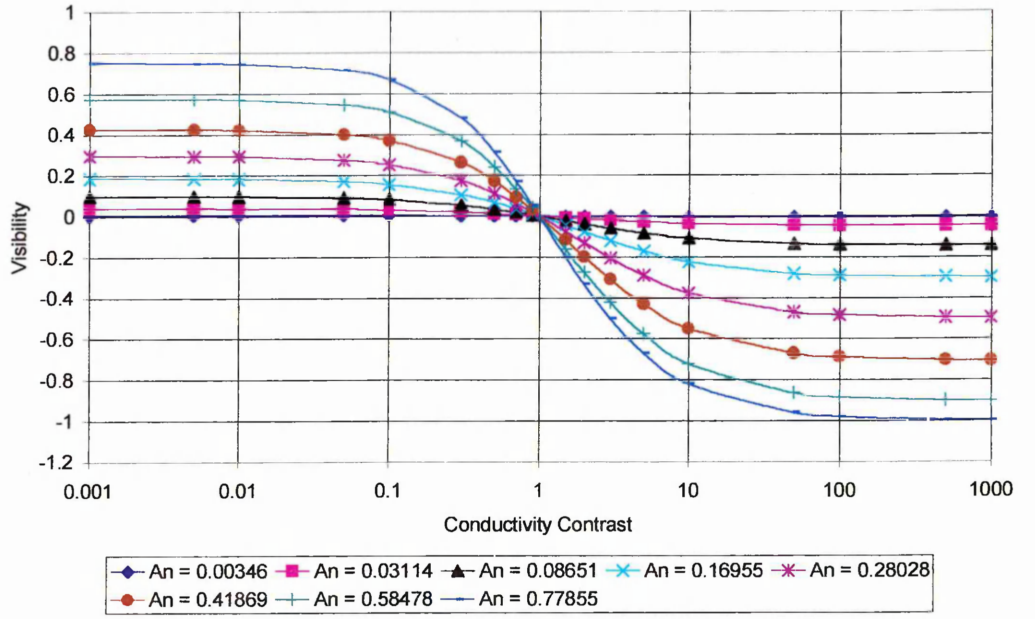


Figure 5.4.14 Visibility against Conductivity Contrast for Centrally Placed Anomalies in Square Phantom with Peripheral Electrodes (Phantom B)

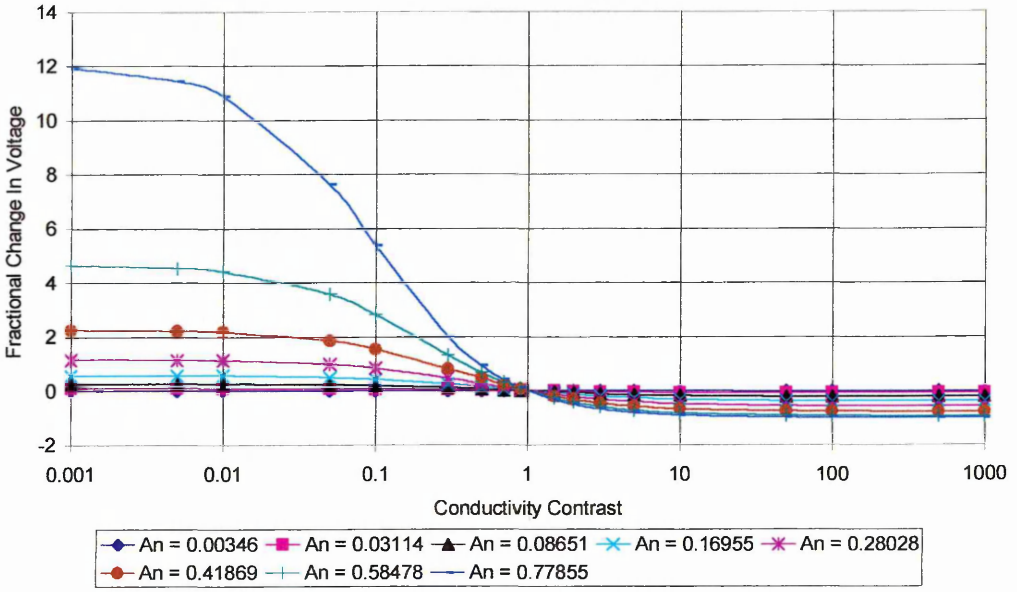


Figure 5.4.15 Fractional Change in Voltage against Conductivity Contrast for Centrally Placed Anomalies in Square Phantom with Peripheral Electrodes (Phantom B)

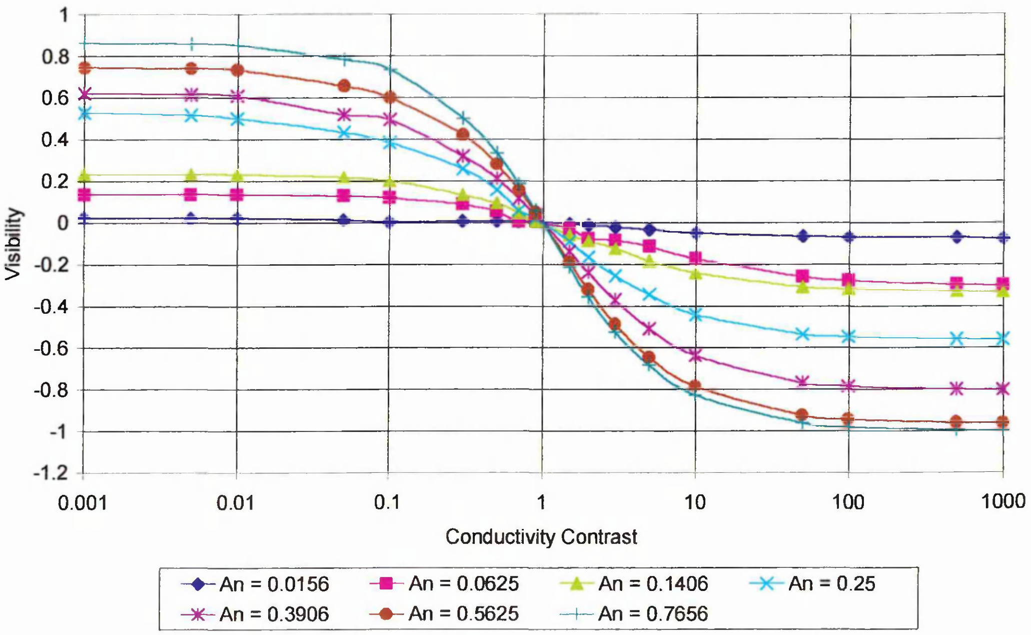


Figure 5.4.16 Visibility against Conductivity Contrast for Centrally Placed Anomalies in Square Phantom with Distributed Electrodes (Phantom C)

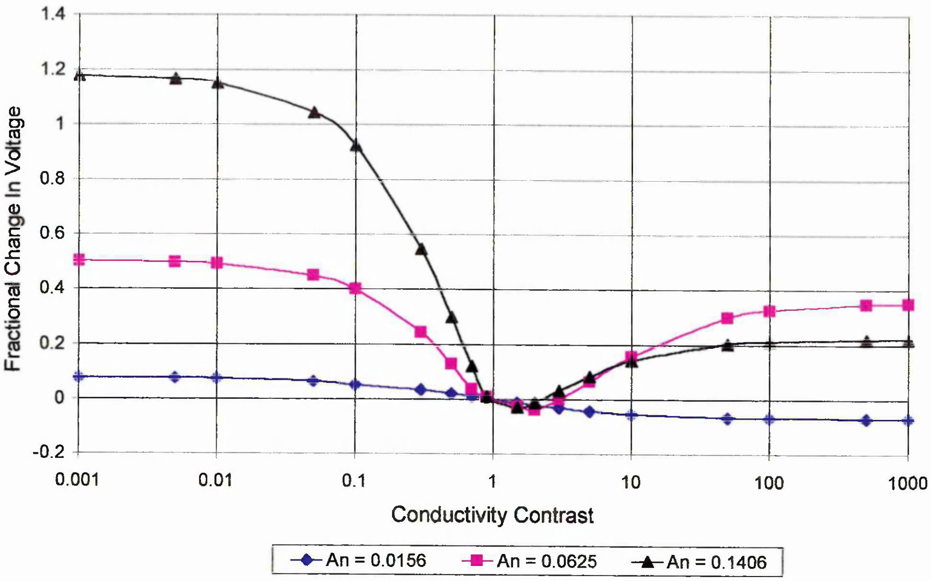


Figure 5.4.17 Fractional Change in Voltage against Conductivity Contrast for Centrally Placed Anomalies in Square Phantom with Distributed Electrodes (Phantom C)

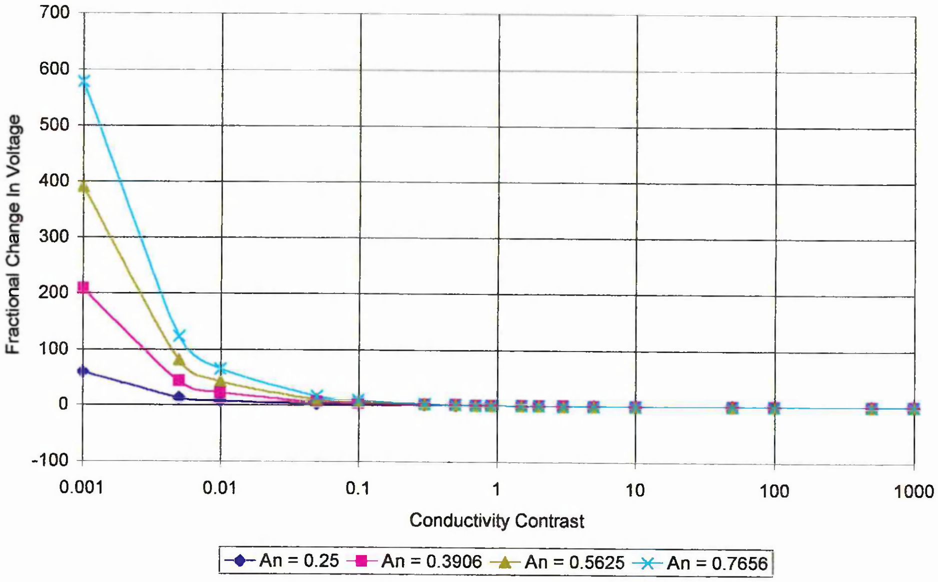


Figure 5.4.18 Fractional Change in Voltage against Conductivity Contrast for Centrally Placed Anomalies in Square Phantom with Distributed Electrodes (Phantom C)

From Figures 5.4.12, to 5.4.18 it is shown that the Square Phantom with distributed electrodes has higher values for both visibility and fractional change in voltage than for both the circular and square phantoms with peripheral electrodes which reaffirms the results of the experimental comparison. They also show that the square phantom with peripheral electrodes has higher values for visibility and fractional change in voltage than the circular phantom with peripheral electrodes. The figures also show that the difference between each phantom's visibility and fractional change in voltage for a given anomaly is higher with a conductivity contrast of less than 1 i.e. for anomalies with lower conductivity than that of the background. Figure 5.4.19 shows visibility for an anomaly with conductivity contrast of 0.001 with a normalised area of approximate 0.7 for each of the phantoms.

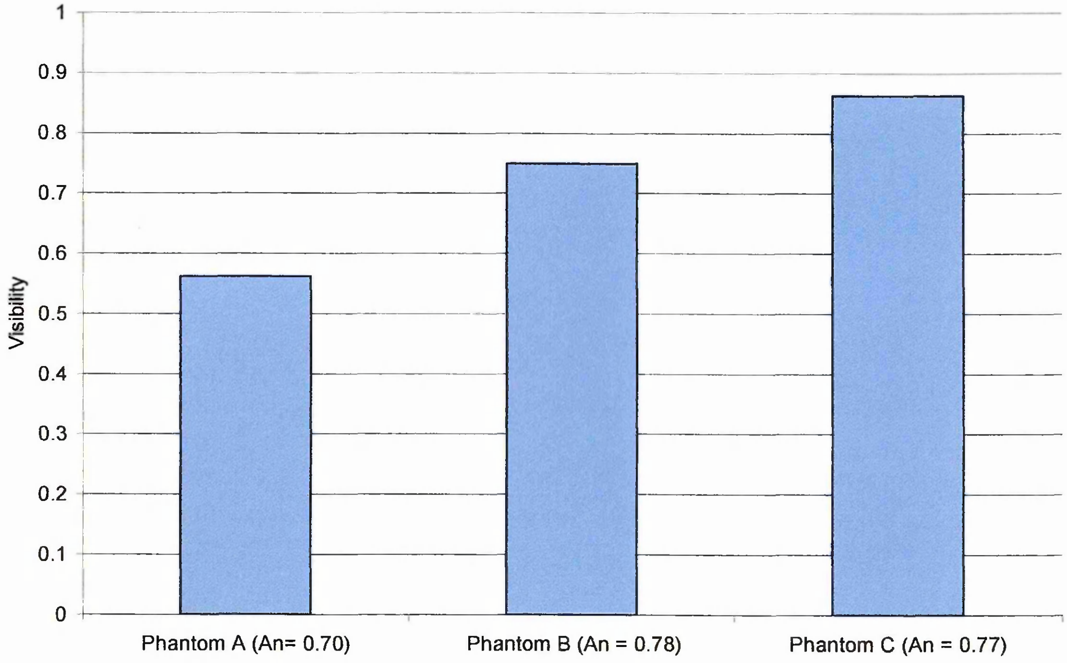


Figure 5.4.19 Visibility for each Phantom with Anomaly with Conductivity Contrast Of 0.001 and Normalised Area of Approximately 0.7 .

Figures 5.4.19 also reaffirms the finding of Basarab-Horwath I., Piotrowski J. and McEwan P. M., 1995, in that the visibility increases with anomaly area. Also from Figures 5.4.11, 5.4.13 and 5.4.15 it is clear, that visibility is always positive with conductivity contrasts less than 1 (i.e. for anomalies with lower conductivity than that of the background) and negative with conductivity contrast greater than 1 (i.e. for anomalies with higher conductivity than the background).

From both the experimental and simulated comparison of the three different electrode configurations, it is clear that the performance of the distributed arrangement is higher than both the peripheral electrode arrangements. The comparison also points out that the shape of the boundary can also effect the performances of an EIT system. This is evident from the higher performance of the square phantom with peripheral electrodes as compared to the circular phantom.

6. Novel Pressure Sensor Design and Results

6.1 Introduction

In this chapter the design and testing of an alternative pressure sensor to that outlined in Chapter 4.3 is presented and is divided into three main sections. Section 6.2 details the alternative pressure sensor design. Section 6.3 outlines the experimental procedure used to investigated the response of two different conductive mediums used for the sensor and section 6.4 presents the results and conclusions.

6.2 Alternative pressure sensor design

A side-view through the alternative pressure sensor design is shown in Figure 6.2.1(a). The sensor consists of a conductive medium covered with an insulating flexible membrane. The prototype has an active sensor area of 10 cm by 10 cm and two different conductive media have been investigated (see Section 6.3).

The distributed electrode arrangement was used for the sensor as shown in Figure 7.2.1(b) and this arrangement has been shown to perform significantly better than the normal peripheral arrangement (Section 5). The load was applied to the upper surface of the flexible membrane deforming the membrane; the underlying conductive solution is redistributed. The changes in this redistribution in conductivity due to the applied load are imaged using the EIT system.

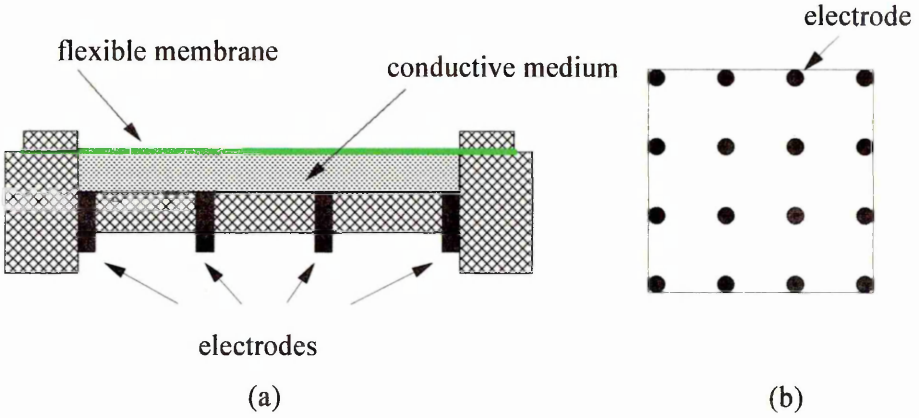


Figure 6.2.1 (a) Prototype pressure sensor, side view, and (b) Electrode arrangement, plan view

A further literature search after the sensor was designed found a similar sensor system (Helsel M, Zemel J.N. and Dominko V, 1988). The sensor was similar to that outlined above in that it consisted of a conductive solution covered in a flexible membrane. The system was described as a one-dimensional tactile sensor with an active sensor area of only 1cm by 1cm. Figure 6.2.2 shows the system developed by Helsel M, et al, 1988 and it can be seen that the electrodes cover the full width of the sensor. A signal of 300mV at 1kHz was applied to the two outermost electrodes and the voltage developed between the remaining electrodes measured. A plot of voltage against electrode number was used to characterise the performance of the system for a given load. They referred to their work as an impedance tomographic tactile sensor as the results they obtained suggested that the array could be used to monitor any impedance variation. The system described in this thesis is different to the work of Helsel M, et al, 1988 in that the sensor described here is two-dimensional. In addition, with appropriate reconstruction techniques the sensor could be further developed to give three-dimensional images of the deformation of the membrane and therefore the applied pressure. In contrast, the Helsel et al sensor is only one-dimensional and no method of reconstruction is shown.

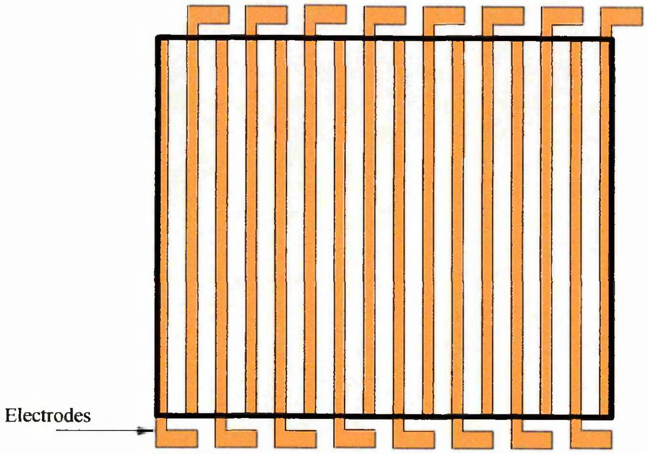


Figure 6.2.2 Plan View of Tactile Sensor Developed by Helsel et al, 1988

6.3 Experimental Procedure

The system used to test the response of the sensor for both the conductive media is shown in Figure 6.3.1. The area in contact with the sensor is varied between 4cm^2 and 25 cm^2 and the pressure being applied is varied by altering weights (0.1kg -1 kg) placed on the top of the test rig. This gives a range of pressures between 0Pa to 20kPa. Table 6.3.1 gives the applied pressure for a given area and weight.

Weight (grams)	Load Area $4cm^2$	Load Area $9cm^2$	Load Area $16cm^2$	Load Area $25cm^2$
100	2.45	1.09	0.61	Not used
200	4.91	2.18	1.23	0.78
300	7.36	3.27	1.84	Not used
400	9.81	4.36	2.45	1.57
500	12.26	5.45	Not used	Not used
600	14.72	6.54	3.68	2.35
700	17.17	7.63	Not used	Not used
800	19.62	8.72	4.91	3.14
900	Not used	9.81	Not used	Not used
1000	Not used	10.90	6.13	3.92

Table 6.3.1 Applied Pressure (kPa)For a Given Area and Weight

Note: The pressures in Table 6.3.1 listed as "Not used" were not used as the change in sensor deformation was not measurable with the Vernier scale.

The magnitude of the current used in the experiments was set at 2.5mA at 17.5kHz. This was the maximum signal magnitude usable as above this level the RMS to DC converter in the EIT system (Chapter 3) went into saturation. This saturation occurred when measuring between the electrode pairs closest to the source electrode pair for a given projection.

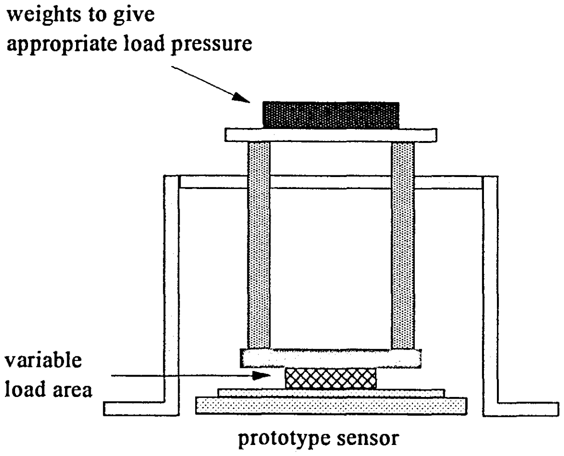


Figure 6.3.1 System for Testing the Alternative Pressure Sensor

A range of pressure form 0 Pa to 20 kPa was applied to the sensor, the actual maximum magnitude was limited by the experimental apparatus used. Three measures were used to characterise the performance of the two conductive media used. Two of the measures used are calculated from the differential voltage measurements and these are the average visibility equation 5.2.3 and the average fractional change in voltage equation 5.2.2. The third measure of performance is the amount of deformation of the flexible membrane, measured to ± 0.02mm using a Vernier scale. In order to obtain a better estimate of the sensor response the test was repeated five times and an average of the five tests was taken.

The system used in the previously described test procedure has some inaccuracies due to the apparatus used. The position of the load applied to the sensor has an inaccuracy of ±1%. This is due to the accuracy to which the load could be placed on the sensor (i.e. 1mm that, relative to the sensor width is ±1%). The application of the weight to alter the magnitude of the applied pressure caused the position of the load to move by ±1% (i.e. 1mm either vertically or horizontally). This produces a total accuracy in load position of ± 2%.

6.4 Results

As mentioned in section 6.2 two conductive media have been used for the sensor. The first medium consisted of medium density conductive foam (from RS components) saturated in saline solution with a conductivity of 125 mS. The second sensor medium consists of commercial grade ethylene glycol (anti-freeze brought from car spares outlet) diluted with saline solution to give a conductivity of 125 mS.

The response of the sensor utilising the saline saturated conductive foam is shown in Figures 6.4.1 to 6.4.3. The response shown is to an object having an area of 4cm^2 applied to the centre of the sensor, over a range of applied pressures (0 Pa to 14.72 kPa). The figures show the effect of increasing the pressure to a maximum and then reducing it back to 0 kPa with a one-minute time interval between each applied pressure reading. Each of the data points shown correspond to an applied pressure and the curves are fitted to these points using MATLAB's polyfit and polyval functions; a fourth order polynomial was used. As mentioned in Section 6.3 each of the tests were repeated five times and the average of these tests are shown. For clarity the standard deviations are not shown in Figures 6.4.1 to 6.4.3 but are given in Tables 6.4.1 and 6.4.2.

