THREE DIMENSIONAL INDUCED CURRENT MAGNETIC RESONANCE - ELECTRIC IMPEDANCE TOMOGRAPHY

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL - ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Levent Özparlak

June, 2004

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Yusuf Ziya İder(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ayhan Altıntaş

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Vakur B. Ertürk

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute Engineering and Science

ABSTRACT

THREE DIMENSIONAL INDUCED CURRENT MAGNETIC RESONANCE - ELECTRIC IMPEDANCE TOMOGRAPHY

Levent Özparlak
M.S. in Electrical - Electronics Engineering
Supervisor: Prof. Dr. Yusuf Ziya İder
June, 2004

Human body contains different tissue types and these different tissue types have different electrical resistivity characteristics. The property, that human body has a large resistivity contrast among its different tissues, introduces us to impedance imaging techniques. In this thesis, we try to find better reconstruction techniques to visualize impedance characteristic of human body. In addition to better imaging, we try to obtain higher resolution images of three dimensional objects than existing imaging techniques. In order to achieve this goal, we proposed a technique named Induced Current Magnetic Resonance - Electric Impedance Tomography (IC MR-EIT) which combines the peripheral voltage measurements of Induced Current Electric Impedance Tomography technique with magnetic flux density measurements acquired using a Magnetic-Resonance Image (MRI) Scanner. The advantage of this method is measurement noise due to electrode will not occur. In this thesis, A new reconstruction algorithm for IC MR-EIT is proposed and computer simulations are made. The proposed algorithm is based on the measurement of one component of the magnetic flux density created by the eddy currents in the object. As can be seen in the results, the reconstruction of the conductivity distribution in three dimensional objects on tomographic planes is made successfully. It also works fine against to the measurement noise up to an acceptable level.

Keywords: Induced Current Magnetic Resonance - Electric Impedance Tomography, Magnetic Resonance Imaging, Impedance Imaging, Image Reconstruction, Finite Element Method.

ÖZET

ÜÇ BOYUTLU NESNELER İÇİN İNDÜKLENEN AKIMLAR İLE MANYETİK REZONANS - ELEKTRİKSEL EMPEDANS GÖRÜNTÜLEME

Levent Özparlak
Elektrik - Elektronik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Yusuf Ziya İder
Haziran, 2004

İnsan vücudu çeşitli yapılardan oluşur ve bu çeşitli vücut yapıları değişik elektriksel direnç karakteristiklerine sahiptir. İnsan vücudunun farklı yapıları arasında büyük bir direnç farkının olması özelliği bizi empedans görüntüleme teknikleriyle tanıştırmaktadır. Bu tezde, insan vücudunu daha iyi görselleştiren geriçatma teknikleri bulmaya çalışıyoruz. Daha iyi görüntülemeye ek olarak, varolan görüntüleme tekniklerinin sahip olduğundan daha yüksek çözünürlükte üç boyutlu nesnelerin görüntülerini elde etmeye çalışıyoruz. Bu amacı gerçekleştirmek için, İndüklenen Akımlar ile Manyetik Rezonans - Elektriksel Empedans Tomografi (İA MR-EET) adında bir teknik öneriyoruz. Bu teknik İndüklenen Akımlar ile Elektriksel Empedans Tomografi'nin (İA-EET) gerilim ölçümleriyle Manyetik Rezonans Görüntü (MRG) Tarayıcısı'nın manyetik akı yoğunluğu ölçümlerini birleştirmektedir. Bu yöntemin faydası elektrotların yerleştirilmesinden dolayı oluşan ölçüm hatalarının gözükmemesidir. Bu tezde, İA MR-EET için yeni bir geriçatma algoritması önerilmiş ve bilgisayar simülasyonları yapılmıştır. Önerilen algoritma burgaç akımlarının yarattığı manyetik akı yoğunluğunun ölçümü üzerine kurulmuştur. Sonuçlardan da anlaşılabileceği gibi, üç boyutlu nesnelerin iletkenlik dağılımının tomografik düzlemlerdeki geriçatması başarıyla yapılmıştır. Ayrıca, kabul edilebilir bir ölçüm gürültüsü seviyesine kadar algoritma çalışmaya devam etmektedir.

Anahtar sözcükler: Akım Tabanlı Manyetik Rezonans - Elektriksel Empedans Tomografi, Manyetik Rezonans Görüntüleme, Empedans Görüntüleme, Görüntü Geriçatma, Sonlu Elemanlar Metodu.

To my mother

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A Summary of Previous Studies on MR-EIT

In human body, various tissue types show different electrical resistivity characteristics. Human body has a large resistivity contrast between a wide range of its tissues [1]. In Table 1.1, some approximate resistivity values for important tissue types of human body are given. In addition, the physiological and pathological states of tissues behave as resistivity variations [2, 3, 4]. As a result, diagnostically valuable information about anatomy, physiological processes and pathology of body would be obtained from the reconstruction of resistivity distribution of body.

Electrical Impedance Tomography (EIT) is a novel technique to reconstruct resistivity distribution of body [5]. This technique is technically based on generating a current distribution inside of the body either by injecting with surface electrodes or inducing by coils placed around the body and simultaneously measuring surface potential changes and/or outside magnetic field produced by internal current distribution. The resistivity distribution of body affects the measured field quantities and some reconstruction algorithms extract this information from them. The reconstructed *image* of the resistivity is unique for noise-free complete boundary data [6].