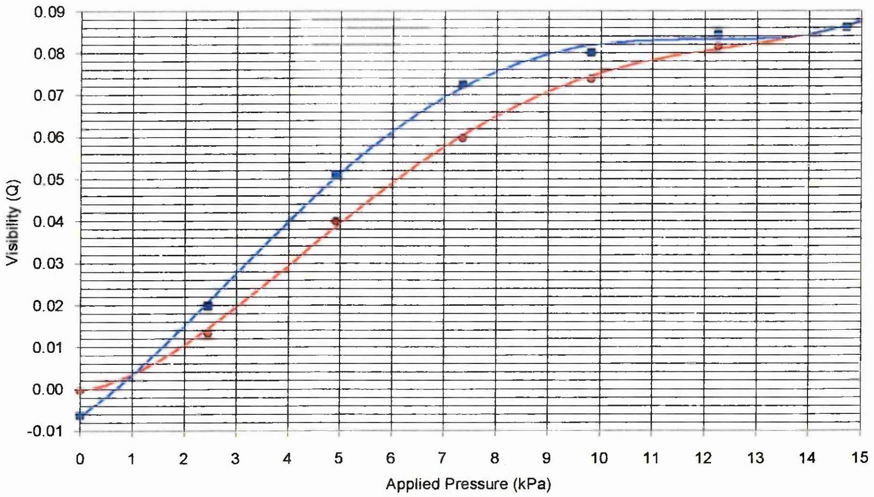


Figure 6.4.1 Visibility against Applied Pressure; 4cm^2 Object At Centre of Sensor ● = Pressure Increasing ■ = Pressure Decreasing

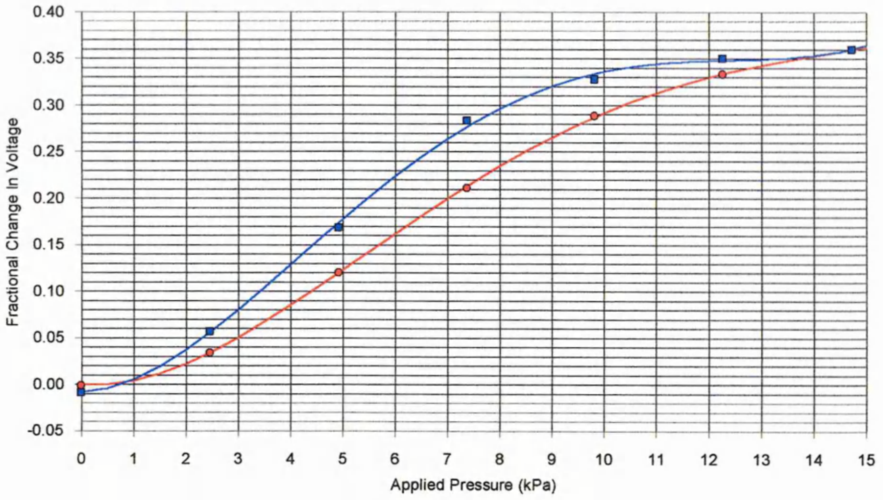


Figure 6.4.2 Fraction Change in Voltage against Applied Pressure: 4cm^2 Object at Centre of Sensor: ● = Pressure Increasing, ■ = Pressure Decreasing

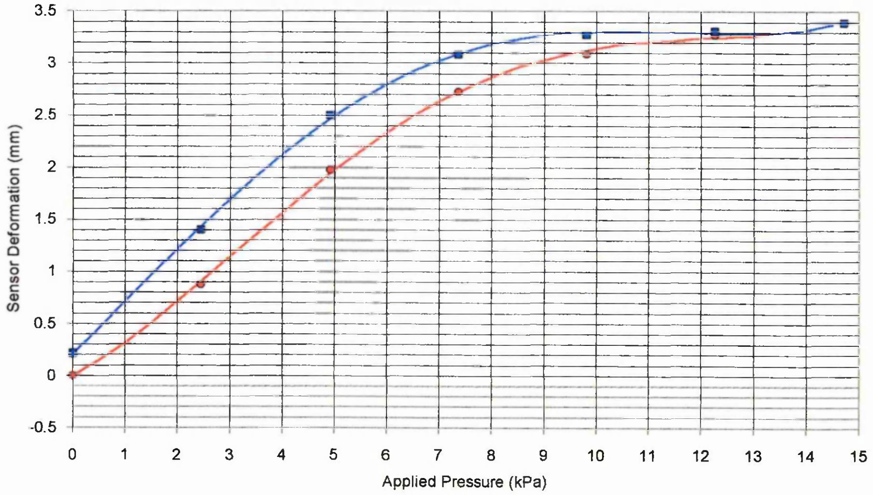


Figure 6.4.3 Sensor Deformation against Applied Pressure: 4 cm^2 Object at Centre of Sensor: ● = Pressure Increasing ■ = Pressure Decreasing

Pressure kPa	Average Visibility	Visibility Standard Deviation	Average δV/V	δV/V Standard Deviation
0.00	-0.0004	0.0006	-0.0008	0.0011
2.45	0.0132	0.0054	0.0335	0.0140
4.91	0.0399	0.0098	0.1199	0.0340
7.36	0.0597	0.0097	0.2107	0.0392
9.81	0.0738	0.0125	0.2884	0.0496
12.26	0.0815	0.0138	0.3322	0.0543
14.72	0.0860	0.0136	0.3595	0.0588
12.26	0.0843	0.0137	0.3500	0.0565
9.81	0.0800	0.0137	0.3278	0.0564
7.36	0.0724	0.0141	0.2831	0.0486
4.91	0.0510	0.0135	0.1682	0.0408
2.45	0.0197	0.0135	0.0569	0.0329
0.00	-0.0064	0.0126	-0.0090	0.0195

Table 6.4.1 Average and Standard Deviation against Applied Pressure for Visibility and Fractional Change In Voltage; 4 cm^2 Object at Centre of Sensor

Pressure kPa	Average Sensor Deformation (mm)	Sensor Deformation Standard Deviation (mm)
0.00	0.00	0.00
2.45	0.87	0.06
4.91	1.97	0.12
7.36	2.72	0.11
9.81	3.09	0.12
12.26	3.27	0.13
14.72	3.39	0.13
12.26	3.31	0.13
9.81	3.27	0.16
7.36	3.08	0.16
4.91	2.50	0.17
2.45	1.40	0.11
0.00	0.21	0.06

Table 6.4.2 Average and Standard Deviation against Applied Pressure for Sensor Deformation; 4 cm^2 Object at Centre of Sensor

From Figures 6.4.1 to 6.4.3, it can be seen that the sensor has a significant amount of hysteresis over the range of input values. From the standard deviations given in Tables 6.4.1 and 6.4.2 it can be seen that the variations in the measures over the five samples used is also significant and well above the inaccuracies in the test procedure used (see Section 6.3). The hysteresis is almost certainly due to the elastic property of the sensor medium and in particular the conductive foam used. This is apparent from Figure 6.4.3 where it is clearly seen that the sensor deformation against applied pressure does not return to zero with the applied pressure removed. The medium can therefore be said to have 'memory' of its pervious load i.e. it is not totally elastic. The percentage hysteresis for each of the measures used at each pressure are shown in Table 6.4.3.

Pressure (kPa)	**Depth Of Sensor Deformation % Hysteresis**	**Visibility (Q) % Hysteresis**	**Fractional Change In Voltage % Hysteresis**
2.45	60.91	49.24	69.85
4.91	26.90	27.81	40.28
7.37	13.24	21.27	34.36
9.81	5.86	8.40	13.66
12.26	1.22	3.44	5.36
14.72	0.00	0.00	0.00

Table 6.4.3 Percentage Hysteresis in Sensor Deformation, Visibility and Fractional Change in Voltage against Applied Pressure; 4cm^2 Object at Centre of Sensor

From Table 6.4.3, it can be seen that the amount of hysteresis in the sensor deformation due the medium not being totally elastic is between 0.00% and 60.91%. However, the visibility and fractional change in voltage have a hysteresis of between 0.00% to 49.24% and 0.00% to 69.85% respectively. The differences between the hysteresis due to the mediums 'memory' property and the hysteresis in the visibility and fractional change in voltage are almost certainly due to small air bubbles within the sensor medium. These air bubbles are redistributed within the medium as the pressure is applied and removed. The conductivity distribution once the load is removed is therefore not the same as before the pressure was applied. By physical inspection of the sensor, the air bubbles can be clearly seen within the sensor medium. From the above tests, it can be seen that due to both the hysteresis and variation in the measures used the conductive foam and saline solution medium is unsuitable as a conductive medium for the sensor. Therefore, no further tests were performed on the conductive foam and saline solution medium.

The response of the sensor utilising the ethylene glycol and saline conductive medium is shown in Figures 6.4.4 to 6.4.6. The response shown is to an object having an area of 4cm^2 applied at the centre of the sensor, over a range of applied pressures (0 kPa to 19.62 kPa). The figures show the effect of increasing the pressure to a maximum and then reducing it back to 0 kPa with a one-minute time period between each applied pressure reading. The upper limit of the applied pressures is higher than with the saline saturated conductive foam medium. The higher limit in applied pressure is due to the absence of the conductive foam. The presence of the conductive foam gives a lower limit, as the foam will only deform to a minimum thickness. Whereas with the absence of the foam the sensor can deform until the flexible membrane touches the base of the sensor. From Figure 6.4.3, it can be seen that the amount of deformation in the sensor element for the first conductive medium levels off above 10kPa. With the ethylene glycol and saline conductive medium, this is not the case, which is seen in Figure 6.4.6. As with the saline saturated conductive foam medium MATLAB's polyfit and polyval are used to fit curves to the data points shown. Again, for clarity the standard deviations are not shown in Figures 6.4.4 to 6.4.6 but are shown in Tables 6.4.4 and 6.4.5.

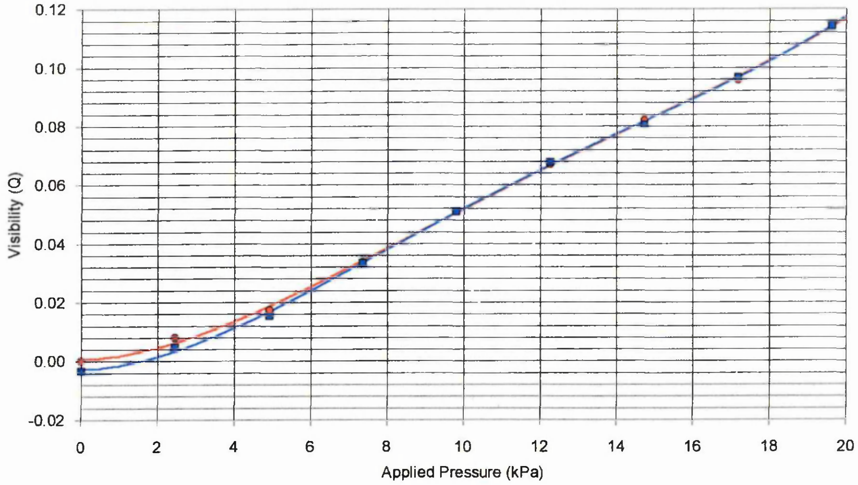


Figure 6.4.4 Visibility against applied pressure: 4cm^2 object at centre of sensor: ● = pressure increasing ■ = pressure decreasing

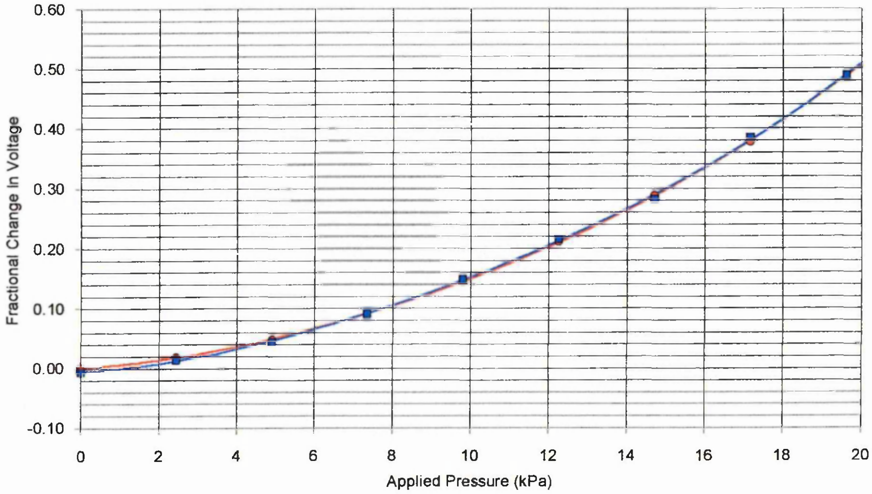


Figure 6.4.5 Fraction Change in Voltage against Applied Pressure: 4cm^2 Object at Centre of Sensor: ● = Pressure Increasing, ■ = Pressure Decreasing

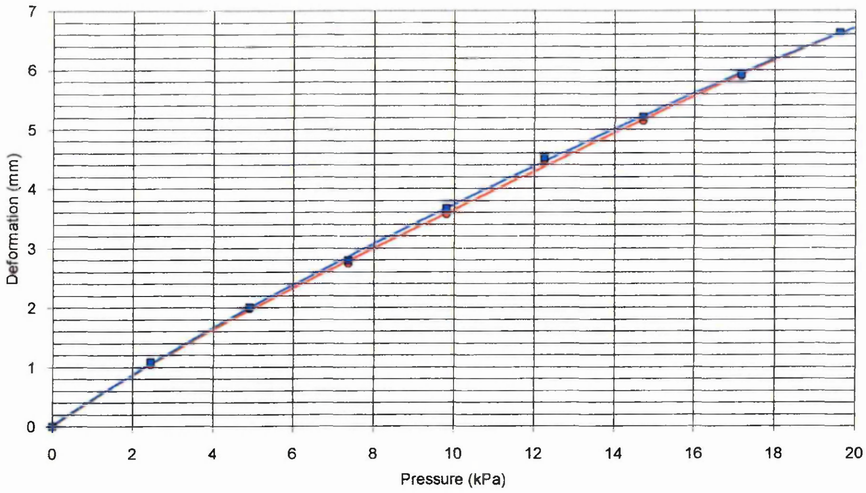


Figure 6.4.6 Sensor Deformation against Applied Pressure: 4cm^2 Object at Centre of Sensor: ● = Pressure Increasing. ■ = Pressure Decreasing

Pressure kPa	Average Visibility	Visibility Standard Deviation	Average δV/V	δV/V Standard Deviation
0.00	0.0000	0.0000	0.0000	0.0000
2.45	0.0078	0.0008	0.0193	0.0025
4.91	0.0173	0.0007	0.0471	0.0015
7.36	0.0335	0.0014	0.0896	0.0039
9.81	0.0509	0.0014	0.1444	0.0046
12.26	0.0672	0.0016	0.2104	0.0063
14.72	0.0819	0.0006	0.2870	0.0027
17.17	0.0958	0.0007	0.3776	0.0052
19.62	0.1142	0.0017	0.4879	0.0104
17.17	0.0967	0.0016	0.3845	0.0086
14.72	0.0805	0.0015	0.2816	0.0100
12.26	0.0677	0.0007	0.2151	0.0040
9.81	0.0509	0.0008	0.1482	0.0025
7.36	0.0333	0.0009	0.0904	0.0021
4.91	0.0153	0.0008	0.0434	0.0016
2.45	0.0048	0.0006	0.0135	0.0012
0.00	-0.0033	0.0007	-0.0066	0.0014

Table 6.4.4 Average and Standard Deviation against Applied Pressure for Visibility and fractional Change in Voltage: 4cm^2 Object at Centre of Sensor

Pressure kPa	Average Sensor Deformation (mm)	Sensor Deformation Standard Deviation (mm)
0.00	0.00	0.00
2.45	1.04	0.08
4.91	1.98	0.10
7.36	2.74	0.07
9.81	3.57	0.05
12.26	4.43	0.08
14.72	5.14	0.04
17.17	5.90	0.05
19.62	6.62	0.08
17.17	5.92	0.06
14.72	5.20	0.05
12.26	4.52	0.07
9.81	3.67	0.03
7.36	2.80	0.06
4.91	2.00	0.09
2.45	1.08	0.08
0.00	0.00	0.00

Table 6.4.5 Average and Standard Deviation against
Applied Pressure for Sensor Deformation; 4cm^2 Object at Centre of Sensor

From Figures 6.4.4 to 6.4.6, it can be seen that the sensor constructed with the second conductive medium does not suffer from the same amount of hysteresis as the first conductive medium. Also, from Tables 6.4.4 and 6.4.5 it can be seen that the variation in the measures used over the five samples is at least an order of magnitude less than with the first medium. In Figure 6.4.6 it can be clearly seen that the deformation of the sensor element returns to zero once the load is removed. The small percentage of hysteresis for each of the measures used is shown in Table 6.4.6 for each of the pressure applied.

Pressure kPa	Depth Of Sensor Deformation % Hysteresis	Visibility (Q) % Hysteresis	Fractional Change In Voltage % Hysteresis
2.45	3.85	-38.46	-30.05
4.91	1.01	-11.56	-7.86
7.36	2.19	-0.60	0.89
9.81	2.80	0.00	2.63
12.26	2.03	0.74	2.23
14.72	1.17	-1.71	-1.88
17.17	0.34	0.94	1.83
19.62	0.00	0.00	0.00

Table 6.4.6 Percentage Hysteresis in Sensor Deformation, Visibility and Fractional Change in Voltage against Applied Pressure; 4cm2 Object at Centre of Sensor

From Table 6.4.6, it can be seen that the hysteresis in the depth of sensor deformation is within the accuracy of the test procedure used. It can also be seen that the hysteresis in visibility and fraction change in voltage alters in sign over the pressure range. This change in sign means that the response of both the visibility and fractional change in voltage to the application and removal of the load does not follow the standard hysteresis curve. Therefore, the hysteresis given in Table 6.4.6 for visibility and fraction change in voltage should be simply quoted as a variation. However, it can be seen that at applied pressures of 4.91 kPa and below the variation in visibility and fractional change in voltage become more than expected, due to the accuracy of the experimental procedure used (see Section 6.3). As there is negligible hysteresis in sensor deformation the variations in the measures used at the lower pressures is almost certainly due the conductive solution not being totally homogeneous. As it is only at the lower applied pressures that these variations are observed the sensor using the ethylene glycol and saline conductive medium was tested further.

Figures 6.4.7 to 6.4.9 show the response of the sensor to a range of pressures from 0 kPa to 10.90 kPa applied to an area of 9 cm^2 at the centre of the sensor.

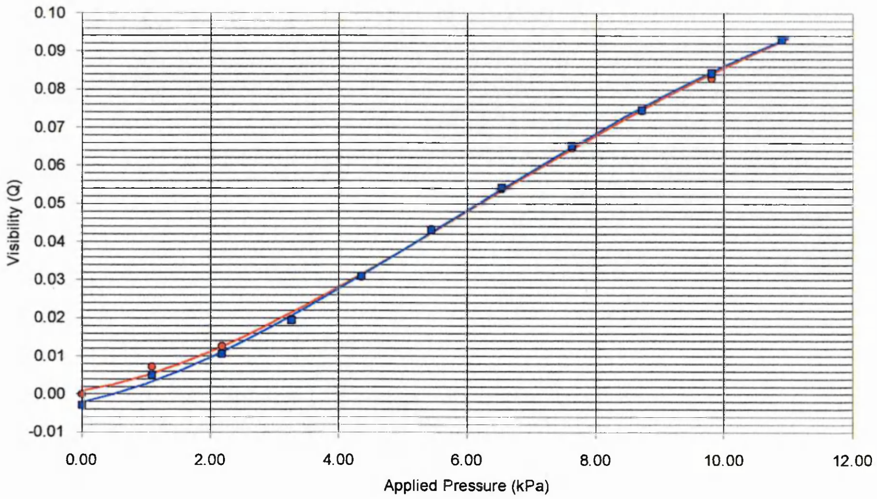


Figure 6.4.7 Visibility against Applied Pressure; 9 cm^2 Object at Centre of Sensor; ● = Pressure Increasing ■ = Pressure Decreasing

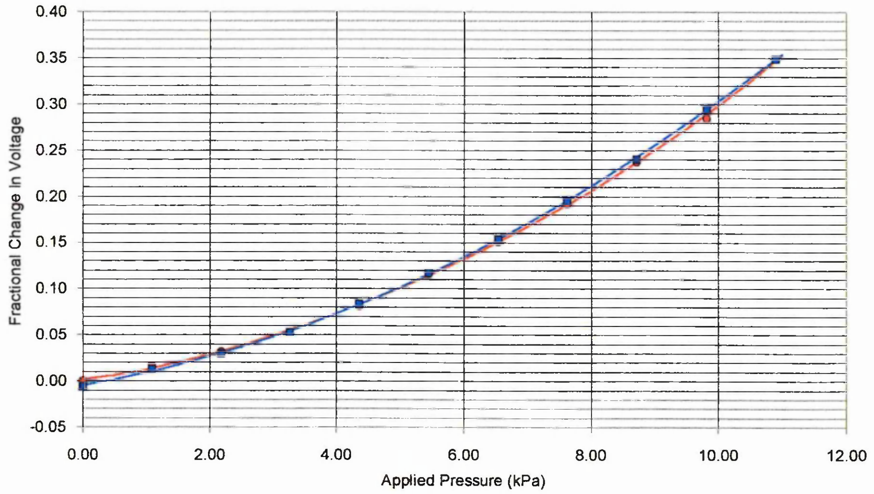


Figure 6.4.8 Fraction Change in Voltage against Applied Pressure; 9 cm^2 Object at Centre of Sensor; ● = Pressure Increasing, ■ = Pressure Decreasing

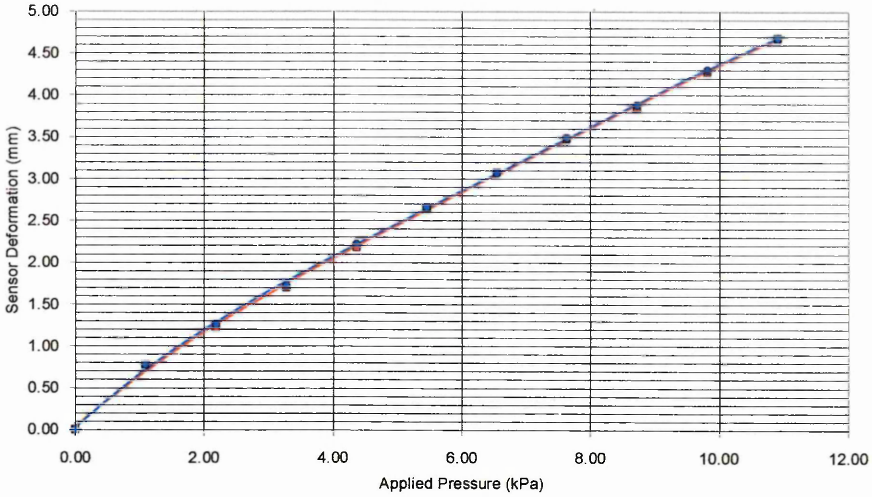


Figure 6.4.9 Sensor Deformation against Applied Pressure: 9cm^2 Object at Centre of Sensor; ● = Pressure Increasing, ■ = Pressure Decreasing

Tables 6.4.7 and 6.4.8 show the standard deviation for the data points shown in Figures 6.4.7 to 6.4.9. Table 6.4.9 shows the hysteresis in the measures due to the application and then removal of the applied load for each of the applied pressures.

Pressure kPa	**Average Visibility**	**Visibility Standard Deviation**	**Average δV/V**	**δV/V Standard Deviation**
0.00	0.0000	0.0000	0.0000	0.0000
1.09	0.0071	0.0006	0.0160	0.0015
2.18	0.0125	0.0011	0.0317	0.0030
3.27	0.0195	0.0014	0.0524	0.0038
4.36	0.0308	0.0019	0.0809	0.0044
5.45	0.0432	0.0018	0.1160	0.0056
6.54	0.0538	0.0022	0.1504	0.0073
7.63	0.0646	0.0025	0.1917	0.0094
8.72	0.0744	0.0032	0.2372	0.0138
9.81	0.0829	0.0031	0.2850	0.0140
10.90	0.0930	0.0034	0.3483	0.0173
9.81	0.0842	0.0037	0.2944	0.0176
8.72	0.0746	0.0035	0.2401	0.0135
7.63	0.0651	0.0037	0.1958	0.0128
6.54	0.0541	0.0039	0.1533	0.0117
5.45	0.0430	0.0038	0.1167	0.0100
4.36	0.0309	0.0042	0.0825	0.0103
3.27	0.0192	0.0044	0.0524	0.0101
2.18	0.0105	0.0040	0.0289	0.0086
1.09	0.0049	0.0038	0.0125	0.0075
0.00	-0.0030	0.0041	-0.0060	0.0081

Table 6.4.7 Average and Standard Deviation against Applied Pressure for Visibility and Fractional Change in Voltage; 9 cm^2 Object at Centre of Sensor

Pressure kPa	Average Sensor Deformation	Sensor Deformation Standard Deviation
0.00	0.00	0.0000
1.09	0.78	0.0261
2.18	1.22	0.0141
3.27	1.70	0.0089
4.36	2.18	0.0374
5.45	2.64	0.0219
6.54	3.08	0.0200
7.63	3.48	0.0200
8.72	3.85	0.0268
9.81	4.28	0.0200
10.90	4.68	0.0456
9.81	4.30	0.0600
8.72	3.88	0.0261
7.63	3.46	0.0415
6.54	3.07	0.0268
5.45	2.66	0.0316
4.36	2.22	0.0639
3.27	1.73	0.0390
2.18	1.26	0.0219
1.09	0.78	0.0167
0.00	0.00	0.0000

Table 6.4.8 Average and Standard Deviation against Applied Pressure for Sensor Deformation; 9 cm^2 Object at Centre of Sensor

Pressure kPa	Depth Of Sensor Deformation % Hysteresis	Visibility (Q) % Hysteresis	Fractional Change In Voltage % Hysteresis
1.09	0.00	-30.99	-21.88
2.18	3.28	-16.00	-8.83
3.27	1.76	-1.54	0.00
4.36	1.83	0.32	1.98
5.45	0.76	-0.46	0.60
6.54	-0.32	0.56	1.93
7.63	0.29	0.77	2.14
8.72	0.78	0.27	1.22
9.81	0.47	1.57	3.30
10.90	0.00	0.00	0.00

Table 6.4.9 Percentage Hysteresis in Sensor Deformation, Visibility and Fractional Change in Voltage against Applied Pressure; 9 cm^2 Object at Centre of Sensor

Figures 6.4.10 to 6.4.12 show the response of the sensor to a range of pressures from 0 kPa to 6.13 kPa applied to an area of 16 cm^2 at the centre of the sensor

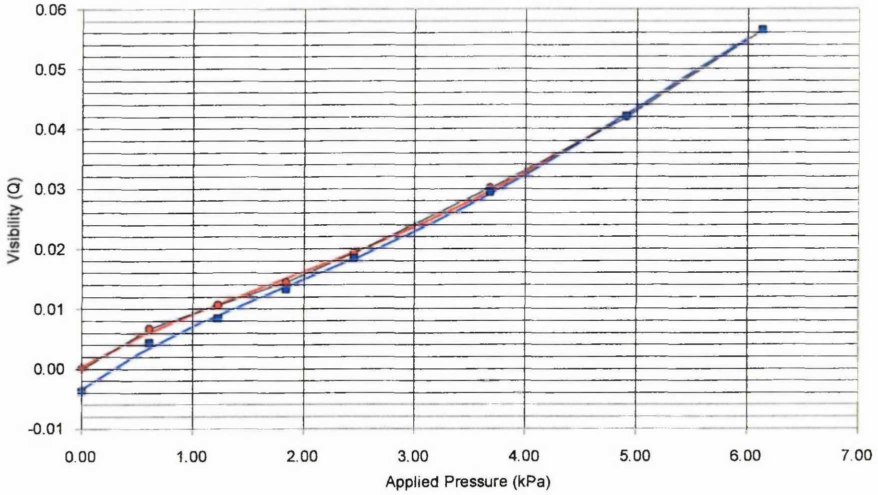


Figure 6.4.10 Visibility against Applied Pressure; 16 cm^2 Object at Centre of Sensor; ● = Pressure Increasing ■ = Pressure Decreasing

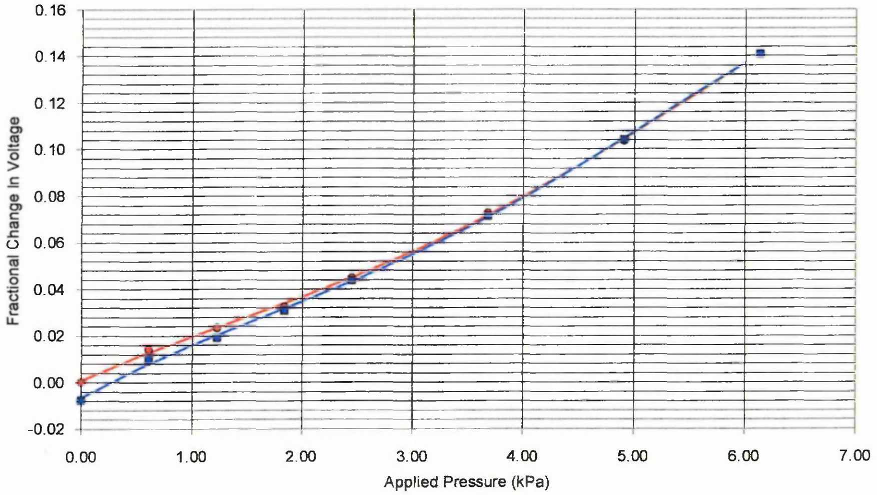


Figure 6.4.11 Fraction Change in Voltage against Applied Pressure; 16 cm^2 Object at Centre of Sensor; ● = Pressure Increasing, ■ = Pressure Decreasing

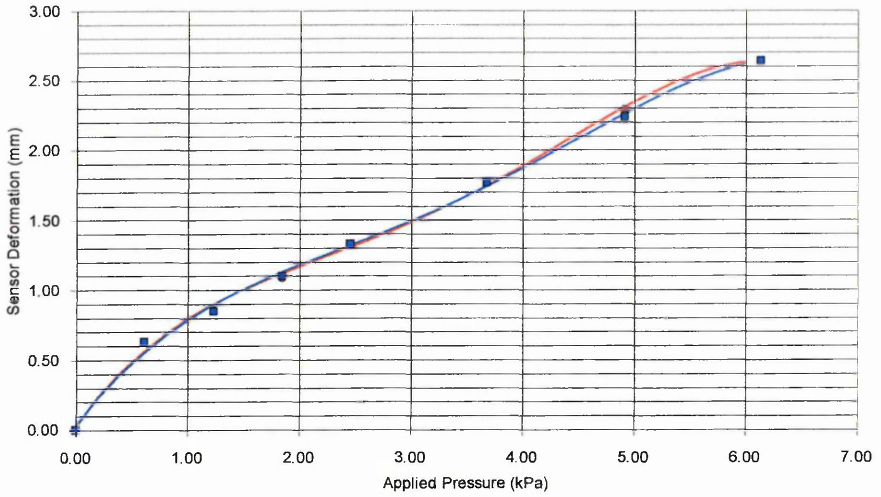


Figure 6.4.12 Sensor Deformation Against Applied Pressure: 16 cm^2 Object at Centre of Sensor; ● = Pressure Increasing, ■ = Pressure Decreasing

Tables 6.4.10 and 6.4.11 show the standard deviation for the data points shown in Figures 6.10 to 6.12. In addition, Table 6.4.12 shows the hysteresis in the measures due to the application and then removal of the applied load for each of the applied pressure.

Pressure kPa	Average Visibility	Visibility Standard Deviation	Average δV/V	δV/V Standard Deviation
0.00	0.0000	0.0000	0.0000	0.0000
0.61	0.0067	0.0003	0.0140	0.0006
1.23	0.0107	0.0006	0.0233	0.0015
1.84	0.0144	0.0011	0.0323	0.0027
2.45	0.0193	0.0006	0.0448	0.0018
3.68	0.0301	0.0002	0.0726	0.0012
4.91	0.0420	0.0010	0.1035	0.0027
6.13	0.0564	0.0012	0.1407	0.0033
4.91	0.0421	0.0004	0.1041	0.0016
3.68	0.0294	0.0008	0.0714	0.0022
2.45	0.0184	0.0005	0.0436	0.0011
1.84	0.0132	0.0011	0.0306	0.0021
1.23	0.0084	0.0012	0.0192	0.0023
0.61	0.0043	0.0013	0.0097	0.0027
0.00	-0.0038	0.0014	-0.0075	0.0027

Table 6.4.10 Average and Standard Deviation against Applied Pressure for Visibility and fractional Change in Voltage; 16 cm^2 Object at Centre of Sensor

Pressure kPa	Average Sensor Deformation	Sensor Deformation Standard Deviation
0.00	0.00	0.0000
0.61	0.63	0.0303
1.23	0.86	0.0498
1.84	1.09	0.0303
2.45	1.32	0.0385
3.68	1.77	0.0502
4.91	2.29	0.1026
6.13	2.64	0.0358
4.91	2.24	0.0583
3.68	1.77	0.0438
2.45	1.33	0.0415
1.84	1.10	0.0245
1.23	0.85	0.0522
0.61	0.63	0.0460
0.00	0.00	0.0000

Table 6.4.11 Average and Standard Deviation against Applied Pressure for Sensor Deformation; 16 cm^2 Object at Centre of Sensor

Pressure kPa	Depth Of Sensor Deformation % Hysteresis	Visibility (Q) % Hysteresis	Fractional Change In Voltage % Hysteresis
0.61	0.00	-35.82	-30.71
1.23	-1.16	-21.50	-17.60
1.84	0.92	-8.33	-5.26
2.45	0.76	-4.66	-2.68
3.68	0.00	-2.33	-1.65
4.91	-2.18	0.24	0.58
6.13	0.00	0.00	0.00

Table 6.4.12 Percentage Hysteresis in Sensor Deformation, Visibility and Fractional Change in Voltage against Applied Pressure; 16 cm^2 Object at Centre of Sensor

Figures 6.4.13 to 6.4.15 show the response of the sensor to a range of pressures from 0 kPa to 3.92 kPa applied to an area of 25 cm^2 at the centre of the sensor.

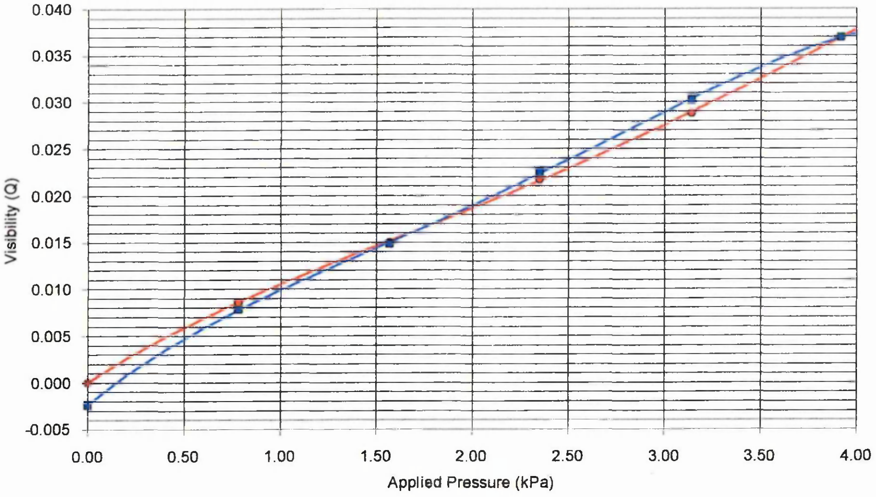


Figure 6.4.13 Visibility against Applied Pressure: 25 cm^2 Object at Centre of Sensor: ● = Pressure Increasing. ■ = Pressure Decreasing

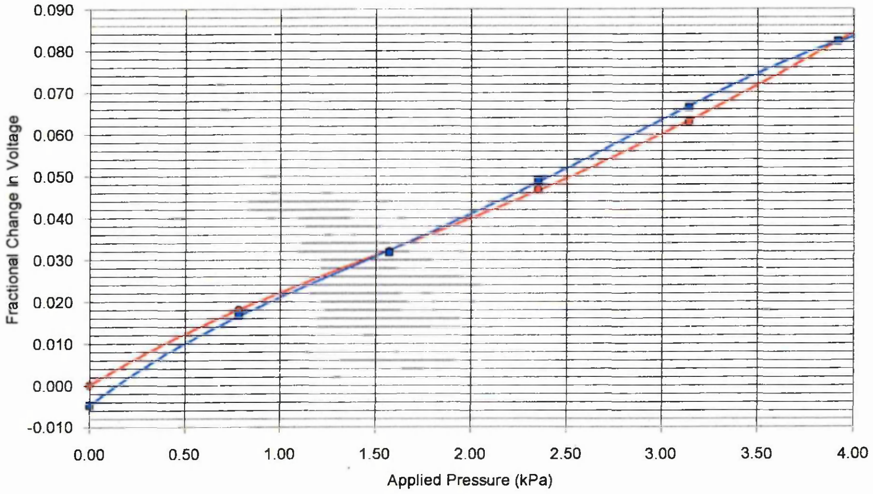


Figure 6.4.14 Fraction Change in Voltage against Applied Pressure: 25 cm^2 Object at Centre of Sensor: ● = Pressure Increasing. ■ = Pressure Decreasing

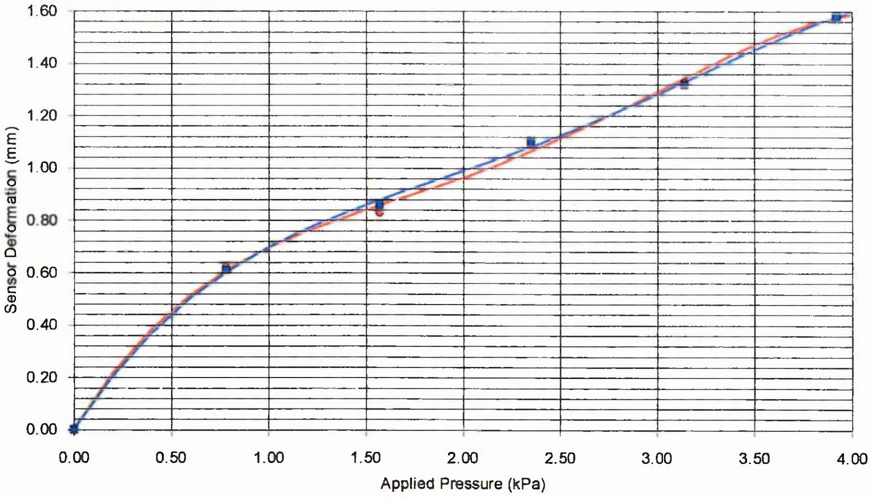


Figure 6.4.15 Sensor Deformation against Applied Pressure: 25 cm^2 Object at Centre of Sensor: ● = Pressure Increasing, ■ = Pressure Decreasing

Tables 6.4.13 and 6.4.14 show the standard deviation for the data points shown in Figures 6.4.13 to 6.4.15. In addition, Table 6.4.15 shows the hysteresis in the measures due to the application and then removal of the applied load for each of the applied pressures.

Pressure kPa	Average Visibility	Visibility Standard Deviation	Average δV/V	δV/V Standard Deviation
0.00	0.0000	0.0000	0.0000	0.0000
0.78	0.0086	0.0004	0.0180	0.0009
1.57	0.0150	0.0005	0.0319	0.0010
2.35	0.0217	0.0008	0.0468	0.0017
3.14	0.0288	0.0013	0.0630	0.0028
3.92	0.0369	0.0015	0.0822	0.0036
3.14	0.0302	0.0014	0.0664	0.0033
2.35	0.0225	0.0013	0.0489	0.0030
1.57	0.0148	0.0012	0.0317	0.0026
0.78	0.0078	0.0012	0.0166	0.0025
0.00	-0.0024	0.0011	-0.0048	0.0022

Table 6.4.13 Average and Standard Deviation against Applied Pressure for Visibility and fractional Change in Voltage: 25cm^2 Object at Centre of Sensor

Pressure kPa	Average Sensor Deformation	Sensor Deformation Standard Deviation
0.00	0.00	0.0000
0.78	0.62	0.0167
1.57	0.83	0.0268
2.35	1.09	0.303
3.14	1.33	0.0303
3.92	1.58	0.0283
3.14	1.32	0.0245
2.35	1.10	0.0358
1.57	0.86	0.0434
0.78	0.61	0.0179
0.00	0.00	0.0000

Table 6.4.14 Average and Standard Deviation against Applied Pressure for Sensor Deformation; 25 cm^2 Object at Centre of Sensor

Pressure kPa	Depth Of Sensor Deformation % Hysteresis	Visibility (Q) % Hysteresis	Fractional Change In Voltage % Hysteresis
0.78	-1.64	-9.30	-7.78
1.57	3.61	-1.33	-0.63
2.35	0.92	3.69	4.49
3.14	-0.75	4.86	5.40
3.92	0.00	0.00	0.00

Table 6.4.15 Percentage Hysteresis in Sensor Deformation, Visibility and Fractional Change in Voltage against Applied Pressure; 25 cm^2 Object at Centre of Sensor

The sensitivity coefficient reconstruction algorithm described in Chapter 3.4 was used to reconstruct the images shown in Figures 6.4.16 to 6.4.19. The images show the changes in impedance for the sensor when a range of pressure from 0.61kPa to 19.62kPa are applied to 4cm^2, 9cm^2 16cm^2 and 25cm^2 square objects placed at the centre of the sensors. The images clearly show that the impedance of the sensor changes with applied pressure. The images however do have a number of artefacts; firstly there are peaks at the corners of the anomaly shown at the centre of each image. These peaks coincide with the position of the centrally placed electrodes and are probably due to larger distortions in the equipotentials around the electrodes. The other artefacts seen in the images are toward the edge of the sensor and could be due to the redistribution of the conductive medium from the centre to the edge again causing large changes in the equipotential fields at the electrodes on the periphery of the sensor.

Figures 6.4.20 to 6.4.23 show the maximum pixel value (ignoring the artefacts mentioned above) against pressure for reconstructed images for each applied pressure. It is observed from these that for the 4cm^2 and 9cm^2 objects the response is linear for the range of applied pressure. However for the 16cm^2 and 25cm^2 objects the response looses its linearity for pressures above 1.84kPa and 3.14kPa respectively.

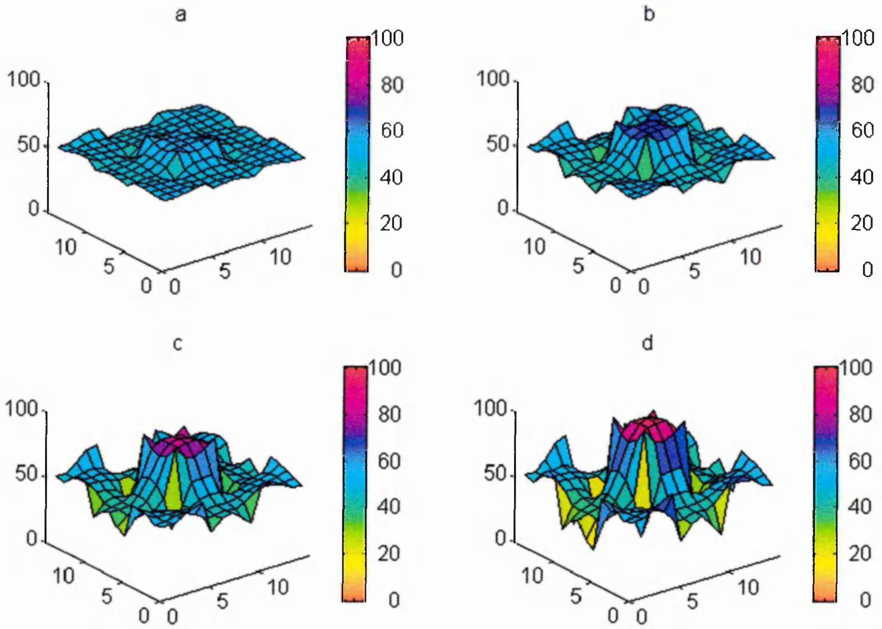


Figure 6.4.16 Reconstructed Images For 4cm2 Object At Centre Of Sensor
(a) Pressure = 4.91kPa, (b) Pressure = 9.81kPa, (c) Pressure = 14.72kPa,
(d) Pressure = 19.62 kPa.

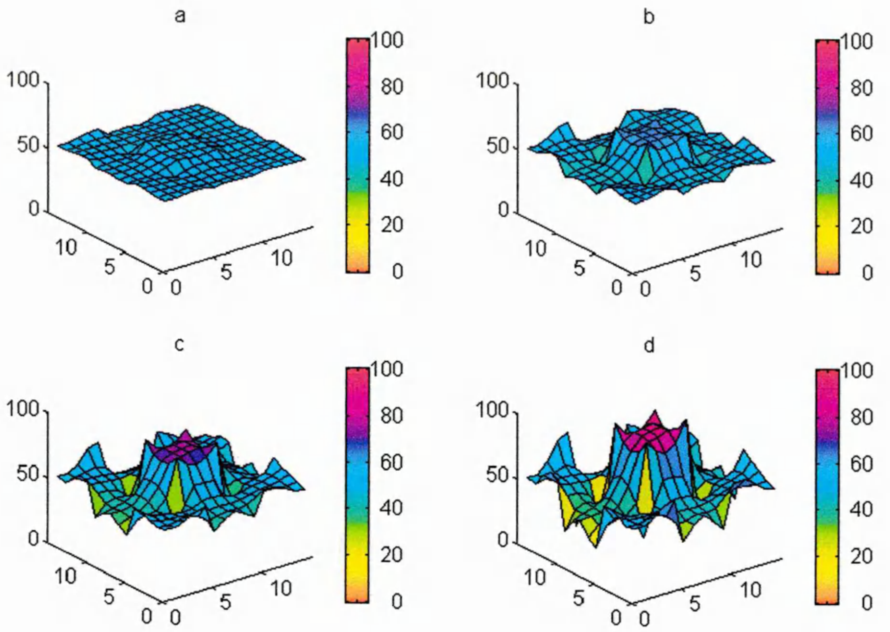


Figure 6.4.17 Reconstructed Images For 9 cm^2 Object at Centre of Sensor (a) Pressure = 1.09kPa, (b) Pressure = 4.36kPa, (c) Pressure = 7.63kPa, (d) Pressure = 10.90 kPa.

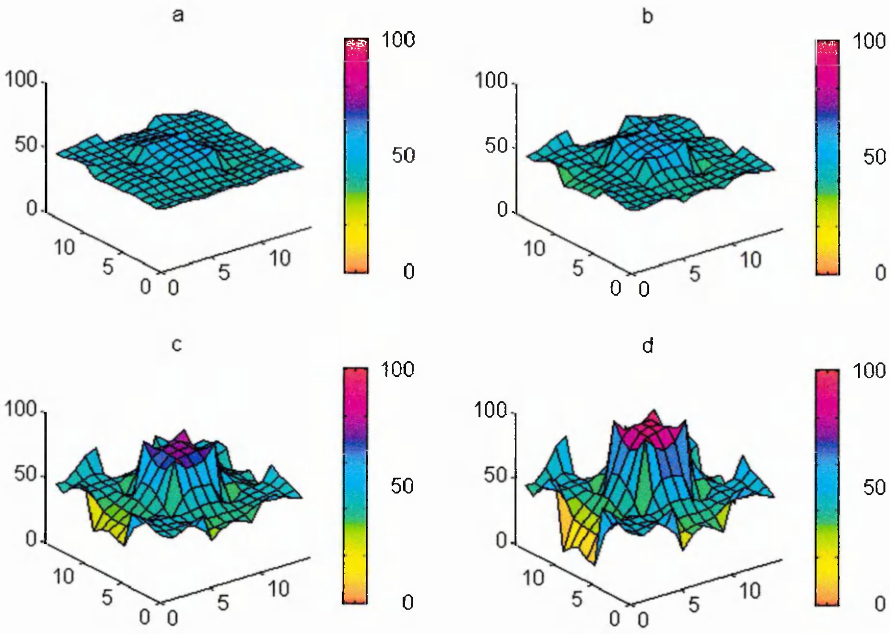


Figure 6.4.18 Reconstructed Images For 16 cm2 Object at Centre of Sensor (b) Pressure = 0.61 kPa, (b) Pressure = 1.84 kPa, (c) Pressure = 3.68 kPa, (c) Pressure = 6.13 kPa.

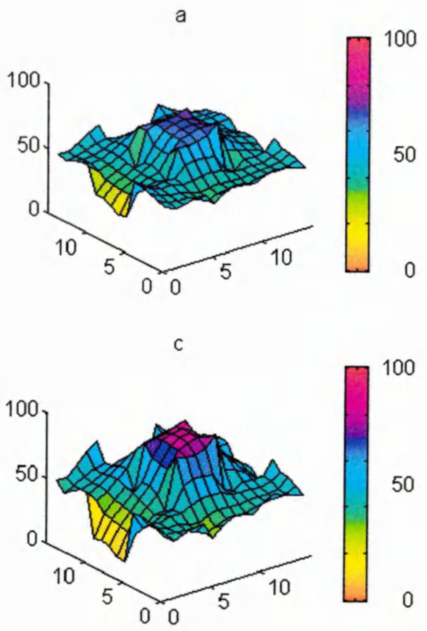


Figure 6.4.19 Reconstructed Images For 25 cm^2 Object At Centre Of Sensor (d) Pressure = 1.57 kPa, (b) Pressure = 2.35 kPa, (c) Pressure = 3.14 kPa, (e) Pressure = 3.92 kPa.