Tissue Type	Resistivity (Ωcm)
Blood Masses	15
Bone	17000
Fat	2000
Heart Fat	2000
Heart Muscle	450
Human Body(average)	460
Left Lung, Right Lung	1325
Liver	600
Gray Matter, White Matter	220
Skeletal Muscle (longitudinal fibers)	300
Skeletal Muscle (transverse fibers)	1500
Skull	17760
Stomach	400

Table 1.1: Typical resistivity values for different tissue types

If the current is generated by induction from a coil and measured quantity is the magnetic field from the outside coils, this special case of EIT is named as *Magnetic Induction Tomography* [7](MIT). The disadvantage of MIT is that the measurements are taken outside the body. Since the measurements are far from the object, obtained data will not exactly represent the object information. In Figure 1.1, MIT scheme is demonstrated with possible induced current path and direction for a cylindrical object.

There are, of course, many other techniques used for tomography imaging. Two examples for them are *Computerized X-Ray Tomography* [8] (CT) and *Positron Emission Tomography* [9] (PET) which have much more expensive hardware and are larger than EIT. Also, they require ionizing radiation. Furthermore, in principle, EIT is capable of producing thousands of images in a second. On the other hand, due to noise and low sensitivity of boundary voltages to inner *conductivity* [1] perturbations, and also due to some practical problems with electrodes which allow only a limited number of surface voltage measurements, EIT can only yield inaccurate low resolution [10] images. As the distance to the surface increases, the sensitivity and resolution decreases [11, 12]. In addition to all of these, another problem is the fixing of electrodes on the body in clinical use of

[1]Conductivity is multiplicative inverse of resistivity

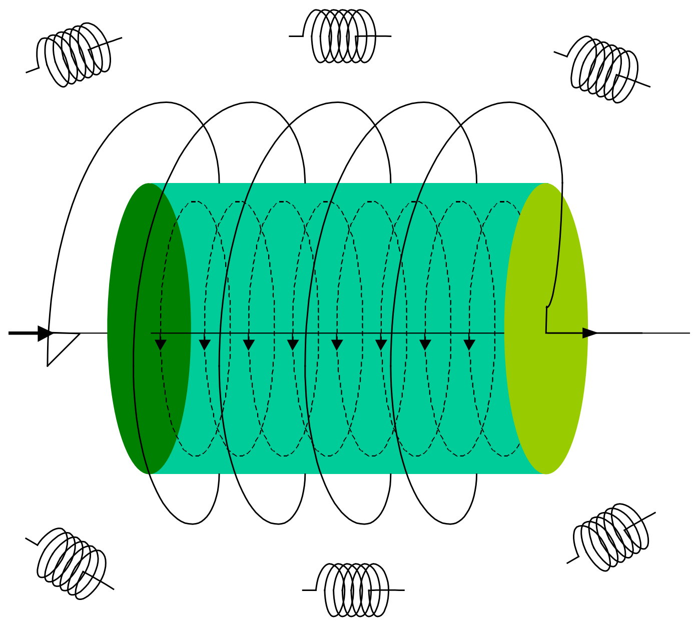


Figure 1.1: MIT imaging scheme for a cylindrical object. The current induction is made by the main coil and surrounding coils are used for magnetic field measurement. Induced circular currents are shown in dashed lines.

EIT.

The problem of position dependency problem of resolution can be solved by using a data set obtained from directly inside of the object. Since it is not possible to measure voltage changes inside of the object, measuring the *magnetic flux density* throughout the imaging region by using a *Magnetic Resonance Imaging* (MRI) system is preferred. This type of measurement can be made with high spatial sampling and with high sensitivity even to inner conductivity perturbations. This reconstruction technique is called as *Magnetic Resonance - Electrical Impedance Tomography* (MR-EIT). In other words, providing high resolution conductivity images is possible by MR-EIT with additional set of measurements from directly inside of the object. An MRI Scanner is used to obtain the distribution of magnetic flux density inside the object due to the internal current density distribution created by either injected currents by surface electrodes or induced currents by coils around the object like EIT. Then, the reconstruction algorithms try to reconstruct the conductivity and/or *current density* images. In this study, instead of injecting currents by surface electrodes, coils induce the currents inside of the object to avoid many difficulties which are mentioned before. Inside current density distribution is dependent on many properties of coils like the size, shape and position data. Currents can be induced in many different ways by either moving the object in the coils or changing the shape of the coils (which is more difficult). A *current induction profile* is defined as the information about where the object positioned around the coil and/or which type of coil is used.

The concept of MR-EIT has been introduced in 1992 by the MSc thesis [13] of N. Zhang. Zhang developed an algorithm, which is capable of reconstructing the correct image using internal current density and also the boundary voltage variation. Method is based on the fact that the potential difference between any two points on the boundary is the integral of *electrical field intensity*, $\mathbf{E}$, which is conservative. Using the point form of Ohm's Law, $\mathbf{E} = \rho\mathbf{J}$, where ρ is the resistivity, and the measured $\mathbf{J}$ is non-conservative, and also for different boundary point pairs, a linear system of equations can be obtained. Solution of this equation set, yields an image, which is unique and correct. The method is valid for 3D reconstructions, as well as for single slice imaging. A drawback of this method is the requirement of many surface voltage measurements to improve the accuracy and resolution of reconstruction.