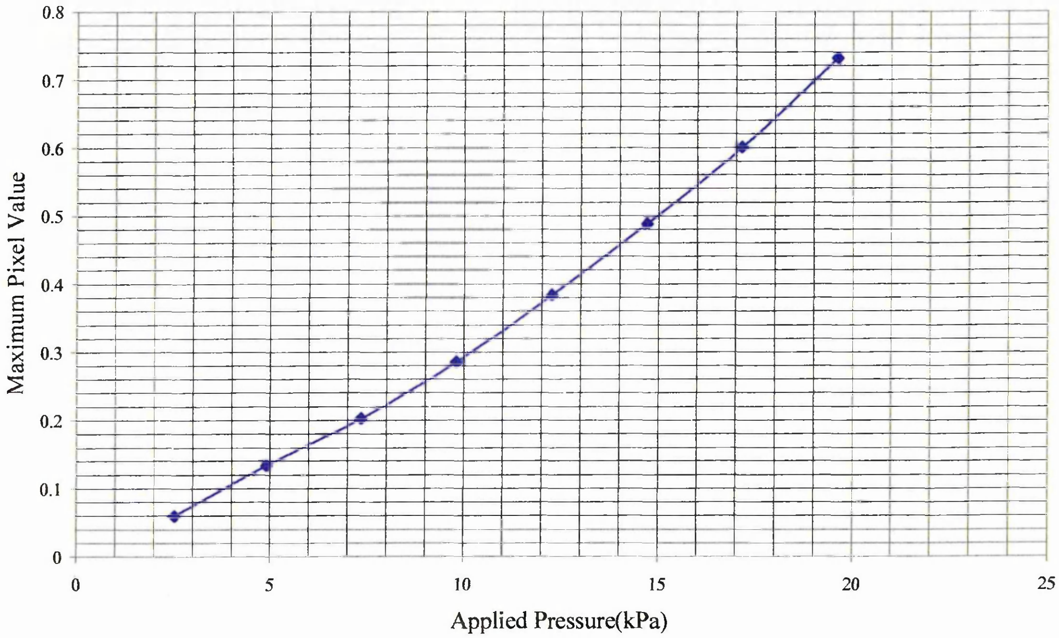


Figure 6.4.20 Maximum Pixel Value against Applied Pressure 4 cm^2, Object at Centre of Sensor

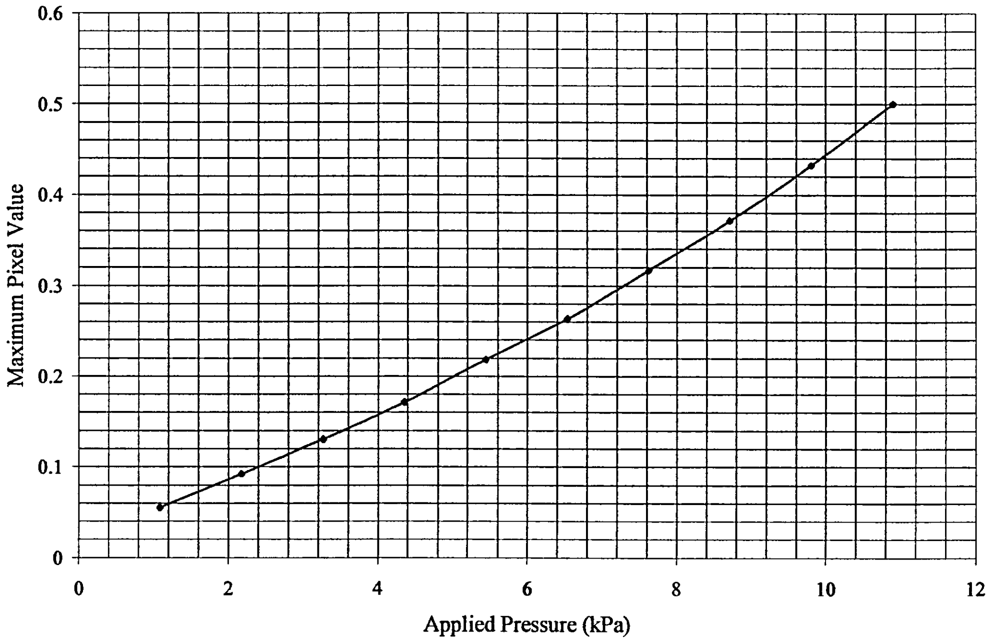


Figure 6.4.21 Maximum Pixel Value against Applied Pressure 9 cm^2 Object at Centre of Sensor

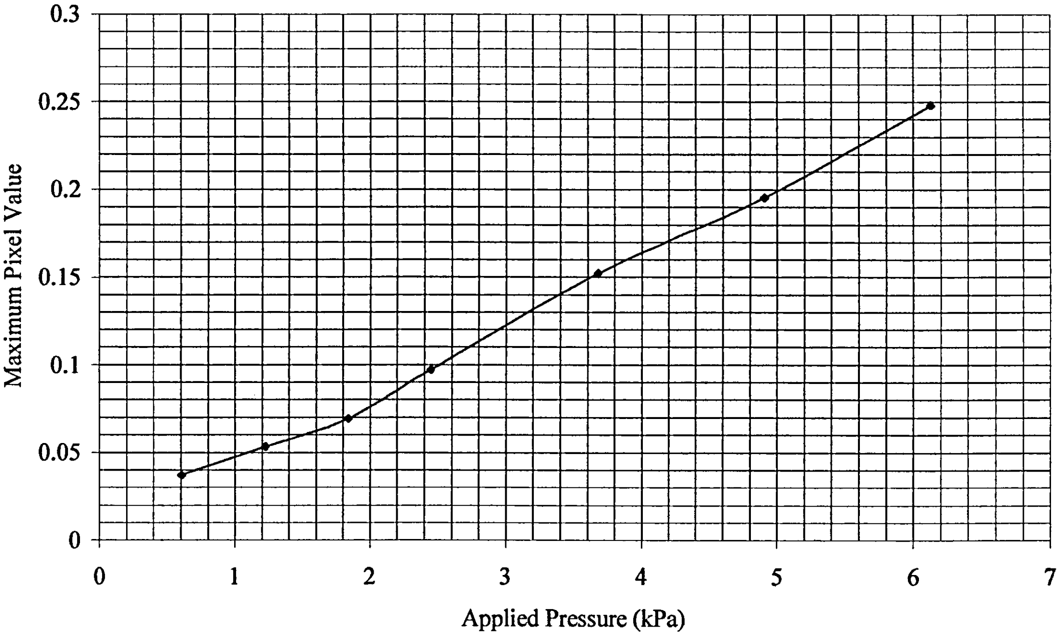


Figure 6.4.22 Maximum Pixel Value Against Applied Pressure, 16 cm^2 Object at Centre of Sensor

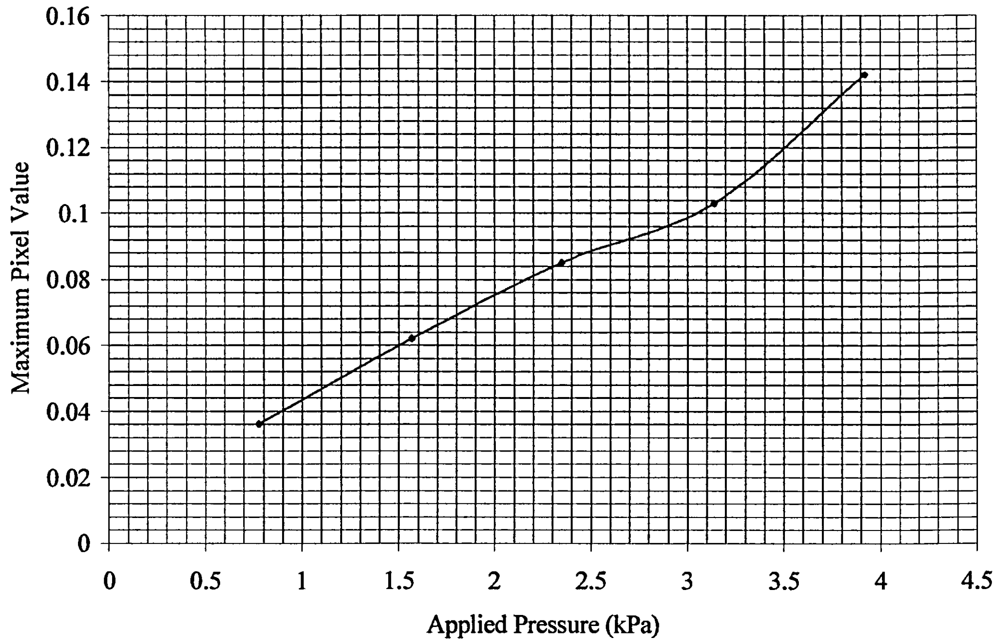


Figure 6.4.23 Maximum Pixel Value against Applied Pressure, 25 cm^2 Object at Centre of Sensor

6.5 Conclusions

It is evident from Figure 6.4.4 to 6.4.15 and Tables 6.4.4 to 6.4.15 that the sensor has a well-defined response in terms of Visibility and Fractional Change in Voltage due to the range of pressure applied. However from Figure 6.4.10 to 6.4.15 and Tables 6.4.9 to 6.4.14 it is clear that the larger the area to which the pressure is applied the more hysteresis is observed. The response is also not as linear as is the case when the pressure is applied to a smaller area. One possible reason for this, which needs researching further, is the way the conductive medium is redistributed by the application of the applied pressure as the size of the area to which pressure is applied is increased. As the size increases more of the conductive medium is forced to the areas where no pressure is applied. This causes the cross section area in these regions to increase and therefore changes the conductivity seen by the EIT system so a point may be reached where the changes in these areas dominates the response of the sensors.

Figures 6.4.16 to 6.4.19 show reconstructed images from the EIT system and they clearly show that the EIT system images the changes in the sensors conductivity due to the increases in applied pressure. The images shown were obtained by loading the raw pixel values generated by the EIT system into MATLAB and scaling the lowest and highest pixel value for each set of image to 0% and 100 % respectively. From the images it is shown that as the area and pressure increases there are changes in conductivity towards the edge of the sensor where there is no load applied. This further proves the hypothesis that the increase in hysteresis and reduction in linearity of the sensor for pressure applied to larger areas could be due to the redistribution of the conductive sensor medium.

It is also evident from Figures 6.4.20 to 6.4.23 which show maximum pixel value against applied pressure, that when pressure is applied to an area of 4 cm^2 and 9 cm^2 the response is linear over the range of applied pressure used were as for areas of 16 cm^2 and 25cm^2 the response loses linearity at the higher pressures.

From the above results it is clear that a system for measuring distributed pressure using EIT is possible however the sensor's response needs to be tailored to the range of pressures expected. This could be achieved by altering the elastic properties of the rubber membrane and or the viscosity of the conductive medium both of which need further research (see Section 7.2).

7 Conclusions And Future Work

The work described in this thesis has been concerned with the development of a distributed pressure measurement system. The main areas of research have been:

- The characterisation of commercially available conductive elastomers for use in a pressure sensor.

- The effect on the performance of an EIT system due to changes in the electrodes distribution.

- The design and development of a novel pressure sensor for use with an EIT system for the measurement of distributed pressure.

7.1 Conclusions

7.1.1 Material Investigation

The investigation into different conductive polymers characterised 7 samples in terms of their V/I characteristics, variation in conductivity due to time (creep) and variation in conductivity due to applied load. It is evident from the investigation into different conductive elastomers for the sensor element (Chapter 4.3.3) that presently commercially available conductive elastomers are not suitable due to:

- The hysteresis in the V/I characteristics due to increasing and subsequently decreasing the applied voltage is too high.

- The changes in resistance with time in the creep characteristics of each sample are too large.

- The changes in conductivity due to the applied pressure are too low.

One polypyrrole sample (sample 7) did have characteristics which may have been suitable for the sensor element but a secure and regular supply of this material of known consistent characteristics could not be found for further investigation.

7.1.2 Electrode Configurations For EIT

Both an experimental and simulation approach have been taken in order to investigate different electrode configurations for EIT. For each of the electrode configurations there is good correlation (0.979 to 1) between the experimental and simulated results for a uniform conductivity distribution.

It is evident from both the experimental and simulated comparisons of the different electrode configurations given in Chapter 5 that distributing the EIT electrodes over the area being interrogated significantly improves the performance of the EIT system. The comparisons also show that the boundary of the area being imaged also has an affect on the performance of an EIT system. This is clearly evident from the comparison of the circular and square phantoms where the electrodes were confined to the periphery. An important effect of the distributed electrode arrangement is that the system has higher sensitivity at the centre than at the periphery. This is opposite to the sensitivity of a system with peripheral electrodes where the sensitivity is higher at the periphery than at the centre.

One measure used in the comparison of the different electrode configurations, namely visibility, can be used to determine if the overall change in conductivity is lower or higher than the background conductivity. This is possible because visibility is positive for an anomaly whose conductivity is less than the background and negative for an anomaly whose conductivity is higher than the background. This characteristic of visibility may be useful in process tomography to determine the specific volume of different phases in 2 or 3 phase flows. Another characteristic of visibility is that for a conductive anomaly of a given size visibility is higher than for an insulating anomaly of the same size.

7.1.3 Novel Sensor Design

The results from the novel pressure sensor described in Chapter 6 show that it is possible to measure distributed pressure using EIT. The results show that above set limits in both area and applied pressure the sensor no longer has a linear response; this would therefore limit its operating range. The construction of the sensor as described would also limit the application area as the sensor has to remain horizontal. When the sensor is vertical more of the conductive fluid moves into the lower half and therefore the conductivity distribution is no longer uniform. The response of the sensor is determined by the elastic properties of the rubber membrane and the viscosity of the conductive fluid. This is evident from the fact that the first conductive medium had a different response to the second conductive medium (i.e. changes in the conductive medium produced changes in the response of the sensor). Changing the properties of both, the rubber membrane and/or conductive fluid should therefore allow the response of the sensor to be tailored to meet a specific application.

One application area would be the constant monitoring of the distributed interfacial pressure between a patient and their support surface, for the prevention of pressure sores. One limitation of the previous work in this area that is described in Chapter 4.2.1 is the effect the measurement systems have on the actual interface between the patient and their support surface and their effect on the pressure distribution. The proposal put forward for the sensor developed in Chapter 6 is that the sensor itself would form the support surface.

7.2 Future Work

A number of suggestions for future work can be identified both for the conductive sensor element for distributed pressure measurement and for EIT in general. The initial proposed sensor element based on a conductive sheet still remains unresolved and the polypyrrole sample which showed that it may have the desired characteristics needs to be further researched as this may provide the ideal sensor material for a range of applications. The ideal sensor material would have negligible hysteresis in their V/I characteristics, negligible creep and large changes in conductivity due to applied load. Also for a sensor based on a conductive sheet, it should be possible for it to be used in any orientation and would be simpler to manufacture. The EIT reconstruction algorithm would be made simpler if the electrodes were made to the same depth as the conductive sheet as the reconstruction problem would be truly 2 dimensional.

The sensor element described in Chapter 6 needs to be developed further in order to better understand how the redistribution of the conductive medium affects the performance of the sensor. In order to achieve this a three-dimensional model of the sensor element would need to be generated. An investigation into the response of the sensor element when pressure in applied in different areas can then be undertaken. The effect on performance of conductive liquids with different viscositys must also be researched along with the effect of changing the elastic properties of the rubber membrane.

The reconstruction algorithm used in this work produced images of changes in conductivity due to applied load and these are shown in Chapter 6. The images produced are basic unfiltered images and as the reconstruction algorithm used is backprojection by sensitivity coefficients some form of filtering should be applied in order to reduce the blurring inherent in any EIT algorithms (Barber D.C., Brown B.H. and Freeston I.L., 1983). To determine a suitable filter for the backprojection by sensitivity coefficient algorithm Kotre C.J., 1993 investigated the width of the point spread function of each pixel in the image and found it to be spatially variant and therefore before applying spatial filtering techniques the image space was transformed to what Kotre call equi-resolution space. A Wiener filter for a Gaussian PSF was then applied to the equi-resolution space after which the inverse of equi-resolution space transformation was applied. The results presented by Kotre C.J. 1993 showed that the filtering technique did improve the resolution of the images.

In terms of EIT in general the effects of boundary shape, electrode configuration, errors in boundary shape and errors in electrode position on the performance of EIT systems needs to be further investigated. All these affect are particularly important in medical imaging where the boundary shape changes from patient to patient. In the case of both the sensor described in Chapter 6 and in process tomography although the boundary shape remains the same for a given sensor/application errors in electrode position can still occur.

Also the effects of three dimensional conductivity distributions needs to be investigated as the real world is three dimensional and the effect on an EIT system due to changes in conductivity above and below the plane of the electrodes needs to be better understood. This is particularly important to the sensor described in Chapter 6 as the EIT electrodes are not in the same plane as the applied load, that is the electrodes are on the bottom face of the sensors whereas the load is applied to the top face of the sensor.

References

Avis N. J. and Barber D.C., 1994, Image Reconstruction Using Non-Adjacent Drive Configurations, Physiol. Meas., 15, Suppl. A, pp153-160.

Barber D.C. and Brown B.H., 1984, Applied Potential Tomography, J. Phys. Eng. Sci. Instrum., 12, pp723-33.

Barber D.C., Brown B.H. and Freeston I.L., 1983, Imaging Spatial Distributions Of Resistivity Using Applied Potential Tomography, Electron. Lett., 19, pp933-5

Barber D. C. and Brown B. H., 1987, Fast Reconstruction Of resistive Images, Clin. Phys. Physiol. Meas., 8 (Suppl A), pp47-54

Brown B.H. and Seagar A.D., 1985, Applied Potential Tomography: Data Collection Problems. Proc. IEE. Int. Conf. Electric and Magnetic Fields In Medicine and Biology, pp79-82.

Brown B.H. and Seagar A.D., 1987, The Sheffield Data Collection System, Clin. Phys. Physiol. Meas., 8, Suppl A, pp91-97.

Basarab-Horwath I., Piotrowski J. and McEwan P. M., 1995, Calculated Measures Of Performance In Electrical Impedance Tomography Using Finite Element Model, Physiol. Meas., 16, pp263-71.

Checinski S. S. and Agrawal A. K., 1984, Magnetoelastic Tactile Sensors, The International Society for Optical Engineering, 449, pt. 2, pp468-74.

Chu Z, Sarro P.M. and Middelhoek S., 1996, Silicon Three axial Tactile Sensor, Sensors and Actuators-A- (Physical), A54 No 1-3, pp505-10.

Daniels A.R., 1996, Dual Modality Tomography For The Monitoring Of Constituent Volumes In Multi-Component Flows, PhD Thesis, Sheffield Hallam University.

Fulton W.S. and Lipczynski R.T., 1993, Body Support Pressure Measurement Using Electrical Impedance Tomography, Proc. 15th International Conf. Of The IEEE Engineering In Medicine And Biology Society, 15, pp98-99

Gisser D.G, Isaacson D. and Newell J.C., 1987, Current Topics In Impedance Imaging, Clin. Phys. Physiol. Meas., 8, Suppl. A, pp36-46.

Ghani N., 1988, Visual And Tactile Senses In Collaboration, Sensors Review, October, pp 210-15.

Griffiths D.H. and Barker R.D., 1993, Two-Dimensional Resistivity Imaging and Modelling in Areas of Complex Geology, Journal of Applied Geophysics, 29, pp211-226.

Griffiths H., 1988, A Phantom for Electrical Impedance Tomography, Clin. Phys. Physiol. Meas., 9 (suppl A), pp5-14.

Griffiths H. and Zhang Z., 1989, Dual Frequency Electrical Impedance Tomography in Vitro and in Vivo, Proc. Annu. Int. Conf. IEEE Eng. in Medicine and Biology Society, 11, pp476-7.

Groover M.P., Weiss M., Nagel R.N. and Odrey N.G., 1986, Industrial Robotics Technology, Programming and Applications, Pub. McGraw-Hill Inc., ISBN 0-07-100442-4 pp144-53.

Helsel M., Zemel J.N. and Dominko V., 1988, An Impedance Tomographic Tactile Sensor, Sensors and Actuators, 14, pp93-98.

Henderson R.P. and Webster J.G., 1978, An Impedance Camera For Spatially Specific Measurements Of The Thorax, IEEE Trans. Biomed. Eng., BME-25, pp250-4.

Hennig E.M., Cavanagh P.R., Albert H.T. and Macmillan N.H., 1982, A Piezoelectric Method of Measuring The Vertical Contact Stress Beneath The Human Foot, Journal of Biomedical Engineering, 10, pp213-22

Horowitz P and Hill W, 1987, The Art Of Electronics, Pub. Cambridge University Press, ISBN 0-23151-5.

Hua P., Webster J.G. and Tompkins W.J. 1987, Effect Of Measurement Method On Noise Handling and Image Quality Of EIT Imaging, Proc. Annu. Int. Conf. IEEE Engineering Medicine and Biology Society, 9, pp1429-30.

Huebner K.H., 1975, The Finite Element Method For Engineers, Pub. John Wiley & Son, ISBN 0-471-41950-8.

Hutchison J.M.S. and Kularni V., 1995, A Novel 16 Electrode Impedance Imaging System, IEE Colloquium, Innovations In Instrumentation For Electrical Tomography, Digest No: 1995/099, pp91-93.

Kim Y. and Woo H.W., 1987, A Prototype System And Reconstruction Algorithm For Electrical Impedance Technique In Medical Imaging, Clin. Phys. Physiol. Meas., 8, Suppl. A, pp63-70.

Kolesar Jr E., Reston R.R., Ford D.G. and Fitch Jr C., 1992, Multiplexed Piezoelectric Polymer Tactile Sensor, Journal of Robotic Systems, 9, No 1, pp37-63.

Kolesar Jr E. and Dyson C. S., 1995, Object Imaging with a Piezoelectric Robotic Tactile Sensor, Journal of Microelectromechanical Systems, 4 No. 2, pp87-96.

Kosina S., Baluch S. and Annus J., 1994, Study On The Electrical Conductivity and Morphology Of Porous Polypyrrole Layers Prepared Electrochemically In The Presence Of Pyridinium Chlorochromate, Journal Of Materials Science, 29, pp3403-3407.

Kotre C.J., 1989, A Sensitivity Coefficient Method For The Reconstruction Of Electrical Tomograms, Clin. Phys. Physiol. Meas., 10, N^{o} 3, pp275-81.

Kotre C.J., 1993, Studies of Image Reconstruction Methods For Electrical Impedance Tomography, PhD These, University Of Newcastle-upon-Tyne.

Kotre C.J., 1995, Subsurface Electrical Impedance Imaging Using Orthogonal Linear Electrode Arrays, IEE Colloquium, Innovations in Instrumentation for Electrical Tomography, Digest No: 1995/099, pp12/1-12/4.

Kotre C.J., 1996, Subsurface Electrical Impedance Imaging Using Orthogonal Linear Electrode Arrays, IEE Proc Sci. Meas. Technol., 143, No 1, pp41-46.

Knight R.A. and Lipczynski R.T., 1990, Body Support Interface Pressure Analysis, IEE Colloquium On Physiological Pressure Measurement, Digest N° 162.

Knight R.A., 1991, The Use Of EIT Techniques To Measure Interface Pressure, Ph.D. Thesis, University Of Bath

Lacy P. Basarab-Horwath I., 1993, An EIT Based Distributed Pressure Sensor Further Results, Process Tomography A Strategy For Industrial Exploitation, pp202-5, ISBN 0-9523165-01.

Lindan O., Greenway R.M., and Piazza J.M., Pressure Distribution On The Surface Of The Human Body: Evaluation in Lying and Sitting Positions Using A "Bed of Springs and Nails", Archives of Physical Medicine and Rehabilitation, 46, pp378-85,May 1965

Lord M., Reynold D.D. and Hughes J.R., 1986, Foot Pressure Measurement, A Review Of Clinical Findings, J. Biomed. Eng., 8, pp283-284.

Lyon G.M. and Oakley J.P., 1992, A Simulation Study Of Sensitivity In Stirred Vessel Electrical Impedance Tomography, Proc. ECAPT Conf. Manchester, pp104-3

Maalej N., Webster J.G., Tompkins W.J., Wertsch J.J., 1988, A Conductive Polymer Pressure Sensor Array, Proc. IEEE Engineering in Medicine and Biology Society 11th Annual Inter. Conf., pp1116-7.

Mayer J.W. and Lau S.S., 1990, Electronic Materials Science: For Integrated Circuits In Si And GaSa, Ed. By Jonnstone D. Pub. Macmillan Publishing Company, ISBN 0-02-378140-8, pp32-35.

Mooney V., Einbund M.J., Rogers J.E. and Stauffer E.S., 1971,Comparison of Pressure Distribution Qualities in Seat Cushions. Bulletin of Prosthetics Research, pp129-43, Spring 1971.

Newell J.C., Gisser D.G. and Isaacson D, 1988, An Electric Current Tomography, IEEE Trans. Biomed. Eng., 35, pp828-33.