Eyüboğlu *et.al.* [14], Özdemir and Eyüboğlu [15] and Kwon *et.al.* [16] have proposed algorithms based on constructing the equipotential lines in the object using peripheral voltages and current density distribution. Current density inside the object is measured and it is known that the equipotential lines and current lines are orthogonal. The potential and thus the electrical field distributions inside the object can be found by equipotential lines and projecting the peripheral voltage measurements into the *Field of View* (FoV) along these lines. Using the calculated electric field distribution and measured current density distribution, conductivity can be found for the entire FoV using Ohm's law. These methods, similar to the method of Zhang, are also non-iterative, and require peripheral voltage measurements and a single current injection pattern.

Woo *et.al.* [17] have proposed a reconstruction algorithm whereby the error between the current density measured by MR-CDI technique and the current density calculated by the *Finite Element Method* (FEM) is minimized as a function of the resistivity distribution. Kwon *et.al.* [18] have expanded on this idea to develop the *J-substitution Algorithm*, which uses at least two injected current patterns and a single voltage measurement to reconstruct the correct image. They also claim that at any point in the object, current densities measured for the two current injection patterns must not be parallel. This requirement of at least two injected current patterns is rigorously proven later by Kim *et.al.* [19]. Khang *et.al.* [20] have applied the J-substitution algorithm successfully to data obtained from saline phantoms. Both methods in [17] and [18] are iterative. Another iterative method which is proposed by Eyüboğlu *et.al.* [21], is based on minimizing the error between measured and calculated current densities and peripheral voltages simultaneously. Recently Birgül *et.al.* [22] have proposed another iterative method in which a single voltage measurement and eight current injection patterns are used. Their method is similar to the J-substitution algorithm in concept, but they have also studied its performance under opposite drive and cosine injection patterns.

İder *et.al.* [23, 24] have developed and applied to real data based on a method for reconstructing from the measured magnetic field without having to calculate the current density, using an iterative sensitivity matrix approach. Seo *et.al.* [25] have also proposed a method which makes use of **B** only. In both of these methods only a single component of **B** is used. This provides a major practical advantage over the methods utilizing **J** because to obtain **J** by taking the curl of **B** requires the measurement of its all three components.

İder *et.al.* [26, 27] have proposed some new reconstruction methods using current injection to an object. Methods can be classified in two different types: First type assumes the current density is known and calculates conductivity distribution at this basis. Second type assumes only one component of magnetic flux density is known and finds the conductivity distribution in one slice of the 3D object.

1.2 The Objective and Scope of the Thesis

This thesis is devoted to develop MR-EIT image reconstruction algorithms for 3D objects which are insensitive to off-slice effects and require less computation time for reconstruction. The proposed algorithm is not implemented on real objects, instead only computer simulations are made. Also the required magnetic flux density data are generated by computer. When doing this, it is always assumed that current is induced to the object by coils around the object.

In this thesis, a new reconstruction algorithm is proposed for a novel imaging technique named as Induced Current Magnetic Resonance-Electrical Impedance Tomography. This algorithm is proposed for the reconstruction of conductivity distribution in an object in three dimensions. The algorithm is based on the induction of a conductive object by a coil and generating induced currents inside of the object. These induced currents will create a new secondary magnetic flux density to reject the primary magnetic flux density generated by the coil. The relation between magnetic flux densities enables to visualize the conductivity distribution by using some image reconstruction techniques.

1.3 Organization of the Thesis

This thesis is organized in the following order: Chapter 2 describes the forward problem of MR-EIT which is used in the calculation of current density inside of the object. In Chapter 3, the reconstruction technique used in this thesis is derived and the iteration process to calculate the conductivity is explained. Then, Chapter 4 introduces the conductivity models used in computer simulations, describes how measurement noise can be simulated and gives all simulation results. Finally, Chapter 5 concludes the thesis.

In Appendix A, primary magnetic flux density and vector magnetic potential calculations are given. Appendix B covers some properties of a tetrahedron. Finite Difference Formulation is given in Appendix C. In Appendix D, how one can measure the magnetic field due to eddy currents is given. In Appendix E, the calculation errors of the forward problem are argued. Finally, the source codes for MATLAB are given in Appendix F.

Chapter 2

The Forward Problem of Induced Current MR-EIT

In Induced Current MR-EIT, a coil induces current in the object and this induced current distribution is a function of conductivity distribution. *Forward Problem* of IC MR-EIT can be stated as finding the magnetic flux density inside the object by using the quantities magnetic vector potential due to the coil, A_p, and conductivity distribution, σ. In this chapter, we will formulate the equations that will start from the conductivity distribution and reach to the magnetic flux density due to eddy currents inside the object.

2.1 Formulation of Forward Problem

By applying a low frequency magnetic flux density, which is created by a coil, into an isotropic nonmagnetic and conductive media, which has a volume Ω and boundary $\partial\Omega$, we can generate a quasi-static current density distribution as a function of conductivity inside of the object, as illustrated in Figure 2.1.

Maxwell's equations for a sinusoidally varying electromagnetic field in a linear,

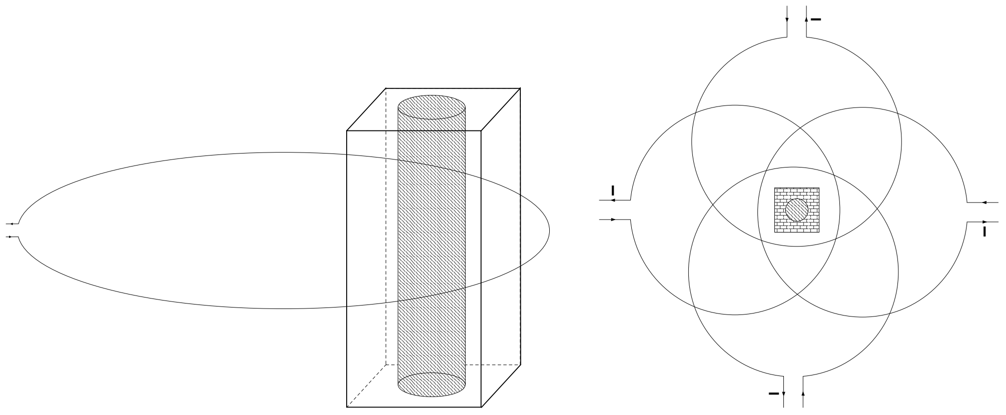

(a) (b)