Naarmann H., 1990, Correlation Between Active Agents and Electrically Conducting Polymers, 6th European Physical Society Industrial Workshop, Lufthus, Norway, 28-31May 1990.

Nordin M.J., 1995, An Image Reconstruction Algorithm for a Dual Modality Tomographic System, PhD Thesis, Sheffield Hallam University.

Norman R.H., 1970, Conductive Rubbers And Plastics, Pub. Applied Science Publishers Ltd., ISBN 0-444-20074-6.

Nowicki D.J. and Webster J.G., 1989, A One Op-Amp Current Source For Electrical Impedance Tomography, Proc. Annu. Int. Conf. IEEE Engineering In Medicine and Biology Society, 11, pp457-8.

Pax Jr. R.A., Webster J.G. and Radwin R.G., 1989, A Conductive Polymer Sensor for the Measurement of Palmer Pressures, Proc. IEEE Engineering in Medicine and Biology Society 11th Annual Int. Conf., pp1483-84.

Patil K.M. and Babu T.S., 1986, Pressure Distribution Under Leprotic Feet and Footwear for Prevention of Ulcers, Proc. IEEE Engineering in Medicine and Biology Society 8th Annual International Conference, pp1844-45.

Powell H.M. Barber D.C. and Freeston I.L., 1985, Impedance Imaging Using A Linear Electrode Array, Proc. IEE Int. Conf. On Electric and Magnetic Fields In Medicine and Biology, pp88-92.

Regtien P. P. L., 1989, Sensor Systems For Robotics, Sensors and Actuators, 17, pp91-101.

Robertson B.E. and Walken A.J., 1985, Tactile Sensor System For Robotics, Measurement And Control, 18, September, pp262-65

Sansen W., Geeraert B., Van Petegem W. and Steyaert M., 1992, Electical Impedance Tomography Systems Based On Voltage Drive, Clin. Phys. Physiol. Meas., 13, Suppl. A, pp39-42.

Seagar A.D., Barber D. C. and Brown B. H., 1987, Theoretical Limits To Sensitivity And Resolution In Impedance Imaging, Clin. Phys. Physiol. Meas., 8, Suppl. A, pp13 - 31

Seekircher J. and Hoffmann B., 1988, Improved Tactile Sensors, IFAC Robot Control, Karlsruhe, FRG, pp312-22.

Simpson J.C., Tozer R.C. and Freeston I.L., 1996. A Fast EIT system Applied To a Linear Array, IEE Colloquium, Advances In Electrical Tomography, Digest No: 96/143, pp-4/1-4/3.

Soames R.W., 1985, Foot Pressure Patterns During Gait, Journal of Biomedical Engineering, 7, pp120-26.

Sommer G., Leibbrandt S.L. and Stojanoff C. G., 1994, New Tactile Sensor Based on Holographic Optical Elements, Proceedings of The-International Society for Optical Engineering, 2247, pp126-36.

Tekada S.,1974, Study of Artificial Tactile Sensors for Shape Recognition, Proc. 4th Inter. Symp. On Industrial Robotics, Tokyo, pp199-208

Wang M., Dickin F.J. and Beck M.S., 1992, Improved Electrical Impedance Tomography Data Collection System And Measurement Protocols, Proc. ECAPT 1992 Process Tomography, A Strategy For Industrial Exploitation.

Warren-Forward M.J., Goodall R.M. and Pratt D.J., 1992, Three Dimensional Displacement And Force Transducer, IEE Proceedings A, 139, No 1, pp21-29.

Webster J.G., 1989, A Pressure Mat For Preventing Pressure Sores, IEEE Engineering In Medicine and Biology Society 11th Annual International Conf., pp1485-6.

Webster J.G, 1990, Electrical Impedance Tomography, Pub. Adam Hilger, ISBN 0-85274-304-1.

Weng Wah Loh and Dickin F.J., 1996, Improved Modified Newton-Raphson Algorithm For electrical Impedance Tomography, Electronic Letters, 32, No 3, pp 206-207.

Xie C.G., 1993, Review of Image Reconstruction Methods for Process Tomography, Process Tomography – A Strategy For Industrial Exploitation- 1993, pp115-19, ISBN0-9523165-01.

Yorkey T.J., 1986, A quantitative Comparison Of The Reconstruction Algorithms Used In Electrical Impedance Tomography, PhD Thesis, Dept Elec. Comp, Eng. University of Wisconsin, USA.

Yu Z.Z., Peyton A.T., Beck M.S., Conway W.F. and Xu L.A., 1993, Imaging system Based On Electromagnetic Tomography (EMT), Electronic Letters, 29 No 7, pp625-26.

Appendix A Publications From Work

Distributed Pressure Measurement Using Electrical Impedance Tomography: Initial Results, M J Booth and I Basarab_Horwath, IEE Colloquium Advances In Electrical Tomography 19th June 1996.

Comparing Electrode Configurations for Electrical Impedance Tomography, M J Booth and I Basarab_Horwath, Electronic Letters, 28th March 1996, pp648-649.

An Experimental Comparison Of Three Electrode Configurations for Electrical Impedance Tomography, M J Booth and I Basarab_Horwath, Sensors and Their Applications VII, Dublin September 1995, pp273-277.

Analysis of Tomographic Data Using Transform Techniques - A Preliminary Study, I Basarab-Horwath and M J Booth, Sensors and Their Applications VII, Dublin September 1995, pp297-302.

A Comparison Of Three Electrode Configurations for Electrical Impedance Tomography, M J Booth and I Basarab_Horwath, IEE Colloquium On Innovations In Instrumentation for Electrical Tomography, 11th May 1995, pp11/1-11/3.

Appendix B EIT MATLAB Simulations

```
% Simulation of circular phantom for electrical impedance tomography
% 09/01/1996

clear
%Element matrix
Gele = [3 -1 -1 -1;-1 3 -1 -1;-1 -1 3 -1;-1 -1 -1 3];

%system  matrix
Gsys = zeros(276);

%EIT Electrode element numbers
Elect = [4 7 31 78 129 180 223 241 238 235 211 164 113 62 19 1];

%EIT Current Source Locations
source = [4 7;7 31;31 78;78 129];

%Load system topology matrix
load topol.dat

%Set all elements weights to 1
Gv=ones(241,1);

%Define
A = [19:31 33:45 48:60 64:76 81:93 98:110 115:127 132:144 149:161
166:178 182:194 197:209 211:223];
d = size(A);
for i = 1:d(2)
  Gv(A(i)) = 1000;
end


%Set electrode conductivities in Gv
for i = 1:16
  Gv(Elect(i)) = 10000;
end

%Assemble system conductance matrix
for i=1:241
  for j=1:4
    x=topol(i,j);
    for k=1:4
      y=topol(i,k);
      Gsys(x,y) = Gsys(x,y)+(Gv(i)*Gele(j,k));
    end
   end
end

clear Gele;
clear Gv;

%Define current vectors for projections 1 to 4

I1 = zeros(276,1);
I2 = zeros(276,1);
I3 = zeros(276,1);
I4 = zeros(276,1);

for i =1:4
     I1(topol(source(1,2),i)) =  1;
```

```
      I2(topol(source(2,2),i)) =  1;
      I3(topol(source(3,2),i)) =  1;
      I4(topol(source(4,2),i)) =  1;
end

%Set boundary conditions for projection 1 to 4
G1=Gsys;
G2=Gsys;
G3=Gsys;
G4=Gsys;
clear Gsys;
for  i = 1:276
   for j = 1:4
      G1(topol(source(1,1),j),i)=0;
      G1(i,topol(source(1,1),j))=0;
      G2(topol(source(2,1),j),i)=0;
      G2(i,topol(source(2,1),j))=0;
      G3(topol(source(3,1),j),i)=0;
      G3(i,topol(source(3,1),j))=0;
      G4(topol(source(4,1),j),i)=0;
      G4(i,topol(source(4,1),j))=0;
   end
end

for i= 1:4
  G1(topol(source(1,1),i),topol(source(1,1),i)) = 1;
  G2(topol(source(2,1),i),topol(source(2,1),i)) = 1;
  G3(topol(source(3,1),i),topol(source(3,1),i)) = 1;
  G4(topol(source(4,1),i),topol(source(4,1),i)) = 1;
end


%Calculate nodal voltages for projections 1 to 4

V1 = G1\I1;
V2 = G2\I2;
V3 = G3\I3;
V4 = G4\I4;

%Calculate differential boudary voltages for projections 1 to 4

proj1 = [3 4 5 6 7 8 9 10 11 12 13 14 15 16];
proj2 = [4 5 6 7 8 9 10 11 12 13 14 15 16 1];
proj3 = [5 6 7 8 9 10 11 12 13 14 15 16 1 2];
proj4 = [6 7 8 9 10 11 12 13 14 15 16 1 2 3];

for i=1:14
   sum1 = 0;
   sum2 = 0;
   sum3 = 0;
   sum4 = 0;
   for j = 1:4
      sum1 = sum1 + V1(topol(Elect(proj1(i)),j));
      sum2 = sum2 + V2(topol(Elect(proj2(i)),j));
      sum3 = sum3 + V3(topol(Elect(proj3(i)),j));
      sum4 = sum4 + V4(topol(Elect(proj4(i)),j));
   end
   Vo1(i) = sum1/4;
   Vo2(i) = sum2/4;
   Vo3(i) = sum3/4;
   Vo4(i) = sum4/4;
end

for i=1:13
```

```
  Vout(1,i) = Vo1(i)-Vo1(i+1);
  Vout(2,i) = Vo2(i)-Vo2(i+1);
  Vout(3,i) = Vo3(i)-Vo3(i+1);
  Vout(4,i) = Vo4(i)-Vo4(i+1);

end

% Simulation of square phantom for electrical impedance tomography
% With electodes on the periphery
clear

%Element matrix
Gele = [3 -1 -1 -1;-1 3 -1 -1;-1 -1 3 -1;-1 -1 -1 3];

%System  matrix
Gsys = zeros(324);

%Define EIT Electrode element numbers
Elect = [1 5 9 13 17 85 153 221 289 285 281 277 273 205 137 69];

%Define EIT Current Source Locations
source = [1 5;5 9;9 13;13 17];

%Assemble topology matrix
A =[1 2 20 19];
for i=0:17:288
  for j = 1:17
    for k = 1:4
      topol(j+i,k) = A(k);
    end
    A=A+1;
  end
  A=A+1;
end
clear A;

%Define elements weights
Gv=ones(289,1);

%set  Conductivity of electrodes

for i = 1:16
  Gv(Elect(i)) = 10000;
end

%Define anomaly size and conductivity
A = [145];
d = size(A);
for j = 1:d(2)
  for i = 1:d(2)
   Gv(A(i)) = 1000;
  end
 A = A+17;
end
clear A

%Assemble system conductance matrix
for i=1:289
  for j=1:4
    x=topol(i,j);
    for k=1:4
      y=topol(i,k);
      Gsys(x,y) = Gsys(x,y)+(Gv(i)*Gele(j,k));
    end
   end
end
clear Gele;
clear Gv;

%Define current vectors for projections 1 to 4
I1 = zeros(324,1);
I2 = zeros(324,1);
I3 = zeros(324,1);
I4 = zeros(324,1);

for i =1:4
     I1(topol(source(1,2),i)) =  1;
     I2(topol(source(2,2),i)) =  1;
     I3(topol(source(3,2),i)) =  1;
     I4(topol(source(4,2),i)) =  1;
end

%Define boundary conditions for projection 1 to 4
G1=Gsys;
G2=Gsys;
G3=Gsys;
G4=Gsys;
clear Gsys;
for  i = 1:324
   for j = 1:4
     G1(topol(source(1,1),j),i)=0;
     G1(i,topol(source(1,1),j))=0;
     G2(topol(source(2,1),j),i)=0;
     G2(i,topol(source(2,1),j))=0;
     G3(topol(source(3,1),j),i)=0;
     G3(i,topol(source(3,1),j))=0;
     G4(topol(source(4,1),j),i)=0;
     G4(i,topol(source(4,1),j))=0;
   end
end
for i= 1:4
  G1(topol(source(1,1),i),topol(source(1,1),i)) = 1;
  G2(topol(source(2,1),i),topol(source(2,1),i)) = 1;
  G3(topol(source(3,1),i),topol(source(3,1),i)) = 1;
  G4(topol(source(4,1),i),topol(source(4,1),i)) = 1;
end

%Calculate nodal voltages for projections 1 to 4

V1 = G1\I1;
V2 = G2\I2;
V3 = G3\I3;
V4 = G4\I4;


%Calculate defferetia boundary voltages for projections 1 to 4

proj1 = [3 4 5 6 7 8 9 10 11 12 13 14 15 16];
proj2 = [4 5 6 7 8 9 10 11 12 13 14 15 16 1];
proj3 = [5 6 7 8 9 10 11 12 13 14 15 16 1 2];
proj4 = [6 7 8 9 10 11 12 13 14 15 16 1 2 3];

for i=1:14
   sum1 = 0;
   sum2 = 0;
   sum3 = 0;
   sum4 = 0;
   for j = 1:4
```

```
     sum1 = sum1 + V1(topol(Elect(proj1(i)),j));
     sum2 = sum2 + V2(topol(Elect(proj2(i)),j));
     sum3 = sum3 + V3(topol(Elect(proj3(i)),j));
     sum4 = sum4 + V4(topol(Elect(proj4(i)),j));
   end
   Vo1(i) = sum1/4;
   Vo2(i) = sum2/4;
   Vo3(i) = sum3/4;
   Vo4(i) = sum4/4;
end

for i=1:13
  Vout(1,i) = Vo1(i)-Vo1(i+1);
  Vout(2,i) = Vo2(i)-Vo2(i+1);
  Vout(3,i) = Vo3(i)-Vo3(i+1);
  Vout(4,i) = Vo4(i)-Vo4(i+1);

end
```

```
% Simulation of square phantom for electrical impedance tomography
% With Distributed Electrodes

%Define element matrix
Gele = [3 -1 -1 -1;-1 3 -1 -1;-1 -1 3 -1;-1 -1 -1 3];

% Setup system  matrix
Gsys = zeros(289);

%Define EIT Electrode element numbers
Elect = [1 6 11 16 96 176 256 251 246 241 161 81 86 91 166 171];

%Define EIT Current Source Locations
source = [1 6;6 11;11 16;81 86;86 91;91 96];

%Assemble topology matrix
A =[1 2 19 18];
for i=0:16:255
  for j = 1:16
    for k = 1:4
      topol(j+i,k) = A(k);
    end
    A=A+1;
  end
  A=A+1;
end
clear A;

%define elements weights
Gv=ones(256,1);


%Define anomaly size and conductivity
A = [35:46];
d = size(A);
for j = 1:d(2)
  for i = 1:d(2)
   Gv(A(i)) = 1000;
  end
 A = A+16;
end
clear A
```

```
%Define Electodes conductivity
for i = 1:16
  Gv(Elect(i)) = 10000;
end

%Assemble system conductance matrix
for i=1:256
  for j=1:4
    x=topol(i,j);
    for k=1:4
      y=topol(i,k);
      Gsys(x,y) = Gsys(x,y)+(Gv(i)*Gele(j,k));
    end
   end
end
clear Gele;
clear Gv;

%Define current vectors for projections 1 to 6

I1 = zeros(289,1);
I2 = zeros(289,1);
I3 = zeros(289,1);
I4 = zeros(289,1);
I5 = zeros(289,1);
I6 = zeros(289,1);

for i =1:4
     I1(topol(source(1,2),i)) =  1;
     I2(topol(source(2,2),i)) =  1;
     I3(topol(source(3,2),i)) =  1;
     I4(topol(source(4,2),i)) =  1;
     I5(topol(source(5,2),i)) =  1;
     I6(topol(source(6,2),i)) =  1;
end

%Define boundary conditions for projection 1 to 6
G1=Gsys;
G2=Gsys;
G3=Gsys;
G4=Gsys;
G5=Gsys;
G6=Gsys;
clear Gsys;

for  i = 1:289
   for j = 1:4
     G1(topol(source(1,1),j),i)=0;
     G1(i,topol(source(1,1),j))=0;
     G2(topol(source(2,1),j),i)=0;
     G2(i,topol(source(2,1),j))=0;
     G3(topol(source(3,1),j),i)=0;
     G3(i,topol(source(3,1),j))=0;
     G4(topol(source(4,1),j),i)=0;
     G4(i,topol(source(4,1),j))=0;
     G5(topol(source(5,1),j),i)=0;
     G5(i,topol(source(5,1),j))=0;
     G6(topol(source(6,1),j),i)=0;
     G6(i,topol(source(6,1),j))=0;
   end
end

for i= 1:4
  G1(topol(source(1,1),i),topol(source(1,1),i)) = 1;
  G2(topol(source(2,1),i),topol(source(2,1),i)) = 1;
  G3(topol(source(3,1),i),topol(source(3,1),i)) = 1;
  G4(topol(source(4,1),i),topol(source(4,1),i)) = 1;
  G5(topol(source(5,1),i),topol(source(5,1),i)) = 1;
  G6(topol(source(6,1),i),topol(source(6,1),i)) = 1;
end


%Calculate nodal voltages for projections 1 to 6

V1 = G1\I1;
V2 = G2\I2;
V3 = G3\I3;
V4 = G4\I4;
V5 = G5\I5;
V6 = G6\I6;


% Calculate Differential boundary voltages for projections 1 to 6

for i=1:16
   sum1 = 0;
   sum2 = 0;
   sum3 = 0;
   sum4 = 0;
   sum5 = 0;
   sum6 = 0;
   for j = 1:4
     sum1 = sum1 + V1(topol(Elect(i),j));
     sum2 = sum2 + V2(topol(Elect(i),j));
     sum3 = sum3 + V3(topol(Elect(i),j));
     sum4 = sum4 + V4(topol(Elect(i),j));
     sum5 = sum5 + V5(topol(Elect(i),j));
     sum6 = sum6 + V6(topol(Elect(i),j));
   end
   Vo1(i) = sum1/4;
   Vo2(i) = sum2/4;
   Vo3(i) = sum3/4;
   Vo4(i) = sum4/4;
   Vo5(i) = sum5/4;
   Vo6(i) = sum6/4;
End


D1 = [3 4;12 13;13 14;14 5;11 15;15 16;16 6;10 9;9 8;8 7;12 11;
      11 10;13 15;15 9;3 14;14 16;16 8;4 5;5 6;6 7];

D2 = [12 13;13 14;14 5;11 15;15 16;16 6;10 9;9 8;8 7;1 12;12 11;
      11 10;13 15;15 9;14 16;16 8;4 5;5 6;6 7;6 6];

D3 = [1 2;12 13;13 14;14 5;11 15;15 16;16 6;10 9;9 8;8 7;1 12;12 11;
      11 10;2 13;13 15;15 9;14 16;16 8;5 6;6 7];

D4 = [1 2;2 3;3 4;14 5;11 15;15 16;16 6;10 9;9 8;8 7;11 10;
      15 9;3 14;14 16;16 8;4 5;5 6;6 7;6 6;6 6];

D5 = [1 2;2 3;3 4;11 15;15 16;16 6;10 9;9 8;8 7;1 12;12 11;
      11 10;15 9;16 8;4 5;5 6;6 7;6 6;6 6;6 6];

D6 = [1 2;2 3;3 4;12 13;11 15;15 16;16 6;10 9;9 8;8 7;1 12;
      12 11;11 10;2 13;13 15;15 9;16 8;6 7;6 6;6 6];

for i=1:20
  Vout(1,i) = abs(Vo1(D1(i,1))-Vo1(D1(i,2)));
  Vout(2,i) = abs(Vo2(D2(i,1))-Vo2(D2(i,2)));
  Vout(3,i) = abs(Vo3(D3(i,1))-Vo3(D3(i,2)));
  Vout(4,i) = abs(Vo4(D4(i,1))-Vo4(D4(i,2)));
  Vout(5,i) = abs(Vo5(D5(i,1))-Vo5(D5(i,2)));
  Vout(6,i) = abs(Vo6(D6(i,1))-Vo6(D6(i,2)));


end
```