Figure 2.1: IC MR-EIT imaging scheme for a cubical conductive object containing a cylindrical region with a different conductivity distribution from the rest of the object. (a) 3D illustration of a coil around the object on the z = 0 plane. (b) 2D illustration of four coils positioned around the object.

nonmagnetic, isotropic and conductive medium is given below [28]:

$$\nabla \times \bar{E} = -j\omega\bar{B} \tag{2.1}$$

$$\nabla \times \bar{B} = \mu_0(\sigma + j\omega\epsilon)\bar{E} \tag{2.2}$$

$$\nabla \cdot \bar{D} = \rho \tag{2.3}$$

$$\nabla \cdot \bar{B} = 0 \tag{2.4}$$

with continuity equation

$$\nabla \cdot \bar{J} = -j\omega\rho_v \tag{2.5}$$

where $\bar{D}$ and $\bar{J}$ related to $\bar{E}$ by

$$\bar{D} = \epsilon\bar{E} \tag{2.6}$$

$$\bar{J} = \sigma\bar{E} \tag{2.7}$$

where σ is the conductivity and $\rho_v$is the charge density.

By using the property $\nabla \cdot (\nabla \times \bar{U}) = 0$ for any vector $\bar{U}$ and combining with Eq. 2.4, it is possible to introduce the vector magnetic potential $\bar{A}$ defined by [28]

$$\bar{B} = \nabla \times \bar{A} \tag{2.8}$$

and using Eq. 2.1, one can find an expression for $\bar{E}$ as

$$\bar{E} = -\nabla\phi - j\omega\bar{A} \tag{2.9}$$

where ϕ is defined as the scalar potential.

After introducing some electromagnetic terms and their definitions, we can derive the forward problem. We will first reconstruct the continuity equation by using Eqs. 2.3, 2.6 and 2.7 [29].

$$\nabla\cdot((\sigma + j\omega\epsilon)\bar{E}) = 0 \tag{2.10}$$

By combining Eqs. 2.9 and 2.10, we get

$$\nabla\cdot((\sigma + j\omega\epsilon)\nabla\phi) = -j\omega\nabla\cdot((\sigma + j\omega\epsilon)\bar{A}) \tag{2.11}$$

Right side of Eq. 2.11 can be rewritten by using the property $\nabla\cdot(f\bar{U}) = f\nabla\cdot\bar{U} + \bar{U}\cdot\nabla f$,

$$\nabla\cdot((\sigma + j\omega\epsilon)\nabla\phi) = -j\omega((\sigma + j\omega\epsilon)\nabla\cdot\bar{A} + \bar{A}\cdot\nabla(\sigma + j\omega\epsilon)). \tag{2.12}$$

In Eq. 2.12, $\bar{A}$ must be specified [28]. Since $\bar{A}$ is an auxiliary function and only its curl is related to an observable physical quantity, we freely specify the divergence of $\bar{A}$. We can use Coulomb's Gauge, which adopts $\nabla\cdot\bar{A} = 0$. When the conductivity distribution in an object is not uniform, the Coulomb's Gauge is a good selection [30]. Then, the formulation becomes

$$\nabla\cdot((\sigma + j\omega\epsilon)\nabla\phi) = -j\omega\bar{A}\cdot\nabla(\sigma + j\omega\epsilon) \;\texttt{in}\; \Omega \tag{2.13}$$

where Ω defines the volume of the object.

The boundary condition of the partial differential equation found above can be formulated by equating the normal component of the total current on $\partial\Omega$ to zero to obtain using Eq. 2.9

$$\frac{\partial\phi}{\partial n} = -j\omega\bar{A}\cdot\bar{n}. \tag{2.14}$$

Here we will make two assumptions to simplify our formulation. First we will assume that the total magnetic vector potential $\bar{A}$ is approximately equal to the primary vector potential, A_p, produced by the coil in the absence of the conductive object [29]. Secondly, the displacement current ($j\omega\epsilon\bar{E}$, where $\omega = 2\pi f$, $f = 1000Hz$) is assumed to be negligible compared to the conduction current. Using these assumptions, Eq. 2.13 and 2.14 become

$$\nabla \cdot (\sigma \nabla \phi) = -j\omega \bar{A}_p \cdot \nabla \sigma \tag{2.15}$$

$$\frac{\partial \phi}{\partial n} = -j\omega \bar{A}_p \cdot \bar{n}. \tag{2.16}$$

If the phase of the current on the coil is taken as 0, then the current will be purely real. As a consequence, the magnetic flux density and magnetic vector potential due to coil will also be purely real. Since $\bar{A}_p$ is taken as purely real and σ cannot be complex, ϕ will be purely imaginary.

2.2 Numerical Solution of the Forward Problem

The complete solution of the forward problem requires the successive solution of Eqs. 2.15, 2.16, 2.9 and 2.7. The first equation of the chain is the most difficult one. In general, it is not possible to find an analytical solution to this equation. We have used *Finite Element Method* (FEM) to solve Eq. 2.15 with the boundary condition Eq. 2.16 numerically.