Appendix C EIT Software Listing

```
/*********************************************************************/
/* Electrical Impedance Tomography: Data Acquisition and Image Reconstruction */
/* Software                                                          */
/*                                                                   */
/*     Written By: M.J.Booth              Date: 16/8/95              */
/*                                                                   */
/*     Version: 4.0                                                  */
/*********************************************************************/

/*********************************************************************/
/*                         Header Files                              */
/*********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <graphics.h>
#include <bios.h>
#include <dir.h>
#include <string.h>
#include <ctype.h>
#include <dos.h>
#include <math.h>
#include <alloc.h>

/*********************************************************************/
/*                      Structures                                   */
/*********************************************************************/
#define max_m   24          // maximum number of projections
#define max_n   21          // maximum number of reading per projection
#define max_p   225         // maximum number of pixels
#define num_pix_r      145  // maximum number of pixels for round phantom
#define num_pix_s      225  // maximum number pf pixels for square phantom
#define CR      0x1C0D      // define carrage return bios key
#define ESC     0x11B       // define escape bios key
#define UP      0x4800      // define up arrow bios key
#define DOWN    0x5000      // define down arrow bios key

struct buttontype
      {
            int number;
            int select;
      };
```

```
struct configtype
	{
		int freq;
		int source[max_m][2];
		int read[max_m][(2*max_n)];
	};

struct button3type
	{
		int first;
		int second;
	};

struct addresstype
	{
		unsigned char gain_freq;
		unsigned char source[max_m];
		unsigned char read[max_m][max_n];
	};

/*****************************************************************/
/*                    Global Variables                           */
/*****************************************************************/
struct configtype config;  // holds all configuration information
struct addresstype addr;  // holds the hex address for the required acquisition sequence
int  ( far *sen_array[max_p])[max_n];     // holds sensitivity coefficients
float data[max_m][max_n];                 // holds most resent data
float pixel_data[max_p];                  // holds pixel values for reconstruction
char patterns[17][8]={
      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},
      {0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x03},
      {0x00,0x00,0x30,0x30,0x00,0x00,0x03,0x03},
      {0x00,0x00,0x30,0x30,0x00,0x00,0x33,0x33},
      {0x00,0x00,0x33,0x33,0x00,0x00,0x33,0x33},
      {0x00,0x00,0x33,0x33,0xc0,0xc0,0x33,0x33},
      {0x0c,0x0c,0x33,0x33,0xc0,0xc0,0x33,0x33},
      {0x0c,0x0c,0x33,0x33,0xcc,0xcc,0x33,0x33},
      {0xcc,0xcc,0x33,0x33,0xcc,0xcc,0x33,0x33},
      {0xcc,0xcc,0x33,0x33,0xcc,0xcc,0x3f,0x3f},
      {0xcc,0xcc,0xf3,0xf3,0xcc,0xcc,0x3f,0x3f},
      {0xcc,0xcc,0xff,0xff,0xcc,0xcc,0x3f,0x3f},
      {0xcc,0xcc,0xff,0xff,0xcc,0xcc,0xff,0xff},
      {0xcc,0xcc,0xff,0xff,0xcf,0xcf,0xff,0xff},
      {0xfc,0xfc,0xff,0xff,0xcf,0xcf,0xff,0xff},
      {0xfc,0xfc,0xff,0xff,0xff,0xff,0xff,0xff},
      {0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff}};

int image_val[max_p];                      // holds pixel values for displayed image
float homog_data[max_m][max_n];            // holds homogenous voltage readings

/*****************************************************************/
/*                    Function Prototypes                        */
```

```
/*******************************************************************/
void config_sys(int active);
void load_config(void);
void save_config(void);
void load_ref_data(void);
void load_data(void);
void save_data(void);
void display_data(void);
void round_image(int cont);
void save_r_image(void);
void save_s_image(void);
void square_image(int mode);
void save_cal(int num_pix);
void load_cal(int num_pix);
void sensitivity_calibration(int num_pix);
void backprojection(int num_pix);
void initialize_graphics(void);
void main_window(void);
void menu1(void);
void recon_r(void);
void recon_s(void);
void acquire_data(int ave, int display);
void acquire_time(void);
void addr_trans(void);
unsigned char comp_addr(int addr_high, int addr_low);
int pause(char *message);
struct buttontype button_select(int max_buttons, int current);
struct buttontype button2_select(int max_contents, int current);
struct viewporttype menu_window(int left, int top,int right,int bottom);
int menu(int left, int top, int right,int bottom, int no_buttons, int num, char *str1[16]);

void button_2(int lwidth, int xloc, int yloc, char *label,int active,
              int *contents, int mult, int no_items);

void button_3(int lwidth, int xloc, int yloc, char *label,int active, int *value1,
              int *value2,int mult,int no_items);

void menu_button(int lwidth, int xloc, int yloc,int acolour,int dcolour,
                   char *string,int status );
```

```
/****************************************************************/
/*                    Main Code                                 */
/****************************************************************/
int main(void)
{
  initialize_graphics();       // Set up system for graphics mode
  main_window();                     // set up screen back ground
  menu1();                           // activate main menu
  closegraph();                      // close graphic mode
  return(0);
}

/****************************************************************/
/*                      Functions                               */
/****************************************************************/

/****************************************************************/
/*    RECON_R: Sets up reconstruction menu for round phantom at right hand   */
/*  side of screen the coordinates are absolute for a 640*350 screen.        */
/*  The function calls the appropriate functions for the menu function chosen. */
/*  Variables passed to function: none                                       */
/*  Variables passed back by function: none                                  */
/*  Include files needed: alloc.h, graphics.h                                */
/*  Global variables:                                                        */
/*            sen_array hold the sensitivity coefficents                     */
/*    max_p,max_m and max_n                                        */
/*  Structures needed: none                                                  */
/*  Functions required: round_image, addr_trans, load_data, load_cal, save_cal  */
/*                      sensitivity_calibration, save_r_image, load_ref_data, menu
     */
/****************************************************************/
void recon_r(void)
{
      int p,m,n,option = 1;
      char *menu_text[16];

      //allocate memory for sensitivity array
      for(p = 0; p < num_pix_r; p++)
      {
            if((sen_array[p]= farmalloc(sizeof(int[max_m][max_n])))== NULL)
            {
                  closegraph();
                  clrscr();
                  printf("Memory allocation error assigning pointer  %d",p);
                  exit(0);
            }
      }
      for(p = 0; p < num_pix_r;p++)
            for(m = 0; m < max_m;m++)
                  for(n = 0; n < max_n;n++)
                        sen_array[p][m][n] = 0;
```

```
        menu_text[0] = "Recon. On-Line";   //set up text for menu buttons
        menu_text[1] = "Recon. Off-Line";
        menu_text[2] = "Recon. With Pause";
        menu_text[3] = "Calibrate System";
        menu_text[4] = "Load Calibration";
        menu_text[5] = "Save Calibration";
        menu_text[6] = "Load Data";
        menu_text[7] = "Save Image";
        menu_text[8] = "Load Ref. Data";
        menu_text[9] = "Return";
        do
        {
                //create menu
                option = menu(470,55,633,344,10,option,menu_text);
                switch( option )
                {
                        case 1:addr_trans(); //complie addresses for hardware
                                round_image(0);
                                break;
                        case 2:round_image(1);
                                break;
                        case 3:addr_trans(); //complie addresses for hardware
                                round_image(2);
                                break;
                        case 4:addr_trans();
                                sensitivity_calibration(num_pix_r);
                                break;
                        case 5:load_cal(num_pix_r);
                                break;
                        case 6:save_cal(num_pix_r);
                                break;
                        case 7:load_data();
                                break;
                        case 8:save_r_image();
                                break;
                        case 9:load_ref_data();
                                break;
                        default:break;
                }
        }
        while(option != 10);
        for(p=0; p < num_pix_r; p++)
                farfree(sen_array[p]);
}
```

```
/*******************************************************************/
/*    RECON_S: Sets up reconstruction menu for square phantom at right hand   */
/*  side of screen the coordinates are absolute for a 640*350 screen.          */
/* The function calls the appropriate functions for the menu function chosen.
       */
/*  Variables passed to function: none                                          */
/*  Variables passed back by function: none                                     */
/*  Include files needed: alloc.h, graphics.h                                   */
/*  Global variables:                                                            */
/*            sen_array hold the sensitivity coefficents                         */
/*     max_p,max_m and max_n                                              */
/*  Structures needed: none                                                     */
/*  Functions required: square_image, addr_trans, load_data, load_cal, save_cal  */
/*                     sensitivity_calibration, save_s_image, load_ref_data, menu
       */
/*******************************************************************/
void recon_s(void)
{
       int p,m,n;
       int option = 1;
       char *menu_text[16];

       for(p = 0; p < num_pix_s; p++) //allocate memory for sensitivity array
       {
              if((sen_array[p]= farmalloc(sizeof(int[max_m][max_n]))) == NULL)
              {
                     closegraph();
                     clrscr();
                     printf("Memory allocation error assigning pointer  %d",p);
                     exit(0);
              }
       }
       for(p = 0; p < num_pix_s;p++)
              for(m = 0; m < max_m;m++)
                     for(n = 0; n < max_n;n++)
                            sen_array[p][m][n] = 0;

       menu_text[0] = "Recon. On-Line";  //set up text for menu buttons
       menu_text[1] = "Recon. Off-Line";
       menu_text[2] = "Recon. With Pause";
       menu_text[3] = "Calibrate System";
       menu_text[4] = "Load Calibration";
       menu_text[5] = "Save Calibration";
       menu_text[6] = "Load Data";
       menu_text[7] = "Save Image";
       menu_text[8] = "Load Ref. Data";
       menu_text[9] = "Return";
do
    {
        //create menu
        option = menu(470,55,633,344,10,option,menu_text);
        switch( option )
        {
            case 1:addr_trans(); //complie addresses for hardware
                square_image(0);
                break;
            case 2:square_image(1);
                break;
            case 3:addr_trans(); //complie addresses for hardware
                square_image(2);
                break;
            case 4:addr_trans();
                sensitivity_calibration(num_pix_s);
                break;
            case 5:load_cal(num_pix_s);
                break;
            case 6:save_cal(num_pix_s);
                break;
            case 7:load_data();
                break;
            case 8:save_s_image();
                break;
            case 9:load_ref_data();
                break;
            default:break;
        }
    }
    while(option != 10);
    for(p=0; p < num_pix_s; p++)
        farfree(sen_array[p]);
}
```

```
/*****************************************************************/
/*    BACKPROJECTION: calculates the pixel values for the reconstructed based
        */  /* on   sensitivity coefficents.
             */
/* Variables passed to function:                                    */
/*          num_pix gives the number of pixels in the reconstructed image        */
/* Variables passed back by function: none                                  */
/* Include files needed: none                                              */
/* Global variables:                                                       */
/*          sen_array holds the sensitivity coefficents                           */
/*    data  holds  the data to reconstruct an image from                  */
/*          homog_data holds data for homogeneous reading                       */
/*          pixel_data holds the pixel values for the reconstructed image        */
/*          max_p,max_m and max_n                                           */
/* Structures needed: none                                                 */
/* Functions required: none                                                */
/*****************************************************************/
void backprojection(int num_pix)
{
    int p,m,n;
    double sum1,sum2;
    double sum3[max_p];

    for(p = 0; p < num_pix ; p++)
    {
        sum3[p]= 0;
        sum1 = 0;
        sum2= 0;
        for(m = 0; m < max_m; m++)
        {
            for(n = 0; n < max_n; n++ )
            {
                if(homog_data[m][n] != 0)
                {
                    sum1=(data[m][n]-
                        homog_data[m][n])/homog_data[m][n];
                    sum2 = sum2 + ((double)sen_array[p][m][n]);
                    sum3[p] =

    sum3[p]+(sum1*((double)sen_array[p][m][n]));
                }
            }
        }
        if (sum2 != 0)
            sum3[p] = sum3[p]/sum2;
        else
            sum3[p] = 0;
    }
```

```
for(p = 0; p < num_pix; p++)
        {
                pixel_data[p] = sum3[p];
                image_val[p]= (int)(200*sum3[p]);
                if(image_val[p] > 16)
                        image_val[p]= 16;
                if(image_val[p] < -16)
                        image_val[p]= -16;
        }
}

/*****************************************************************/
/*    SAVE_R_IMAGE: Save image from round phantom to disk.        */
/* Variables passed to function: none                            */
/* Variables passed back by function none                        */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h */
/* Global variables:                                             */
/*              pixel_data holds image to be saved               */
/* Structures needed: none                                       */
/* Functions required: menu_window, button2 and pause            */
/*****************************************************************/
void save_r_image(void)
{
        struct viewporttype vp;
        int xloc,yloc,m,n,p;
        int file_number = 1;
        int pixel_mask[13][13]=   {0,0,0,1,1,1,1,1,1,1,0,0,0,
                                   0,0,1,1,1,1,1,1,1,1,1,0,0,
                                   0,1,1,1,1,1,1,1,1,1,1,1,0,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   1,1,1,1,1,1,1,1,1,1,1,1,1,
                                   0,1,1,1,1,1,1,1,1,1,1,1,0,
                                   0,0,1,1,1,1,1,1,1,1,1,0,0,
                                   0,0,0,1,1,1,1,1,1,1,0,0,0};
        char cont;
        char str1[35];
        char file_name[15];
        char *path;
        FILE *file_p;

        settextjustify( CENTER_TEXT, CENTER_TEXT ); //set text attributes
        settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );
        vp = menu_window(10,60,460,339);                    //draw screen window
        setfillstyle(SOLID_FILL,EGA_LIGHTBLUE); //draw message window
        bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
        button_2(220,16,32,"Image File Number To Save",0,&file_number,1,100);
```

```
itoa(file_number, str1, 10 );           //build up file name
strcpy(file_name, "RIMAG");
strcat(file_name, str1);
strcat(file_name, ".DAT");

xloc = (vp.right - vp.left)/2;          //calculate centre of window
yloc = (vp.bottom-vp.top)/2;
cont = 'Y';
path = searchpath(file_name);
//if file exists check to overwrite file
if(path != NULL)
{
    strcpy(str1, "File Already Exists Overwrite Y");
    outtextxy(xloc, yloc, str1);    //display check message
    cont = toupper(getch());
}
// if ok to over write file
if(cont == 'Y')
{
    //clear check message from message window
    setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
    bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
    //if can't open file print error message
    if((file_p = fopen(file_name,"wt+")) == NULL)
    {
        strcpy(str1,"Error can't open ");
        strcat(str1,file_name);
        outtextxy(xloc, yloc, str1);
    }
    //else save image to disk
    else
    {
        p = 0;
        for(m = 0; m < 13 ; m++)
        {
            for(n = 0; n < 13; n++)
            {
                if(pixel_mask[m][n] != 0)
                {
                    fprintf(file_p,"%5.3f\t",pixel_data[p]);
                    p++;
                }
                else
                    fprintf(file_p," 0.000\t");
            }
            fprintf(file_p,"\n");
        }
        outtextxy(xloc,yloc,"Image File Saved");
        fclose(file_p);
    }
}
pause("Press Any Key To Continue");
```

```
    setfillstyle(SOLID_FILL,EGA_BLUE);          //close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top );
}

/*******************************************************************/
/* ROUND IMAGE: Displays reconstructed image for round and has three      */
/* modes of operation.                                                    */
/*          Mode 0: continually acquires data and updates image
      */
/*          Mode 1: updates image for current data                        */
/*          Mode 2: as mode 0 but with pause  between up dates            */
/* Variables passed to function:                                          */
/*          mode defines which mode to operate in:                        */
/* Variables passed back by function: none                                */
/* Include files needed:graphics.h,stdlib.h and math.h                    */
/* Global variables:                                                      */
/*          pixel_data array holds pixel values                           */
/*          Data array holds most rescent data acquired               */
/* Structures needed: none                                                */
/* Functions required: menu_window, menu_button, aquire_data, pause,      */
/*******************************************************************/
void round_image(int mode)
{
    struct viewporttype vp;
    struct palettetype palette;
    int x,y,p,xloc,yloc,key;
    int n = 0;
    int pixel_mask[13][13]= {0,0,0,1,1,1,1,1,1,1,0,0,0,
                             0,0,1,1,1,1,1,1,1,1,1,0,0,
                             0,1,1,1,1,1,1,1,1,1,1,1,0,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             1,1,1,1,1,1,1,1,1,1,1,1,1,
                             0,1,1,1,1,1,1,1,1,1,1,1,0,
                             0,0,1,1,1,1,1,1,1,1,1,0,0,
                             0,0,0,1,1,1,1,1,1,1,0,0,0};

    settextjustify(CENTER_TEXT, CENTER_TEXT);
    settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
    vp = menu_window(10,60,460,339);            //draw screen window

    //draw border to image
    setlinestyle(SOLID_LINE,0xffff,THICK_WIDTH);
    setcolor(EGA_BLACK);
    circle((vp.right-vp.left)/2,(vp.bottom - vp.top)/2,115);
    setlinestyle(SOLID_LINE,0xffff,NORM_WIDTH);

    //draw colour scales each side of image
```

```
for(y= 0; y < 17; y++)
{
    setfillpattern(patterns[y],EGA_BLUE);
    bar3d(20,45+12*y,40,57+12*y,0,0);
    setfillpattern(patterns[y],EGA_RED);
    bar3d(420,45+12*y,440,57+12*y,0,0);
}
if(mode == 0)
    menu_button(210,(vp.right-vp.left)/2 - 105,vp.bottom-vp.top - 20,
        EGA_LIGHTCYAN,EGA_CYAN,"Press Any Key To Abort",0);
do
{
    if(mode != 1)
        acquire_data(1,0);
    backprojection(num_pix_r);
    p=0;
    yloc = ((vp.bottom-vp.top)/2) - 78;
    for(y=0; y < 13; y++)
    {
        xloc = ((vp.right  - vp.left)/2) - 98;
        for(x = 0; x < 13; x++)
        {
            if(pixel_mask[x][y] != 0)
            {
                if(image_val[p] < 0 )
                    setfillpattern(patterns[fabs(image_val[p])],
                                EGA_BLUE);
                else
                    setfillpattern(patterns[fabs(image_val[p])],
                                EGA_RED);

                bar3d(xloc,yloc,xloc+15,yloc+12,0,0);
                p++ ;
            }
            xloc = xloc + 15;
        }
        yloc = yloc + 12;
    }
    if(mode == 2)
    {
        key = pause("Press Escape To Exit Any Other Key To Update
                  Image");
        if(key == ESC)
            mode = 3;
    }
    n++;
    if(bioskey(1) != 0)
    {
        mode = 3;
        bioskey(0);
    }
}while((mode == 0)||(mode == 2));
```

```
	if(mode == 1)
		pause("Press Any Key Continue");

	setfillstyle( SOLID_FILL,EGA_BLUE );	// close window
	bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}

/****************************************************************/
/*	SAVE_S_IMAGE: Save image from square phantom to disk.		*/
/* Variables passed to function: none					*/
/* Variables passed back by function none				*/
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h	*/
/* Global variables: array pixel_data holds image to be saved		*/
/* Structures needed: none						*/
/* Functions required: menu_window, button2 and pause		*/
/****************************************************************/
void save_s_image(void)
{
	struct viewporttype vp;
	int xloc,yloc,m,n,p;
	int file_number=1;
	char cont;
	char str1[35];
	char file_name[15];
	char *path;
	FILE *file_p;

	settextjustify( CENTER_TEXT, CENTER_TEXT ); //set text attributes
	settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

	vp = menu_window(10,60,460,339);		//draw screen window

	setfillstyle(SOLID_FILL,EGA_LIGHTBLUE); //draw message window
	bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

	button_2(220,16,32,"Image File Number To Save",0,&file_number,1,100);
	itoa(file_number, str1, 10 );		//build up file name
	strcpy(file_name, "SIMAG");
	strcat(file_name, str1);
	strcat(file_name, ".DAT");

	xloc = (vp.right - vp.left)/2;		//calculate centre of window
	yloc = (vp.bottom-vp.top)/2;
	cont = 'Y';
	path = searchpath(file_name);
	//if file exists check to overwrite file
	if(path != NULL)
	{
		strcpy(str1, "File Already Exists Overwrite Y");
		outtextxy(xloc, yloc, str1);	//display check message
		cont = toupper(getch());
	}
```

```
    // if ok to over write file
    if(cont == 'Y')
    {
        //clear check message from message window
        setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
        bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
        //if can't open file print error message
        if((file_p = fopen(file_name,"wt+")) == NULL)
        {
            strcpy(str1,"Error can't open ");
            strcat(str1,file_name);
            outtextxy(xloc, yloc, str1);
        }
        else //else save image to disk
        {
            p = 0;
            for(m = 0; m < 15; m++)
            {
                for(n = 0; n < 15; n++)
                {
                    fprintf(file_p,"%5.3f\t",pixel_data[p]);
                    p++;
                }
                fprintf(file_p,"\n");
            }
            outtextxy(xloc,yloc,"Image File Saved");
            fclose(file_p);
        }
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);        //close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top);
}

/*****************************************************************/
/* SQUARE IMAGE: Displays reconstructed image for square and has three     */
/* modes of operation.                                                      */
/*          Mode 0: continually acquires data and updates image
    */
/*          Mode 1: updates image for current data                          */
/*          Mode 2: as mode 0 but with pause  between up dates              */
/* Variables passed to function:                                            */
/*          mode defines which mode to operate in:                          */
/* Variables passed back by function: none                                  */
/* Include files needed:graphics.h,stdlib.h and math.h                      */
/* Global variables:                                                        */
/*          image_val array holds pixel values                              */
/*          Data array holds most rescent data acquired                 */
/* Structures needed: none                                                  */
/* Functions required: menu_window, menu_button, aquire_data, pause,        */
/*****************************************************************/
```

```
void square_image(int mode)
{
	struct viewporttype vp;
	int x,y,p,xloc,yloc,key;
	int n = 0;

	settextjustify(CENTER_TEXT, CENTER_TEXT);
	settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);

	vp = menu_window(10,60,460,339);		//draw screen window

	//draw border to image
	setlinestyle(SOLID_LINE,0xffff,THICK_WIDTH);
	setcolor(EGA_BLACK);
	setlinestyle(SOLID_LINE,0xffff,NORM_WIDTH);

	//draw colour scales each side of image
	for(y= 0; y < 17; y++)
	{
		setfillpattern(patterns[y],EGA_BLUE);
		bar3d(20,45+12*y,40,57+12*y,0,0);
		setfillpattern(patterns[y],EGA_RED);
		bar3d(420,45+12*y,440,57+12*y,0,0);
	}
	if(mode == 0)
		menu_button(210,(vp.right-vp.left)/2 - 105,vp.bottom-vp.top - 20,
			EGA_LIGHTCYAN,EGA_CYAN,"Press Any Key To Abort",0);
```

```
    do
    {
        if(mode != 1)
            acquire_data(1,0);
        backprojection(num_pix_s);
        p=0;
        yloc = ((vp.bottom-vp.top)/2) - 90;
        for(y=0; y < 15; y++)
        {
            xloc = ((vp.right  - vp.left)/2) - 113;
            for(x = 0; x < 15; x++)
            {
                if(image_val[p] < 0 )
                    setfillpattern(patterns[fabs(image_val[p])],EGA_BLUE);
                else
                    setfillpattern(patterns[fabs(image_val[p])],EGA_RED);
                bar3d(xloc,yloc,xloc+15,yloc+12,0,0);
                p++ ;
                xloc = xloc + 15;
            }
            yloc = yloc + 12;
        }
        if(mode == 2)
        {
            key = pause("Press Escape To Exit Any Other Key To Update
                              Image");
            if(key == ESC)
                mode = 3;
        }
        n++;
        if(bioskey(1) != 0)
        {
            mode = 3;
            bioskey(0);
        }
    }while((mode == 0)||(mode == 2));
    if(mode == 1)
        pause("Press Any Key Continue");

    setfillstyle( SOLID_FILL,EGA_BLUE );        // close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}
```

```
/*******************************************************************/
/*    SENSITIVITY_CALIBRATION: Sets up calibration window  and guides     */
/* uses throught the calibration procedure.                                        */
/*  Variables passed to function:                                          */
/*          num_pix gives the number of pixels for which there are coefficents     */
/*  Variables passed back by function: none                                          */
/*  Include files needed: graphic.h                                                  */
/*  Global variables:                                                                */
/*          sen_array holds the sensitivity coefficents                              */
/*          array data holds the acquired  data form function acquire data           */
/*          homog_data holds data for homogeneous reading                            */
/*          max_p,max_m and max_n                                                    */
/*  Structures needed: none                                                          */
/*  Functions required: main_window,pause and acquire_data                           */
/*******************************************************************/
void sensitivity_calibration(int num_pix)
{
    struct viewporttype vp;
    int m,n,p = 0,xloc,yloc,key = 0,homog= 1,y = 1;
    char str1[50];
    char str2[50];

    settextjustify( CENTER_TEXT, CENTER_TEXT );//set text attributes
    settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

    vp = menu_window(10,60,460,339);            //draw screen window
    xloc = vp.right-vp.left;
    yloc = vp.bottom-vp.top;

    setcolor(EGA_BLACK);

    while((key != ESC)&&(p < (num_pix)))
    {
        if(homog == 1)
        {
            setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
            bar3d(16,56,xloc-16,yloc-56,0,0); //draw message window
            outtextxy(xloc/2,yloc/2,"Set Phantom For Homogeneous Reading");
            key = pause("Press Escape to exit Any Other Key To Acquire
                            Data");
            if(key != ESC)
            {
                acquire_data(1,0);
                sound(1500);
                delay(100);
                nosound();
                for( m = 0; m < max_m; m++)
                {
                    for( n = 0; n < max_n; n++)
                        homog_data[m][n] = data[m][n];
                }
                homog = 0;
                y = 1;
            }
        }
        else
        {
            setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
            bar3d(16,56,xloc-16,yloc-56,0,0);
            if(p < num_pix)
            {
                itoa(p + 1, str2,10 );
                strcpy(str1,"Load Pixel Location ");
                strcat(str1,str2);
                outtextxy(xloc/2,yloc/2, str1);
                key = pause("Press Escape To Exit Any Other Key To Acquire
                            Data");
            }
            if(key !=ESC)
            {
                acquire_data(1,0);//acquire data for sensitivity coefficent "p+1"
                sound(1500);
                delay(100);
                nosound();
                //calulate sensitivity coefficents
                for( m = 0; m < max_m; m++)
                {
                    for( n = 0; n < max_n; n++)
                    {
                        if((homog_data[m][n] > 0)&&(data[m][n] > 0 ))
                            sen_array[p][m][n]=(int)(1000*
                                (data[m][n]-homog_data[m][n]));
                        else
                            sen_array[p][m][n] = 0;
                    }
                }
                p++;
                if(y == 15)
                    homog = 0;
                y++;
            }
        }
    }
    if(key != ESC)
        pause("Calibration Finished Press Any Key To Continue");
    setfillstyle( SOLID_FILL,EGA_BLUE );     // close window
    bar(0, 0, xloc,yloc);
}

/*****************************************************************/
/*    SAVE CAL: Save sensitivity coefficents and homogeneous data to disk.    */
/* Variables passed to function:                                    */
/*         num_pix gives the number of pixels for which there are coefficents    */
/* Variables passed back by function none                           */
/*  Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h           */
/*  Global variables:                                                                  */
/*          array sen_array holds sensitivity coefficents to be saved                  */