2.2.1 Finite Element Method (FEM)

In FEM, the exact value of the solution is found by solving for finite numbers of points in space named as *nodes.* Nodes are also corners of *finite elements* which divide the region into small closed regions. The solution domain for a geometry which contains elements and nodes is called the *finite element mesh.* In this study, tetrahedral elements are used as finite elements. In three dimensional space, tetrahedron is the simplest form of volume with minimum number of corners in order to make a volume. The object is assumed to be a rectangular prism and then it is divided into 64 slices in the z direction. Every slice is also divided into 32×32 cubic elements. After all, each cubic element can be represented by 6 tetrahedrons. Conductivity is assumed to be constant in a cube. As will be described in the next section, magnetic flux density will be calculated at hexahedron (cube) centers.

In a tetrahedron element, potential field can be approximated as a planar function of space. Since there exists four corners of tetrahedron, four newly defined functions N_i^j can be used in three dimensions which have the value 1 on one corner and 0 on the rest of them. The formulation of N_i^j is

$$N_i^j = a_i^j + b_i^j x + c_i^j y + d_i^j z \tag{2.17}$$

where a_i^j, b_i^j, c_i^j and d_i^j are selected as the values which provide the above conditions of N_i^j.

Potential at a point in the tetrahedron can be calculated as

$$\phi_i = \begin{cases} N_i^1\phi_i^1 + N_i^2\phi_i^2 + N_i^3\phi_i^3 + N_i^4\phi_i^4, & \text{Inside of the } i\text{th element,} \\ 0, & \text{Outside of the } i\text{th element.} \end{cases} \tag{2.18}$$

where ϕ_i^js are the potential values at the corners of tetrahedrons.

Eq. 2.15 is converted into a matrix relation by using the formulation above for node potentials. By calculating the node potentials, it will be possible to calculate potential inside of the object at any point.

In this thesis, it is assumed that the conductivity inside of a hexahedron is constant. Therefore, left side of Eq. 2.15 is equal to zero,

$$\nabla \cdot (\sigma \nabla \phi) = 0 \texttt{ in a hexahedron.} \tag{2.19}$$

To convert this equation to a matrix form, Eq. 2.19 are tested with N_i^j functions inside the i^{th} tetrahedron,

$$\int\limits_{\Omega_i} N_i^j \nabla \cdot (\sigma_i \nabla \phi_i) d\Omega = 0 \tag{2.20}$$

By using the property $\nabla \cdot (f\bar{G}) = \nabla f \cdot \bar{G} + f\nabla \cdot \bar{G}$, we get

$$\int\limits_{\Omega_i} [\nabla \cdot (N_i^j \sigma_i \nabla \phi_i) - \nabla N_i^j \cdot \sigma_i \nabla \phi_i] d\Omega = 0, \tag{2.21}$$

and using the divergence theorem, equation becomes

$$\int\limits_{\Omega_i} \nabla N_i^j \cdot \sigma_i \nabla \phi_i d\Omega = \int\limits_{\partial\Omega_i} N_i^j \sigma_i \nabla \phi_i \cdot \hat{n} dS \tag{2.22}$$

The boundary condition for a tetrahedron is

$$\sigma \frac{\partial \phi}{\partial n} = -j\omega\sigma \bar{A}_p \cdot \hat{n} + J^* \cdot \hat{n} \tag{2.23}$$

where J^* is the current on the boundary. By applying Eq. 2.23 into Eq. 2.22, it can be obtained that

$$\int\limits_{\Omega_i} \nabla N_i^j \cdot \sigma_i \nabla \phi_i d\Omega = \int\limits_{\partial\Omega_i} N_i^j (-j\omega\sigma_i \bar{A}_i \cdot \hat{n} + J_{ij}^* \cdot \hat{n}) dS \tag{2.24}$$

Since N_i^j functions are linear inside the tetrahedron, their gradient will be constant and the gradient of potential inside the tetrahedron, which is equal to the constant multiplicatives of the gradient of these functions, is also constant. The only unknowns are the multiplicatives of the planar functions to obtain potential inside the tetrahedron. Since there exist four multiplicatives and four equations for each planar function, left side of Eq. 2.24 can be reshaped as a matrix-vector multiplication at this point. The right side of the Eq. 2.24 can also be reformed as a vector.

$$\begin{pmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{pmatrix} \cdot \begin{pmatrix} \phi_i^1 \\ \phi_i^2 \\ \phi_i^3 \\ \phi_i^4 \end{pmatrix} = \begin{pmatrix} \int\limits_{\partial\Omega_i} N_i^1 (-j\omega\sigma_i \bar{A}_i \cdot \hat{n} + J_{i1}^* \cdot \hat{n}) dS \\ \int\limits_{\partial\Omega_i} N_i^2 (-j\omega\sigma_i \bar{A}_i \cdot \hat{n} + J_{i2}^* \cdot \hat{n}) dS \\ \int\limits_{\partial\Omega_i} N_i^3 (-j\omega\sigma_i \bar{A}_i \cdot \hat{n} + J_{i3}^* \cdot \hat{n}) dS \\ \int\limits_{\partial\Omega_i} N_i^4 (-j\omega\sigma_i \bar{A}_i \cdot \hat{n} + J_{i4}^* \cdot \hat{n}) dS \end{pmatrix} \tag{2.25}$$

$$S_{jk} = \sigma_i V_i (b_i^j b_i^k + c_i^j c_i^k + d_i^j d_i^k) \tag{2.26}$$

where V_i is the volume of i^{th} tetrahedron.

Here, the Eq. 2.15 is converted to a linear equation system of the form

$$S_{con} \cdot \phi = b \tag{2.27}$$

where S_{con} is a sparse matrix representing the connected coefficients and named as *connected coefficient matrix*, ϕ is the node potentials vector to be found, b is the boundary condition vector. For a mesh containing N nodes, the size of S_{con} is N×N and the size of ϕ and b is N×1.

There will be only two tetrahedrons m and n which share the same surface and on this surface J^*_{mi} and J^*_{ni} will be exactly the same. However, these two currents are multiplied with the opposite directed vector normals. When we combine the element matrices in the connected coefficient matrix, they will cancel each other. Therefore, it is possible to ignore the surface currents of tetrahedrons.

The square matrix S_{con} has to be nonsingular for invertibility. Singularity means that the rank of the matrix should be equal to N. However, Neumann type boundary value problem needs a reference potential to be solved. Then, this problem has a rank of N-1 without a reference potential. This can be solved by replacing the ith row of the matrix with all zeros except the (i,i)th entry with 1 instead. In addition, ith element of the b vector is set to zero.

After finding the node potentials, it is possible to calculate electric field intensity and current density inside of the object. If the gradient of the potential inside of a tetrahedron is taken, it is possible to see from Eqs. 2.18 and 2.17 that it will be constant inside of the tetrahedron. By applying Eqs. 2.9 and 2.7, the electric field intensity and the current density are calculated at the centers of the tetrahedrons. It is assumed that magnetic vector potential due to coil is constant inside of a tetrahedron so that electric field intensity and current density inside of a tetrahedron is also constant.

2.3 Computation of Secondary Magnetic Flux Density

Biot-Savart relation describes the magnetic flux density produced by various types of electrical current distributions. The numerical calculation of magnetic flux density requires the discretization of Biot-Savart relation which is given below,

$$\bar{B}(\bar{r}) = \frac{\mu_0}{4\pi} \int_{\Omega} \bar{J}(\bar{r}') \times \frac{\bar{r} - \bar{r}'}{|\bar{r} - \bar{r}'|^3} dv'. \tag{2.28}$$

In general, the reconstruction algorithms utilizing magnetic flux density needs the values of magnetic flux density at cartesian grids for simplicity. For this reason magnetic flux density will be found at the centers of the cubes.

Since current density assumed to be constant in a tetrahedron, Eq. 2.28 becomes to

$$\bar{B}_{ji} = \frac{\mu_0}{4\pi} \int_{\Omega_i} \bar{J}_i \times \frac{\bar{r} - \bar{r}'}{|\bar{r} - \bar{r}'|^3} dv'. \tag{2.29}$$

If the tetrahedron is small enough and field points are adequately far from its gravity center then source points can be replaced with gravity center point of tetrahedron as below,

$$\bar{r} - \bar{r}' \approx \bar{r} - \bar{r}_g' \equiv \bar{R} \tag{2.30}$$

where $\bar{R}$ is the distance vector between field point $\bar{r}$ and center of gravity point $\bar{r}_g$ of tetrahedron. However, the tetrahedron centers and the hexahedron center sharing the same cube are very close each other. Fortunately, in our case, the current changes inside a cubic element very slowly and assuming the current is nearly constant inside a hexahedron fixes the error we made with the assumption given in Eq. 2.30. An error analysis for the calculation of magnetic flux densities is given in Appendix E.

Continuing that Eq. 2.30 is a good approximation, Eq. 2.29 can be written as,

$$\bar{B}_{ji} = \frac{\mu_0}{4\pi} \bar{J}_i \times \frac{\bar{R}_{ji}}{|\bar{R}_{ji}|^3} V_i \tag{2.31}$$

where $\bar{R}_{ji}$ is distance vector between center of gravity of the i^{th} tetrahedron and geometric center of j^{th} cube. If all the contributions of total tetrahedrons inside the object is summed up, the magnetic flux density at the center of a cube is calculated as,

$$\bar{B}_{ji} = \frac{\mu_0}{4\pi} \sum_{i=1}^{P} \bar{J}_i \times \frac{\bar{R}_{ji}}{|\bar{R}_{ji}|^3} V_i \tag{2.32}$$

where j = 1,2,...,Q, Q is the number of cubes and P is the number of tetrahedrons inside of the object. By applying the curl operation, all three components of magnetic flux density can be calculated as shown below,

$$B_x^{ji} = \frac{\mu_0}{4\pi} \sum_{i=1}^{P} \frac{\bar{J}_y^i R_z^{ji} - \bar{J}_z^i R_y^{ji}}{|\bar{R}_{ji}|^3} V_i \tag{2.33}$$

$$B_y^{ji} = \frac{\mu_0}{4\pi} \sum_{i=1}^{P} \frac{\bar{J}_z^i R_x^{ji} - \bar{J}_x^i R_z^{ji}}{|\bar{R}_{ji}|^3} V_i \tag{2.34}$$

$$B_z^{ji} = \frac{\mu_0}{4\pi} \sum_{i=1}^{P} \frac{\bar{J}_x^i R_y^{ji} - \bar{J}_y^i R_x^{ji}}{|\bar{R}_{ji}|^3} V_i \tag{2.35}$$

In our meshing strategy, there is no possibility of singularity since there is no coincidence between the center of cubes and center of tetrahedrons.