/*          array homog_data holds homogeneous readings to be saved                    */
/*  Structures needed: none                                                            */
/*  Functions required: menu_window, button2 and pause                                 */
/**************************************************************************************/
void save_cal(int num_pix)
{
    struct viewporttype vp;
    int xloc,yloc, m,n,x,file_number = 1;
    char cont;
    char str1[35];
    char file_name[15];
    char *path;
    FILE *file_p;

    settextjustify( CENTER_TEXT, CENTER_TEXT ); //set text attributes
    settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

    vp = menu_window(10,60,460,339);            //draw screen window
    setfillstyle(SOLID_FILL,EGA_LIGHTBLUE); //draw message window
    bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

    button_2(260,16,32,"Calibration File Number To Save",0,&file_number,1,100);

    itoa(file_number, str1, 10 );           //build up file name
    strcpy(file_name, "EIT");
    strcat(file_name, str1);
    strcat(file_name, ".CAL");

    xloc = (vp.right - vp.left)/2;          //calculate centre of window
    yloc = (vp.bottom-vp.top)/2;
    cont = 'Y';
    path = searchpath(file_name);
    //if file exists check to overwrite file

    if(path != NULL)
    {
        strcpy(str1, "File Already Exists Overwrite Y");
        outtextxy(xloc, yloc, str1);    //display check message
        cont = toupper(getch());
    }
    // if ok to over write file
    if(cont == 'Y')
    {
        //clear check message from message window
        setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
        bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
        //if can't open file print error message
        if ((file_p = fopen(file_name,"w+t")) == NULL)
        {
            strcpy(str1,"Error can't open ");
            strcat(str1,file_name);
            outtextxy(xloc, yloc, str1);
        }
        //else save sensitivity coefficents and homogeneous reading
        else
        {
            for(x=0;x < num_pix; x++)
            {
                for(m = 0; m < max_m ; m++)
                {
                    for(n = 0; n < max_n; n++)
                    {
                        fprintf(file_p,"%d\t",sen_array[x][m][n]);
                    }
                    fprintf(file_p,"\n");
                }
                fprintf(file_p,"\n");
            }
            outtextxy(xloc,yloc,"Calibration File Saved");
            fclose(file_p);
        }
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);          //close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}
```

```
/*******************************************************************/
/*    LOAD CAL: Load sensitivity coefficents and homogeneous data to disk.     */
/*  Variables passed to function:                                     */
/*          num_pix gives the number of pixels for which there are coefficents    */
/*  Variables passed back by function none                                */
/*  Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h    */
/*  Global variables:                                                 */
/*          array sen_array holds sensitivity coefficents once loaded         */
/*          array homog_data holds homogeneous reading once loaded           */
/*  Structures needed: none                                            */
/*  Functions required: menu_window, button2 and pause                     */
/*******************************************************************/
void load_cal(int num_pix)
{
    struct viewporttype vp;
    int xloc,yloc,m,n,x,file_number = 1;
    char str1[35];
    char file_name[15];
    char *path;
    FILE *file_p;

    settextjustify( CENTER_TEXT, CENTER_TEXT ); //set test attibutes
    settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

    vp = menu_window(10,60,460,339);        //draw screen window

    setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);       //draw message window
    bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

    button_2(260,16,32,"Calibration File Number To Load",0,&file_number,1,100);

    itoa(file_number, str1, 10 );              // build up file name
    strcpy(file_name, "EIT");
    strcat(file_name, str1);
    strcat(file_name, ".CAL");

    xloc = (vp.right - vp.left)/2;         //calculate centre of screen
    yloc = (vp.bottom-vp.top)/2;

    path = searchpath(file_name);
    //if file exists
    if ( path != NULL)
    {
        //if can't open file print error
        if((file_p = fopen(file_name,"rt+")) == NULL)
        {
            strcpy(str1, "Error can't open ");
            strcat(str1, file_name);
            outtextxy(xloc, yloc, str1);
        }
        //else load sensitivity coefficents and homogeneous readings
        else
        {
            for(x = 0; x < num_pix; x++)
            {
                for(m = 0; m < max_m ; m++)
                {
                    for(n = 0; n < max_n; n++)
                    {
                        fscanf(file_p,"%d\t",&sen_array[x][m][n]);
                    }
                    fscanf(file_p,"\n");
                }
                fscanf(file_p,"\n");
            }
            outtextxy(xloc,yloc,"Calibration File Loaded");
            fclose(file_p);
        }
    }
    // else print "file name"dose not exist
    else
    {
        strcpy(str1, file_name);
        strcat(str1, " Does Not Exist");
        outtextxy(xloc,yloc,str1);
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);              //close screen window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}
```

```
/*****************************************************************/
/*    Menu1: Sets up main system menu at right hand side of screen the        */
/* coordinates are for a 640*350 screen. The function call the appropriate     */
/* function for the menu item chosen.                                          */
/*  Variables passed to function: none                                         */
/*  Variables passed back by function: none                                    */
/*  Include files needed: none                                                 */
/*  Global variables: none                                                     */
/*  Structures needed: none                                                    */
/*  Functions required: menu                                                   */
/*****************************************************************/
void menu1(void)
{
        int option = 1;
        char *menu_text[16];

        menu_text[0]  = "Reconstruction R";
        menu_text[1]  = "Reconstruction S";
        menu_text[2]  = "Configure System";
        menu_text[3]  = "Display Config";
        menu_text[4]  = "Save Config";
        menu_text[5]  = "Load Config";
        menu_text[6]  = "Acquire Data(1)";
        menu_text[7]  = "Acquire Data(30)";
        menu_text[8]  = "Acquire Data(Time)";
        menu_text[9]  = "Display Data";
        menu_text[10]  = "Save Data";
        menu_text[11] = "Load Data";
        menu_text[12] = "Quit system";
        do
        {
                option = menu(470,55,633,344,13,option,menu_text);
                switch( option )
                {
                        case 1:         recon_r();
                                        break;
                        case 2:         recon_s();
                                        break;
                        case 3:         config_sys(0);
                                        break;
                        case 4:         config_sys(1);
                                        break;
                        case 5:         save_config();
                                        break;
                        case 6:         load_config();
                                        break;
                        case 7:         addr_trans();
                                        acquire_data(1,1);
                                        break;
                        case 8:         addr_trans();
                                        acquire_data(30,1);
                                        break;
```

```
                    case 9:     addr_trans();
                                acquire_time();
                                break;
                    case 10:display_data();
                                break;
                    case 11:save_data();
                                break;
                    case 12:load_data();
                                break;
                    default:break;
                }
            }
            while(option != 13 );
        }

/******************************************************************/
/*    DISPLAY_DATA: Display voltage readings  as a series of bar graphs      */
/*  Variables passed to function: none                                      */
/*  Variables passed back by function: none                                 */
/*  Include files needed: graphics.h and stdlib.h
        */
/*  Global variables:                                                       */
/*          array data hold voltage readings to be displayed               */
/*  Structures needed: none                                                 */
/*  Functions required: menu_window,button_2 and pause                      */
/******************************************************************/
void display_data(void)
{
    struct viewporttype vp;
    int m,i;
    int key = 0;
    char str[10];

    settextjustify(CENTER_TEXT, CENTER_TEXT);   //set text attributes
    settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
    vp = menu_window(10,60,460,339);           //open window
    m = 0;
    do
    {
        m++;
        // display projection number
        button_2(150,20, 20,"Projection Number", 1, &m, 1, 1);

        setcolor(EGA_BLACK);
        line(35,48,35,223);     //draw left axes
        line(35,218,419,218);//draw bottom axes
        //draw left axes scale and tick marks
        setlinestyle(DASHED_LINE, 0xFFFF, NORM_WIDTH);
        for(i = 0 ; i <= 10; i++)
        {
            itoa(i*2.5,str,10);
            outtextxy(20, 218-17*i, str);
            line(25, 218-17*i, 35, 218-17*i);
        }
        //draw bottom axes scale and tick marks
        for(i = 1 ; i <= max_n; i++)
        {
            if((i%2 == 0))
            {
                itoa(i,str,10);
                outtextxy(35+16*i,230 , str);
            }
            line(35+16*i, 218, 35+16*i,223);
        }
        //draw bar graph
        for(i = 0 ; i < max_n; i++)
        {
            setlinestyle (SOLID_LINE,0xffff,NORM_WIDTH);
            setfillstyle(SOLID_FILL,EGA_LIGHTGREEN);
            bar3d(46+16*i, 218 , 56+16*i,48, 0, 0);
            if(data[m-1][i] > 0.00125)
            {
                setfillstyle(SOLID_FILL,EGA_LIGHTRED);
                if(data[m-1][i] > 40)
                    bar3d(46+16*i, 218 , 56+16*i,48, 0, 0);
                 else
                    bar3d(46+16*i, 218 , 56+16*i,
                        218-6.8*data[m-1][i], 0, 0);
            }
        }
        if (m < max_m)
            key=pause("Press Esc To Quit, Any Other Key For Next
                        Projection");
        else
            key=pause("Press any key to continue");
    }
    while((m < max_m)&&(key != ESC));
    setfillstyle( SOLID_FILL,EGA_BLUE );        //close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}
```

```
/*****************************************************************/
/*    ACQUIRE DATA: Acquires data in the sequence given by system config.    */
/* Variables passed to function:                                      */
/*    ave gives the number of samples over which averaging should be done
      */
/*    display gives display status required if 1 then aquire data is active
      */
/* Variables passed back by function: none                                */
/* Include files needed: none                                             */
/* Global variables:                                                      */
/*          array data hold the data acquired by system                   */
/*          structure addr holds the addressing sequence for system hardware    */
/*    structure config holds data acquisition sequence                */
/*          max_m, max_n                                                  */
/* Structures needed: none                                                */
/* Functions required:                                                    */
/*          compile_address, menu_window and menu_button and pause        */
/*****************************************************************/
void acquire_data(int ave, int display)
{
    struct viewporttype vp;
    int high_byte,low_byte,m,n,x;
    int gain,status,result;
    float sum1[max_m][max_n];
    int actual_gain[] = {106,216,309,423};
    int base = 512;

    //clear data arrays
    for(m = 0; m < max_m; m++)
    {
        for(n=0; n < max_n; n++)
        {
            sum1[m][n] = 0;
            data[m][n] = 0;
        }
    }
    outportb(base + 11,  0x80);          /* Set Port2 A,B,C, as output */
    outportb(base + 12,0x00);            /* Select analogue channel 1  */
    if(display == 1)
    {
        vp = menu_window(10,60,460,339);          //open window
        menu_button(250,100,50,EGA_CYAN,EGA_CYAN,
                        "Please Wait Acquiring Data" ,1);
            setcolor(EGA_BLACK);
            line(109,100,341,100);
            line(109,100,109,200);
            setcolor(EGA_WHITE);
            line(109,200,341,200);
            line(341,100,341,200);
    }
    for(x=0; x < ave; x++)
    {
        for(m = 0; m < max_m; m++)
        {
            if(addr.source[m] != 0)
            {
                outportb(base + 10 ,addr.source[m]);//set source electrodes
                for(n=0; n < max_n; n++)
                {
                    if(display == 1)
                    {
                        if(m == 0)
                        {
                            setfillstyle(SOLID_FILL,EGA_LIGHTRED);
                            bar(110,101,340,199);
                        }
                        setfillstyle(SOLID_FILL,EGA_LIGHTGREEN);
                        bar(110,101,110+10*m,199);
                    }
                    result = 0;
                    if(addr.read[m][n] != 0)
                    {   //set reading electrodes
                        outportb(base+9, addr.read[m][n]);
                        gain = 500;
                        do
                        {
                            gain = gain - 100;
                            addr.gain_freq = comp_addr(
                                (pow(2,gain/100-1)((config.freq/5)-1));
                            outportb(base+8, addr.gain_freq);
                            delay(10);
                            low_byte = inp(base + 2);  // Start conversion
                            status = inp(base);
                            while((status & 0x80) != 0x80 )
                                status = inp(base);
                            high_byte = inp(base+1);       //read high byte
                            low_byte  = inp(base+2);       //read low byte
                            result = (high_byte*256)+low_byte;
                        }while((result > 3600)&&(gain > 100));
                        sum1[m][n]= sum1[m][n]+
                            (((float)(result)/400)
                                /actual_gain[(gain/100)-1]);
                    }
                }
            }
        }
    }
// store data
    for(m = 0; m < max_m; m++)
        for(n=0; n < max_n; n++)
            data[m][n] =200*(sum1[m][n]/ave);
    if(display == 1)
    {
        sound(1500);
        delay(100);
        nosound();
        pause("Press any key to continue");
        setfillstyle(SOLID_FILL,EGA_BLUE);
        bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
    }
}

/*******************************************************************/
/*    ADDR TRANS :Traslates configuration of system in to hardware addresses  */
/* for data  acquisition.                                               */
/*  Variables passed to function: none                                  */
/*  Variables passed back by function: none                             */
/*  Include files needed: none                                          */
/*  Global variables:                                                   */
/*    structure addr holds the addressing sequence generated for      */
/*    system hardware                                                   */
/*    structure config holds data acquisition sequence                */
/*    max_m, max_n                                                    */
/*  Structures needed: none                                             */
/*  Functions required: none                                            */
/*******************************************************************/
void addr_trans(void)
{
    int m,n;

    // compute gain and frequency address
    addr.gain_freq = comp_addr(0,(config.freq/5)-1);

    for(m = 0; m < max_m; m++)
    {
        //compute source address
        addr.source[m] = comp_addr(config.source[m][0]-1,
                                   config.source[m][1]-1);
        for(n = 0; n < (2*max_n) ; n = n + 2)
            //compute readings address
            addr.read[m][n/2] = comp_addr(config.read[m][n]-1,
                                    config.read[m][n+1]-1);
    }
}
```

```
/*****************************************************************/
/*    COMP ADDR : Compile an 8 bit address from two 4 bit values          */
/* Variables passed to function:                                          */
/*          addr_high holds top 4 bits of address                         */
/*          addr_low holds bottom 4 bits of address                       */
/* Variables passed back by function:                                     */
/*          8 bit address return by function                              */
/* Include files needed: none                                             */
/* Global variables: none                                                 */
/* Structures needed: none                                                */
/* Functions required: none                                               */
/*****************************************************************/
unsigned char comp_addr(int addr_high, int addr_low)
{
      int temp1_addr,temp2_addr;

      if (((addr_high < 17) && (addr_low < 17))&&
            ((addr_high >= 0) && (addr_low >= 0)))
      {
            temp1_addr = addr_high<<4;
            temp1_addr = temp1_addr & 0xf0;
            temp2_addr = addr_low & 0x0f;
            temp1_addr = temp1_addr + temp2_addr;
            return(temp1_addr);
      }
      else
      {
            return(0);
      }
}

/*****************************************************************/
/*    ACQUIRE_TIME:Acquire data at time intervals given continuously for given*/
/* number of samples                                                      */
/* Variables passed to function: none                                     */
/* Variables passed back by function none                                 */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h */
/* Global variables:                                                      */
/*          data holds data acquired                                      */
/*          structure config                                              */
/* Structures needed: none                                                */
/* Functions required: menu_window, button2 acquire_data and pause
      */
/*****************************************************************/
void acquire_time(void)
{
      struct viewporttype vp;
      int xloc,yloc,m,n;
      int time_delay = 1,int samples= 1,file_number = 0;
      char str1[35];
      char file_name[15];
      FILE *file_p;
```

```
settextjustify( CENTER_TEXT, CENTER_TEXT ); //set text attributes
settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );
vp = menu_window(10,60,460,339);      //draw screen window
xloc = (vp.right - vp.left)/2;     //calculate centre of window
yloc = (vp.bottom-vp.top)/2;
setfillstyle(SOLID_FILL,EGA_LIGHTBLUE); //draw message window
bar3d(16,66,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
button_2(220,16,32,"Enter Time Delay In Seconds",0,&time_delay,1,60);
button_2(220,16,48,"Enter Number Of Samples    ",0,&samples,1,60);
while(samples >= 1)
{
    acquire_data(1,0);
    file_number++;
    itoa(file_number, str1, 10 );          //build up file name
    strcpy(file_name, "Creep");
    strcat(file_name, str1);
    strcat(file_name, ".Dat");
    //if can't open file print error message
    if ((file_p = fopen(file_name,"wt+")) == NULL)
    {
      strcpy(str1,"Error can't open ");
      strcat(str1,file_name);
      outtextxy(xloc, yloc, str1);
      samples = 0;
    }
    //else save data to disk
    else
    {
        for(m = 0; m < max_m ; m++)
        {
            //if projection active save readings
            if((config.source[m][0] > 0)&&(config.source[m][0]< 17))
            {
                for(n = 0; n < (2*max_n); n = n + 2)
                {
                    //if reading active save reading
                    if((config.read[m][n] > 0)&&
                      (config.read[m][n] < 17))
                        fprintf(file_p,"%5.3f\t",data[m][n/2]);
                    else
                        data[m][n/2] = 0;
                }
                fprintf(file_p,"\n");
            }
        }
        setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
        bar3d(16,66,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
        strcpy(str1,"Data File ");
        strcat(str1,file_name);
        strcat(str1," Saved");
        outtextxy(xloc,yloc,str1);
```

```
                fclose(file_p);
            }
            samples--;
            if(samples >= 1)
            {
                for(m =0; m < time_delay; m++)
                    delay(1000);
            }
        }
        pause("Press Any Key To Continue");
        setfillstyle(SOLID_FILL,EGA_BLUE);          //close window
        bar(0, 0, vp.right-vp.left,vp.bottom-vp.top);
}

/****************************************************************/
/*    LOAD REF DATA: Load configurations file from disk.          */
/* Variables passed to function: none                             */
/* Variables passed back by function none                         */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h  */
/* Global variables:                                              */
/*          data used to hold loaded file data.                   */
/*          structure config                                      */
/* Structures needed: none                                        */
/* Functions required: menu_window,button2 and pause             */
/****************************************************************/
void load_ref_data(void)
{
        struct viewporttype vp;
        int xloc,yloc,m,n,file_number = 1;
        char str1[35];
        char file_name[15];
        char *path;
        FILE *file_p;

        settextjustify( CENTER_TEXT, CENTER_TEXT ); //set test attibutes
        settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );
        vp = menu_window(10,60,460,339);        //draw screen window
        setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);      //draw message window
        bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
        button_2(220,16,32,"Ref. File Number To Load",0,&file_number,1,100);
        itoa(file_number, str1, 10 );              // build up file name
        strcpy(file_name, "EIT");
        strcat(file_name, str1);
        strcat(file_name, ".DAT");
        xloc = (vp.right - vp.left)/2;          //calculate centre of screen
        yloc = (vp.bottom-vp.top)/2;

        path = searchpath(file_name);
        //if file exists
        if ( path != NULL)
        {
            //if can't open file print error
```

```
          if ((file_p = fopen(file_name,"rt+")) == NULL)
          {
                strcpy(str1, "Error can't open ");
                strcat(str1, file_name);
                outtextxy(xloc, yloc, str1);
          }
          else          //else load data file
          {
                for(m = 0; m < max_m ; m++)
                {
                      //if projection active load readings
                      if((config.source[m][0] > 0)&&(config.source[m][0]< 17))
                      {
                            for(n = 0; n < (2*max_n); n = n + 2)
                            {
                                  //if reading active load reading
                                  if((config.read[m][n] > 0)
                                              &&(config.read[m][n] < 17))
                                        fscanf(file_p,"%f\t",&homog_data[m][n/2]);
                                  else
                                        homog_data[m][n/2] = 0;
                            }
                            fscanf(file_p,"\n");
                      }
                }
                outtextxy(xloc,yloc,"Ref. Data File Loaded");
                fclose(file_p);
          }
    }
    // else print "file name"dose not exist
    else
    {
          strcpy(str1, file_name);
          strcat(str1, " Does Not Exist");
          outtextxy(xloc,yloc,str1);
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);          //close screen window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}
```

```
/*****************************************************************/
/*    SAVE DATA: Save data to disk.                              */
/* Variables passed to function: none                            */
/* Variables passed back by function none                        */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h  */
/* Global variables:                                             */
/*          data holds data to be saved                          */
/*          structure config                                     */
/* Structures needed: none                                       */
/* Functions required: menu_window, button2 and pause            */
/*****************************************************************/
void save_data(void)
{
     struct viewporttype vp;
     int xloc,yloc,m,n;
     int file_number = 1;
     char cont;
     char str1[35];
     char file_name[15];
     char *path;
     FILE *file_p;

     settextjustify( CENTER_TEXT, CENTER_TEXT ); //set text attributes
     settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

     vp = menu_window(10,60,460,339);          //draw screen window

     setfillstyle(SOLID_FILL,EGA_LIGHTBLUE); //draw message window
     bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

     button_2(220,16,32,"Data File Number To Save",0,&file_number,1,100);

     itoa(file_number, str1, 10 );        //build up file name
     strcpy(file_name, "EIT");
     strcat(file_name, str1);
     strcat(file_name, ".DAT");

     xloc = (vp.right - vp.left)/2;       //calculate centre of window
     yloc = (vp.bottom-vp.top)/2;
     cont = 'Y';
     path = searchpath(file_name);
     //if file exists check to overwrite file
     if(path != NULL)
     {
          strcpy(str1, "File Already Exists Overwrite Y");
          outtextxy(xloc, yloc, str1);   //display check message
          cont = toupper(getch());
     }
     // if ok to over write file
     if(cont == 'Y')
     {
          //clear check message from message window
          setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
          bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
          //if can't open file print error message
          if ((file_p = fopen(file_name,"wt+")) == NULL)
          {
                strcpy(str1,"Error can't open ");
                strcat(str1,file_name);
                outtextxy(xloc, yloc, str1);
          }
          //else save data to disk
          else
          {
                for(m = 0; m < max_m ; m++)
                {
                      //if projection active save readings
                      if((config.source[m][0] > 0)&&(config.source[m][0]< 17))
                      {
                            for(n = 0; n < (2*max_n); n = n + 2)
                            {
                                  //if reading active save reading
                                  if((config.read[m][n] > 0)
                                              &&(config.read[m][n] < 17))
                                        fprintf(file_p,"%5.3f\t",data[m][n/2]);
                                  else
                                        data[m][n/2] = 0;
                            }
                            fprintf(file_p,"\n");
                      }
                }
                outtextxy(xloc,yloc,"Data File Saved");
                fclose(file_p);
          }
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);          //close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top);
}
```

```
/****************************************************************/
/*    LOAD DATA: Load configurations file from disk.            */
/* Variables passed to function: none                           */
/* Variables passed back by function none                       */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h */
/* Global variables:                                            */
/*          data used to hold loaded file data.                 */
/*          structure config                                    */
/* Structures needed: none                                      */
/* Functions required: menu_window,button2 and pause            */
/****************************************************************/
void load_data(void)
{
    struct viewporttype vp;
    int xloc,yloc,m,n;
    int file_number = 1;
    char str1[35];
    char file_name[15];
    char *path;
    FILE *file_p;

    settextjustify( CENTER_TEXT, CENTER_TEXT ); //set test attibutes
    settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

    vp = menu_window(10,60,460,339);        //draw screen window

    setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);     //draw message window
    bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

    button_2(220,16,32,"Data File Number To Load",0,&file_number,1,100);

    itoa(file_number, str1, 10 );               // build up file name
    strcpy(file_name, "EIT");
    strcat(file_name, str1);
    strcat(file_name, ".DAT");

    xloc = (vp.right - vp.left)/2;          //calculate centre of screen
    yloc = (vp.bottom-vp.top)/2;

    path = searchpath(file_name);
    //if file exists
    if ( path != NULL)
    {
        //if can't open file print error
        if ((file_p = fopen(file_name,"rt+")) == NULL)
        {
            strcpy(str1, "Error can't open ");
            strcat(str1, file_name);
            outtextxy(xloc, yloc, str1);
        }
        else    //else load data file
        {
            for(m = 0; m < max_m ; m++)
            {
                //if projection active load readings
                if((config.source[m][0] > 0)&&(config.source[m][0]< 17))
                {
                    for(n = 0; n < (2*max_n); n = n + 2)
                    {
                        //if reading active load reading
                        if((config.read[m][n] > 0)
                                &&(config.read[m][n] < 17))
                            fscanf(file_p,"%f\t",&data[m][n/2]);
                        else
                            data[m][n/2] = 0;
                    }
                    fscanf(file_p,"\n");
                }
            }
            outtextxy(xloc,yloc,"Data File Loaded");
            fclose(file_p);
        }
    }
    // else print "file name"dose not exist
    else
    {
        strcpy(str1, file_name);
        strcat(str1, " Does Not Exist");
        outtextxy(xloc,yloc,str1);
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);          //close screen window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}

/*******************************************************************/
/*  LOAD CONFIG: Load configurations file from disk.
     */