Chapter 3

The Inverse Problem of Induced Current MR-EIT

The aim of MR-EIT is to reconstruct the unknown interior resistivity distribution of three dimensional objects. In the previous chapter, we tried to explain how some measurable field quantities can be calculated numerically from a known conductivity distribution and boundary conditions. The formulation method we introduced in the previous chapter is named as *Forward Problem* , and the computer program that realizes this process is named as *Forward Solver.* In this chapter, we will explain how to calculate unknown conductivity distribution from known field quantities and current densities. The method for this process is named as *Inverse Problem.*

Finding simple equations for the inverse problem is not as easy for as the forward problem. Inverse problem contains nonlinear differential equations which may be not easy to solve. Furthermore, both of the problems generally do not have analytic solutions, have highly nonlinear relations, and require non-standard, equation specific solution techniques. A systematic solution method of resistivity from an equation is named as a *Reconstruction Algorithm.*

In our problem, Reconstruction algorithms may use inside measured magnetic flux density, some peripheral voltage measurements and boundary information which includes shape and position of boundary. Today's systems can only measure magnetic flux density. As a result, in MR-EIT, all reconstruction algorithms have to utilize magnetic flux density first.

3.1 Formulation of the Inverse Problem

Assume that, an object Ω is in same conditions with the object which was described in the beginning of the previous chapter and satisfies all assumptions we have made for it. So the relation between electric field intensity $\bar{E}$ and magnetic flux density $\bar{B}$ is

$$\nabla \times \bar{E} = -jw\bar{B} \tag{3.1}$$

and $\bar{B}$ will have both real and imaginary parts

$$\bar{B} = \bar{B}_R + j\bar{B}_I. \tag{3.2}$$

Total $\bar{B}$ has two parts. First one is created by the coil and second part is created by the currents induced inside of the object. Primary magnetic flux density $\bar{B}_P$ is purely real since the current inside the coil is real. Secondary magnetic flux density $\bar{B}_S$ will be purely imaginary since the potentials inside the object is purely imaginary and the current density is purely imaginary.

$$\bar{B} = \bar{B}_{P_R} + j\bar{B}_{P_I} + \bar{B}_{S_R} + j\bar{B}_{S_I} \tag{3.3}$$

and since, $\bar{B}_{P_I}$ and $\bar{B}_{S_R}$ in the above equation will be equal to zero,

$$\bar{B} = \bar{B}_{P_R} + j\bar{B}_{S_I}. \tag{3.4}$$

If we combine Eqs. 3.1 and 3.4, the equation below is obtained,

$$\nabla \times \bar{E} = w(\bar{B}_{S_I} - j\bar{B}_{P_R}). \tag{3.5}$$

By taking the imaginary part of the equation, we get

$$\nabla \times \bar{E}_I = -w\bar{B}_{P_R}. \tag{3.6}$$

As given before, Ohm's Law states that $\bar{E} = \rho \bar{J}$. Substituting this in the previous formula yields,

$$\nabla \times \rho \bar{J}_I = -w \bar{B}_{P_R}. \tag{3.7}$$

Here, $\bar{J}_I$ is the imaginary part of the induced current inside of the object. An equivalent form of the previous formula is

$$\nabla \rho \times \bar{J}_I + \rho \nabla \times \bar{J}_I = -w \bar{B}_{P_R} \tag{3.8}$$

by using the vector identity $\nabla \times f\bar{G} = \nabla f \times \bar{G} + f \nabla \times \bar{G}$ for scalar field f and vector field $\bar{G}$. If we know $\bar{J}$, Eq. 3.8 can be interpreted as yielding information on the gradient of ρ. Current density inside of the object can be calculated by $\bar{J}_I = \nabla \times \frac{\bar{B}_{S_I}}{\mu_0}$, the point form of Ampere's Law. By taking the curl of both sides, we obtain

$$\nabla \times \bar{J}_I = \nabla \times (\nabla \times \frac{\bar{B}_{S_I}}{\mu_0}). \tag{3.9}$$

By using the vector identity $\nabla \times \nabla \times \bar{G} = \nabla(\nabla \cdot \bar{G}) - \nabla^2 \bar{G}$ for any vector field, right side of the equation becomes

$$\nabla \times \bar{J}_I = \nabla(\nabla \cdot \frac{\bar{B}_{S_I}}{\mu_0}) - \nabla^2 \frac{\bar{B}_{S_I}}{\mu_0}. \tag{3.10}$$

Since $\nabla \cdot \bar{B} = 0$ for any magnetic flux density, the equation becomes

$$\nabla \times \bar{J}_I = -\nabla^2 \frac{\bar{B}_{S_I}}{\mu_0} \tag{3.11}$$

If we use this equation inside of Eq. 3.8

$$\nabla \rho \times \bar{J}_I - \rho \nabla^2 \frac{\bar{B}_{S_I}}{\mu_0} = -w \bar{B}_{P_R} \tag{3.12}$$

If we know the all three components of magnetic flux density, it is better to use Ampere's Law and 3.8 together instead of this formulation. Since the second derivatives of magnetic flux density is inside of the formulation, this formula is more likely to be noise sensitive than Ampere's Law. However, it is difficult to measure all three components of magnetic flux density. Experimental small object can be easily rotated to measure magnetic flux density through each direction of (x,y,z). But human body cannot be rotated as easily as an experimental object. Eq. 3.8 can be separated into three components. So if we measure magnetic flux density in one dimension, we will be able to solve conductivity distribution inside of the object.