/* Variables passed to function: none                                  */
/* Variables passed back by function none                              */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h */
/* Global variables: structure config used to hold configuration loaded.  */
/* Structures needed: none                                             */
/* Functions required: menu_window,button2 and pause                   */
/*******************************************************************/
void load_config(void)
{
    struct viewporttype vp;
    int xloc,yloc,m,n;
    int file_number = 1;
    char str1[35];
    char file_name[15];
    char *path;
    FILE *file_p;
```

```
settextjustify( CENTER_TEXT, CENTER_TEXT ); //set test attibutes
settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

vp = menu_window(10,60,460,339);        //draw screen window

setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);        //draw message window
bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

button_2(220,16,32,"Configuration File Number",0,&file_number,1,10);

itoa(file_number, str1, 10 );                    // build up file name
strcpy(file_name, "EIT");
strcat(file_name, str1);
strcat(file_name, ".CFG");

xloc = (vp.right - vp.left)/2;          //calculate centre of screen
yloc = (vp.bottom-vp.top)/2;

path = searchpath(file_name);
//if file exists
if ( path != NULL)
{
    //if can't open file print error
    if ((file_p = fopen(file_name,"rt+")) == NULL)
    {
        strcpy(str1, "Error can't open ");
        strcat(str1, file_name);
        outtextxy(xloc, yloc, str1);
    }
    else                        //else load configuration file
    {
        fscanf(file_p, "%d\n",&config.freq);
        for(m = 0; m < max_m ; m++)
        {
            fscanf(file_p,"%d %d",&config.source[m][0],
                 &config.source[m][1]);

            for(n = 0; n < (2*max_n); n++)
            {
                fscanf(file_p,"%d",&config.read[m][n]);
            }
            fscanf(file_p,"\n");
        }
        outtextxy(xloc,yloc,"Configuration File Loaded");
        fclose(file_p);
    }
}
// else print "file name"dose not exist
else
{
    strcpy(str1, file_name);
```

```
        strcat(str1, " Does Not Exist");
        outtextxy(xloc,yloc,str1);
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);          //close screen window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}

/******************************************************************/
/*    SAVE CONFIG: Save configurations to disk.                    */
/* Variables passed to function: none                              */
/* Variables passed back by function none                          */
/* Include files needed: graphic.h, stdio.h, stdlib.h, string.h and dir.h */
/* Global variables: structure config holds configuration to be saved. */
/* Structures needed: none                                         */
/* Functions required: menu_window, button2 and pause              */
/******************************************************************/
void save_config(void)
{
    struct viewporttype vp;
    int xloc,yloc,m,n;
    int file_number = 1;
    char cont;
    char str1[35];
    char file_name[15];
    char *path;
    FILE *file_p;

    settextjustify( CENTER_TEXT, CENTER_TEXT ); //set text attributes
    settextstyle(DEFAULT_FONT,HORIZ_DIR, 1 );

    vp = menu_window(10,60,460,339);            //draw screen window

    setfillstyle(SOLID_FILL,EGA_LIGHTBLUE); //draw message window
    bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);

    button_2(220,16,32,"Configuration File Number",0,&file_number,1,10);

    itoa(file_number, str1, 10 );         //build up file name
    strcpy(file_name, "EIT");
    strcat(file_name, str1);
    strcat(file_name, ".CFG");

    xloc = (vp.right - vp.left)/2;        //calculate centre of window
    yloc = (vp.bottom-vp.top)/2;
    cont = 'Y';
    path = searchpath(file_name);
    //if file exists check to overwrite file
    if(path != NULL)
    {
        strcpy(str1, "File Already Exists Overwrite Y");
        outtextxy(xloc, yloc, str1);    //display check message
        cont = toupper(getch());
    }
    // if ok to over write file
    if(cont == 'Y')
    {
        //clear check message from message window
        setfillstyle(SOLID_FILL,EGA_LIGHTBLUE);
        bar3d(16,56,vp.right-vp.left-16,vp.bottom-vp.top-56,0,0);
        //if can't open file print error message
        if((file_p = fopen(file_name,"wt+")) == NULL)
        {
            strcpy(str1,"Error can't open ");
            strcat(str1,file_name);
            outtextxy(xloc, yloc, str1);
        }
        //else save configuration to disk
        else
        {
            fprintf(file_p, "%d\n",config.freq);
            for(m = 0; m < max_m ; m++)
            {
                fprintf(file_p,"%d  %d   ",config.source[m][0],
                        config.source[m][1]);
                for(n = 0; n < (2*max_n); n++)
                {
                        fprintf(file_p,"%d ",config.read[m][n]);
                }
                fprintf(file_p,"\n");
            }
            outtextxy(xloc,yloc,"Configuration File Saved");
            fclose(file_p);
        }
    }
    pause("Press Any Key To Continue");
    setfillstyle(SOLID_FILL,EGA_BLUE);         //close window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}
```

```
/*********************************************************************/
/*    CONFIG SYS: Sets up configuration window for configuring the systems    */
/*  data acquition system. The components of the window can be either         */
/*  active or passive. Therefore the function can be used to either just
      */
/*  display the configuration(one projection at a time or to actually         */
/*  configure the system.                                                     */
/*  Variables passed to function:                                        */
/*            active  sets the active status of the window(active if 0)       */
/*  Variables passed back by function: none                                   */
/*  Include files needed: graphic.h, stdlib.h and string.h
      */
/*  Global variables:                                                         */
/*            structure config used to hold configuration, constant max_m,    */
/*            max_n and  ESC                                                  */
/*  Structures needed:  none                                                  */
/*  Functions required: menu_window,button_2, button_3 and pause              */
/*********************************************************************/
void config_sys(int active)
{
      struct viewporttype vp;
      char str1[50];
      char str2[20];
      int xloc,yloc,m,n,i,times,int key= 'a';

      settextjustify(CENTER_TEXT, CENTER_TEXT); //set text attributes
      settextstyle(DEFAULT_FONT,HORIZ_DIR, 1);

      vp = menu_window(10,60,460,339);        //draw config screen window

      button_2(136,20,24,"Frequency (KHz)", active, &config.freq, 5, 8);
      if(active == 0)
            key = pause("Press ESC to exit any other key to continue");
      if(key != ESC)
      {
            for(m = 0; m <  max_m; m++)
            {
                  strcpy(str1,"Projection ");
                  itoa(m+1,str2,10);
                  strcat(str1,str2);
                  strcat(str1," Source Electrodes");
                  button_3(260,20,44,str1,active,&config.source[m][0],
                              &config.source[m][1],1,17);
                  yloc = 44;
                  for(n=0; n < max_n; n = n + 3)
                  {
                        yloc=yloc+25;
                        xloc = 20;
                        for(i = 0; i < 3; i++)
                        {
                              strcpy(str1,"VR");
                              itoa((n+i+1),str2,10);
```

```
                    strcat(str1,str2);
                    button_3(45,xloc,yloc,str1,active,
                                 &config.read[m][(n+i)*2],
                                 &config.read[m][((n+i)*2)+1],1,17);
                    xloc = xloc+152;
                }
            }
            if(m < (max_m-1))
                key = pause
                    ("Press ESC to exit any other key for next projection");
            else
                key = pause("Press any key to exit");
            if(key == ESC)
                m = max_m;
        }
    }
    setfillstyle( SOLID_FILL,EGA_BLUE );              //close config window
    bar(0, 0, vp.right-vp.left,vp.bottom-vp.top  );
}

/*****************************************************************/
/*     pause: Display a status line at the bottom of the screen and waits for     */
/*  key stroke from operator and return that key stroke.                           */
/*  Variables passed to function:                                                  */
/*            *message pointer to text displayed in pause button                   */
/*  Variables passed back by function:                                             */
/*            key returns key stroke to continue (bios key codes used)             */
/*  Include files needed: graphic.h and bios.h                                     */
/*  Global variables: none                                                         */
/*  Structures needed: none                                                        */
/*  Functions required: none                                                       */
/*****************************************************************/
int pause(char *message)
{
    struct viewporttype vp;
    struct textsettingstype text;
    int height,width,x_left,x_right;
    int y_top,y_bottom,key;

    gettextsettings(&text);     //save current text settings
    getviewsettings(&vp);       //get current viewport settings

    settextstyle(DEFAULT_FONT,HORIZ_DIR,1); // set text and line attributes
    settextjustify(CENTER_TEXT,CENTER_TEXT);
    setlinestyle(SOLID_LINE, 0, NORM_WIDTH);
    height = textheight("H");                   //determine message height
    width  = 10+textwidth(message);                  //determine message width

    x_left   = ((vp.right-vp.left)-(width))/2; //calculate loction
    y_top    = (vp.bottom-vp.top)-(2*height);  //for pause button
    x_right  =  x_left + width;
    y_bottom = (vp.bottom-vp.top)-(0.5*height);
```

```
        setfillstyle(SOLID_FILL, EGA_LIGHTCYAN);  //draw pause button
        bar(x_left, y_top, x_right,y_bottom);

        setcolor(EGA_BLACK);
        line(x_left, y_top,x_right, y_top);       //draw top border
        line(x_left, y_top, x_left, y_bottom);    //draw left border

        setcolor(EGA_WHITE);
        line(x_left, y_bottom, x_right, y_bottom); //draw bottom border
        line(x_right, y_top, x_right,y_bottom);   //draw right border
        setcolor(EGA_BLACK);
        outtextxy((vp.right-vp.left)/2, y_bottom-(0.6*height) ,message);
        key = bioskey(0);                         //wait for key stroke
        setfillstyle(SOLID_FILL,EGA_LIGHTGRAY);  //clear pause button
        bar(x_left, y_top, x_right,y_bottom);
        settextstyle(text.font,text.direction,text.charsize);
        settextjustify(text.horiz,text.vert);
        return(key);
}

/*************************************************************************/
/*     BUTTON_SELECT: Controls the selection of the menu buttons          */
/*     The selection of a button is controlled by the up/down arrow keys  */
/*  Return maximum number of selected button if escape key is pressed.   */
/*  or currently selected button if carrage return key is pressed         */
/*  Variables passed to function:                                         */
/*            max_button gives the maximum number of button selectable    */
/*            current gives the currently selected button number          */
/*  Variables passed back by function:                                    */
/*            button structure passes back the newly selected button and the */
/*        last key stroke(bios key number)                                */
/*  Include files needed: bios.h                                          */
/*  Global variables: constants ESC, UP, DOWN and CR                      */
/*  Structures needed: buttontype                                         */
/*  Functions required: none                                              */
/*************************************************************************/
struct buttontype button_select(int max_buttons, int current)
{
        struct buttontype button;

        button.number = current;
        do
        {
                while(bioskey(1) == 0);
                button.select = bioskey(0);
        }while((button.select != UP)&&(button.select != DOWN)&&
                 (button.select != ESC)&&(button.select != CR));

        switch(button.select)
        {
                case UP:    if(button.number > 1)
```

```
                        button.number--;
                    else
                        button.number=max_buttons;
                    break;
        case DOWN:  if(button.number < max_buttons)
                        button.number++;
                    else
                        button.number = 1;
                    break;
        case ESC:  button.number = max_buttons;
                    break;
        default:    break;
    }
    return(button);
}

/*****************************************************************/
/*   BUTTON2_SELECT: Controls the selection of the contents button in    */
/* button types 2 and 3.                                                  */
/*   The increment or decrement of the contents is controlled by the     */
/* up/down arrow keys                                                     */
/* The contents is return if the carrage return key is pressed           */
/* Variables passed to function:                                          */
/*         max_button gives the maximum number of button selectable       */
/*         current gives the current contents of the button               */
/* Variables passed back by function:                                     */
/*         button structure passes back the newly contents and the        */
/*       last key stroke(bios key number)                                 */
/* Include files needed: bios.h                                           */
/* Global variables: constants ESC, UP, DOWN and CR                       */
/* Structures needed: buttontype                                          */
/* Functions required: none                                               */
/*****************************************************************/
struct buttontype button2_select(int max_contents, int current)
{
    struct buttontype button;

    button.number = current;
    do
    {
        button.select = bioskey(0);
    }while((button.select != UP)&&(button.select != DOWN)&&
          (button.select != ESC)&&(button.select != CR));

    switch(button.select)
    {
        case UP:    if(button.number < max_contents)
                        button.number++;
                    else
                        button.number= 1;
                    break;
        case DOWN:  if(button.number > 1)
```

```
                                        button.number--;
                                else
                                        button.number = max_contents;
                                break;
                default:        break;
        }
        return(button);
}

/*****************************************************************/
/* MENU: Generates menu system and returns the number of menu item              */
/* selected when either the carrage return or escape key is pressed             */
/* maximum number of button available is 16                                     */
/* Variables passed to function:                                                */
/*      left,top,right and bottom give the screen coordinates of the menu window. */
/*      no_button sets the number of menu buttons                               */
/*      num gives the number of the menu button currently active                */
/*      *str1 an array of pointers pointing to teh text displayed in each button. */
/* Variables passed back by function: none                                      */
/*      return the number of the selected menu function to run.                 */
/* Include files needed: graphics.h                                             */
/* Global variables: constants CR and ESC                                       */
/*  Structures needed: buttontype                                               */
/* Functions required: menu_window,menu_button and button_select                */
/*****************************************************************/
int menu(int left, int top, int right,int bottom, int no_buttons, int num, char *str1[16])
{
        struct viewporttype vp;
        struct buttontype b_select;
        int x_max,i,a1,a2,a3;
        int xloc = 5;
        int yloc = 10;

        vp =  menu_window(left,top,right,bottom);//Draw menu background window
        x_max = (vp.right-vp.left); //Calculate maximum x coordinates for window
        // Draw all menu buttons buttons active
        for(i = 0; i < no_buttons; i++)
        {
                if((i+1) == num)
                        menu_button(x_max-
8,xloc,yloc,EGA_LIGHTGREEN,EGA_GREEN,str1[i],0);
                else
                        menu_button(x_max-
8,xloc,yloc,EGA_LIGHTGREEN,EGA_GREEN,str1[i],1);
                yloc = yloc + 18;
        }
        b_select.number = num;   //set button selected to previous button
        do
        {
                b_select = button_select( no_buttons, b_select.number);
                if(b_select.number == 1)
                {
```

```
            a1 = no_buttons -1;
            a2 = 0;
            a3 = 1;
        }
        else
        {
            if(b_select.number == no_buttons)
            {
                a1 = 0;
                a2 = no_buttons - 1;
                a3 = no_buttons - 2;
            }
            else
            {
                a1 = b_select.number -2;
                a2 = b_select.number -1;
                a3 = b_select.number;
            }
        }
        menu_button(x_max-8,xloc,10+(a1*18),EGA_LIGHTGREEN,
                    EGA_GREEN,str1[a1],1);
        menu_button(x_max-8,xloc,10+(a2*18),EGA_LIGHTGREEN,
                    EGA_GREEN,str1[a2],0);
        menu_button(x_max-8,xloc,10+(a3*18),EGA_LIGHTGREEN,
                    EGA_GREEN,str1[a3],1);
    }
    while((b_select.select != CR)&&(b_select.select != ESC));
    return(b_select.number); //return menu item selected
}
```

```
/*******************************************************************/
/*     BUTTON_2: Draw menu button type two. Has two components a label       */
/* button which is passive and a selection button to the right of the label button     */
/* which can  be either active or passive.                                              */
/*  Variables passed to function:                                              */
/*     lwidth gives width of label button in pixels.                                    */
/*           xloc,yloc give coordinates of buttons top left hand corner.                */
/*     *label pointer to label text to be displayed                                     */
/*     active gives the active status of the button(active if 0)                        */
/*     *contents pointer to variable in selection button                       */
/*           mult multiplication factor for when variable in selection button          */
/*        is increased.                                                                 */
/*     no_items give maximum range of increases for selection button                   */
/*  Variables passed back by function                                                   */
/*     *contents pointer to vaiable in selection button                                 */
/*  Include files needed: graphics.h and stdlib.h
        */
/*  Global variables: constant CR                                                       */
/*  Structures needed: buttontype,                                                      */
/*  Functions required: menu_button and button2_select                                  */
/*******************************************************************/
void button_2(int lwidth, int xloc, int yloc, char *label,
                        int active, int *contents, int mult, int no_items)
{
      struct buttontype b_select;
      char string[50];
      settextjustify(CENTER_TEXT,CENTER_TEXT); //set text attributes
      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
      //draw label button
      menu_button(lwidth,xloc,yloc,EGA_LIGHTCYAN,EGA_CYAN,label,-1);

      itoa(*contents,string,10);
      if(active == 0)  //if selection button active
      {
            b_select.number = *contents/mult; //calculate selection number
            do
            {
                  itoa(*contents,string,10);
                  // draw active selection button
                  menu_button(32,xloc+lwidth+6,yloc,
                                    EGA_LIGHTGREEN,EGA_GREEN,string,0);
                  b_select = button2_select(no_items,b_select.number);
                  //calculate next selection
                  *contents = b_select.number*mult;
            }
            while(b_select.select != CR);
      }
      menu_button(32,xloc+lwidth+6,yloc,EGA_LIGHTGREEN,
                  EGA_GREEN,string,1); //draw passive selection button
}

/*******************************************************************/
```

```
/*    BUTTON_3: Draw menu button type two. Has three component a label        */
/*  button which is passive and 2 selection buttons to the right of the       */
/*  label button which can be either active or passive.                       */
/*  Variables passed to function:                                         */
/*    lwidth gives width of label button in pixels.                           */
/*    xloc,yloc give coordinates of buttons top left hand corner.             */
/*    *label pointer to label text to be displayed                            */
/*    active gives the active status of the button(active if 0)               */
/*    *value1 pointer to variable in selection button 1                   */
/*    *value2 pointer to variable in selection button 2                   */
/*    mult multiplication factor for when variable in selection button is increased. */
/*    no_items give maximum range of increases for selection button          */
/*  Variables passed back by function                                         */
/*    *value1 pointer to vaiable in selection button 1                        */
/*    *value2 pointer to vaiable in selection button 2                        */
/*  Include files needed: graphics.h and stdlib.h
      */
/*  Global variables: constant CR                                             */
/*  Structures needed: buttontype,button3type                                 */
/*  Functions required: menu_button and button2_select                        */
/****************************************************************************/
void button_3(int lwidth, int xloc, int yloc, char *label, int active, int *value1, int
*value2,int mult,
                  int no_items)
{
      struct buttontype b_select;
      struct button3type b3_select;
      char str1[50];
      char str2[50];

      settextjustify(CENTER_TEXT,CENTER_TEXT);    //set text attibutes
      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

      //draw label button
      menu_button(lwidth,xloc,yloc,EGA_LIGHTCYAN,EGA_CYAN,label,1);

      itoa(*value1,str1,10);
      itoa(*value2,str2,10);
      //draw passive selection buttons
      menu_button(24,xloc+lwidth+6,yloc,EGA_LIGHTGREEN,EGA_GREEN,str1,1)
;
      menu_button(24,xloc+lwidth+36,yloc,EGA_LIGHTGREEN,
                        EGA_GREEN,str2,1);
      if(active == 0)
      {
            b_select.number = *value1/mult;   //calculate selection number for
            do                          //selection button 1
            {
                  //draw active selection button 1
                  menu_button(24,xloc+lwidth+6,yloc,
                                    EGA_LIGHTGREEN,EGA_GREEN,str1,0);
                  b_select = button2_select(no_items, b_select.number);
```

```
                    *value1 = b_select.number*mult;
                    itoa(*value1,str1,10); //calculate next selection number
               }
               while(b_select.select != CR);
               //draw passive selection button 1
               menu_button(24,xloc+lwidth+6,yloc,EGA_LIGHTGREEN,
                                   EGA_GREEN,str1,1);
               b_select.number = *value2/mult; //calculate selection number for
               do                        //selection button 2
               {
                    //draw active selection button 2
                    menu_button(24,xloc+lwidth+36,yloc,
                                   EGA_LIGHTGREEN,EGA_GREEN,str2,0);
                    b_select = button2_select(no_items, b_select.number);
                    *value2 = b_select.number*mult;
                    itoa(*value2,str2,10); //calculate next selection number
               }
               while(b_select.select != CR);
               //draw passive selection button 2
               menu_button(24,xloc+lwidth+36,yloc,EGA_LIGHTGREEN,
                                   EGA_GREEN,str2,1);
          }
     }

/*******************************************************************/
/*     MENU_BUTTON: Draws menu button which can be either active or passive.*/
/* Variables passed to function:                                  */
/*     lwidth gives width of label button in pixels.                    */
/*          xloc,yloc give coordinates of buttons top left hand corner.  */
/*          acolour gives active button colour                          */
/*          dcolour gives deactive button colour                        */
/*          status gives active status of the button (status = 0 when active)   */
/* Variables passed back by function none                               */
/* Include files needed: graphics.h and stdlib.h
     */
/* Global variables: constant CR                                         */
/* Structures needed: buttontype,                                        */
/* Functions required: none                                              */
/*******************************************************************/
void menu_button(int lwidth, int xloc, int yloc,int acolour,int dcolour,
                    char *string,int status )
{
     int colour = getcolor();
     if (status == 0)
     {
          setfillstyle(SOLID_FILL, acolour);           //draw active button
          bar(xloc, yloc, xloc+lwidth, yloc+12);

          setcolor(EGA_BLACK);
          line(xloc, yloc, xloc+lwidth, yloc);       //draw top border
          line(xloc, yloc, xloc, yloc+12);           //draw left border
```

```
		setcolor(EGA_WHITE);
		line(xloc+lwidth, yloc, xloc+lwidth, yloc+12); //draw right border
		line(xloc, yloc+12, xloc+lwidth, yloc+12); //draw bottom border
	}
	else
	{
		setfillstyle( SOLID_FILL, dcolour);	//draw deactive button
		bar(xloc, yloc, xloc+lwidth, yloc + 12);
		setcolor( EGA_WHITE );
		line(xloc, yloc, xloc+lwidth, yloc);	//draw top border
		line(xloc, yloc, xloc, yloc+12);		//draw left border
		setcolor(EGA_BLACK);
		line(xloc+lwidth, yloc, xloc+lwidth, yloc+12); //draw right border
		line(xloc, yloc+12, xloc+lwidth, yloc+12);	//draw bottom border
	}
	setcolor(EGA_BLACK);
	settextjustify(CENTER_TEXT, CENTER_TEXT);		//set text attributes
outtextxy(xloc+lwidth/2, yloc+7, string);		//print button text
	setcolor(colour);
}

/*******************************************************************/
/* MENU_WINDOW:  Sets up background for system menus. All coordinates are */
/* absolute to the a screen size of 640*350.				*/
/* Variable passed to function:						*/
/*		left, top,right,bottom define screen coordinates of window	*/
/* Variable passed back by function:					*/
/*		viewport set by MENU_WINDOW return via structure viewporttype	*/
/* Include files needed:							*/
/*		stdio.h,conio.h,stdlib.h and graphics.h			*/
/* Global variables: none							*/
/* Structures needed: none							*/
/* Functions:									*/
/*******************************************************************/
struct viewporttype menu_window(int left, int top, int right,int bottom)
{
	struct viewporttype vp;
	int max_x,max_y;

	setviewport(left, top, right, bottom, 1);	// sets screen area for menu
	getviewsettings( &vp );			// get menu window settings
	max_x = vp.right-vp.left;			// calulate maximum screen
	max_y = vp.bottom-vp.top;			//coordinates

	// set window background colour and pattern
	setfillstyle( SOLID_FILL,EGA_LIGHTGRAY );
	setcolor( EGA_WHITE );			// set boarder colour
	bar3d(0, 0, max_x, max_y, 0, 0);		// draw menu background
	return(vp);					// return menu window setting
}

/*******************************************************************/
```

```
/* MAIN_WINDOW:  Sets up main graphic screen system menus. All coordinates
        */
/* are absolute to the a screen size of 640*350.                           */
/* Variable passed to function:                                            */
/*            left, top,right,bottom define screen coordinates of window   */
/* Variable passed back by function:                                       */
/*            viewport set by MENU_WINDOW return via structure viewporttype */
/* Include files needed:                                                   */
/*            stdio.h,conio.h,stdlib.h and graphics.h                      */
/* Global variables: none                                                  */
/* Structures needed: none                                                 */
/* Functions: none                                                         */
/*****************************************************************/
void main_window(void)
{
     setfillstyle(SOLID_FILL,EGA_BLUE);          // draw full screen area as solid
     bar3d(0,0,639,349,0,0);                    // blue with white border.

     setfillstyle(CLOSE_DOT_FILL,EGA_BLUE);
     bar3d(3,3,636,50,0,0); // draw title box as close blue dot  with white border.


     settextstyle(DEFAULT_FONT,HORIZ_DIR,1); // display title text in title area
     outtextxy(95,15, "Sheffield Hallam University, School Of Engineering I. T.");
     outtextxy(167,35,"Electrical Impedance Tomography System");

     rectangle(3,53,636,346);           // draw white border round main working area
}
```

```
/*****************************************************************/
/*    INITIALISE GRAPHICS:  Initialises EGA 640*350 graphics mode using a  */
/* registered EGAVGA driver. The driver should be link in by including in project  */
/* file list,for further information see util.doc. Any error which occur during       */
/* initialisation are echoed to screen and program execution is terminated.         */
/* Variable passed to function: none                                                  */
/* Variable passed back by function: none                                             */
/* Include files needed stdio.h,conio.h,stdlib.h and graphics.h                       */
/* Global variables: none                                                             */
/* Structures needed: none                                                            */
/* Functions: none                                                                    */
/*****************************************************************/
void initialize_graphics(void)
{
        int gdriver = VGA;
        int gmode = VGAMED;
        int graphics_error_code;

        registerbgidriver(EGAVGA_driver);          //register EGAVGA driver
        initgraph(&gdriver,&gmode,"");             //initailize graphics in
                                               //EGA 640*350 mode
        graphics_error_code = graphresult();       //read result of initialization
        if (graphics_error_code != grOk)
        {
                //echo error messages to screen
                printf("Graphics system error: %s\n",grapherrormsg(graphics_error_code));
                exit(1);
        }
}
```