Let's now return to Eq. 3.12. To gain further insight about it, expand its components in all three directions,

$$\begin{pmatrix} 0 & J_z & -J_y \\ -J_z & 0 & J_x \\ J_y & -J_x & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial \rho}{\partial x} \\ \frac{\partial \rho}{\partial y} \\ \frac{\partial \rho}{\partial z} \end{pmatrix} = \begin{pmatrix} \rho \nabla^2 \frac{B_{S_{I_x}}}{\mu_0} - wB_{P_{R_x}} \\ \rho \nabla^2 \frac{B_{S_{I_y}}}{\mu_0} - wB_{P_{R_y}} \\ \rho \nabla^2 \frac{B_{S_{I_z}}}{\mu_0} - wB_{P_{R_z}} \end{pmatrix} \tag{3.13}$$

which provides an equation system consisting of three first order quasi-linear [1] partial differential equations. Each row of this system conveys information about gradient of ρ on two dimensional domains which are actually planes. For example first row deals with the gradient of ρ on constant x planes. Similarly second and third rows are related with constant y and z planes respectively. Therefore planes of different equations are perpendicular to each other. An illustration of perpendicular planes in the object is shown in Figure 3.1. It can be seen clearly that, if $\bar{J}$ is known on a plane then reconstruction of the conductivity distribution on that plane will be easy. It also gives an ability to reconstruct single slice instead of whole object.

3.2 Numerical Methods for Inverse Problem

In this section, an iterative magnetic flux density based image reconstruction algorithm will be presented. The algorithm is based on reconstruction of resistivity by using measured magnetic flux density in any of x, y, and z directions. This method removes the necessity of object rotations, which is not easily applicable. Furthermore, slice reconstruction can be made by this algorithm, which will

[1]A special form of the nonlinear partial differential equations [31].

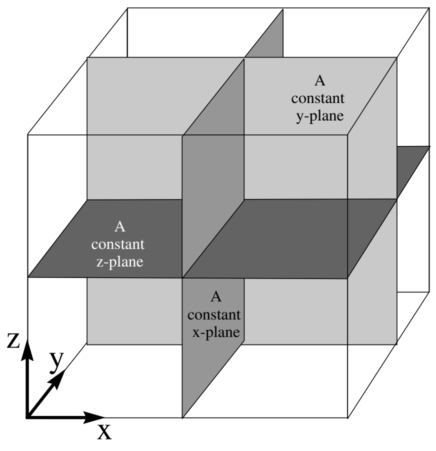


Figure 3.1: Reconstruction planes for x,y and z dimensions.

enable to get results in computationally less time.

The algorithm used in this section is implemented by using the idea of linear systems. In previous section, the inverse problem of Induced Current MR-EIT is introduced. The nonlinear relation between conductivity and magnetic flux density is given in Eq. 3.12. This type of nonlinear equations can be solved with linear systems of equations. Eq. 3.12 can be rewritten around a conductivity distribution point using Taylor series expansion with eliminated higher order terms. A linear matrix-vector relation can be generated from conductivity distribution domain to magnetic flux density domain. In this case, the solution of the inverse problem reduces to the inversion of the matrix generated and multiplication of it with the difference magnetic flux density vector. However, the solution of the linear system is a good predict of conductivity distribution but not enough close to the exact result. So, a few iterations of solution process by combining forward and inverse problems pair will give a better approximation to exact conductivity distribution.

3.2.1 Reconstruction by Solution of Linear Equation System

The algorithm used in this section uses only one component of magnetic flux density and tries to find a solution for the related row of Eq. 3.13,

If it is assumed that only **z** component of magnetic flux density is measured, then we have to concentrate on the third row of this matrix equation,

$$\frac{\partial\rho}{\partial x}J_y - \frac{\partial\rho}{\partial y}J_x - \rho\nabla^2\frac{B_{S_{I_z}}}{\mu_0} = -wB_{P_{R_z}} \tag{3.14}$$

which is a first order quasi-linear[2] partial differential equation which relates the partial derivatives of ρ and ρ itself with Laplacian of B_{sz}/μ_0 to ωB_{pz}. If both B_{sz} and B_{pz} are measured and $\nabla^2 B_{sz}$ is calculated from measured B_{sz} on a constant z slice, then the ρ distribution can be derived from the solution of this equation.

Generally, it is not easy to find a unique solution for Eq. 3.14. At this manner, to obtain the solution two or more current profiles is used.

Eq. 3.14 is discretized in a square Cartesian mesh in order to reconstruct the conductivity distribution. This newly defined mesh is a mesh with nodes which are the centers of hexahedrons of a slice of the original mesh. By replacing the derivatives with finite difference equations, the discretization is done. In Appendix C, finite difference approximation is derived. In the interior region of the object, central difference formula is used. On the other hand, at the edge hexahedrons of the object forward and backward difference formulas are used for convenience. One equation from interior of the object can be found by finite difference approximation for $(i,j)^{th}$ hexahedron as shown in Figure 3.2 is:

$$\frac{\rho_{(i+1,j)} - \rho_{(i-1,j)}}{2\Delta x}J_{y(i,j)} - \frac{\rho_{(i,j+1)} - \rho_{(i,j-1)}}{2\Delta y}J_{x(i,j)} - \rho_{(i,j)}\nabla^2(\frac{B_{sz(i,j)}}{\mu_0}) = -\omega B_{pz(i,j)}. \tag{3.15}$$

Since there are NM hexahedrons on a slice, NM equations can be formulated

[2]Equation is linear in the gradient of ρ on a slice perpendicular to the **z** direction,but not linear for ρ itself

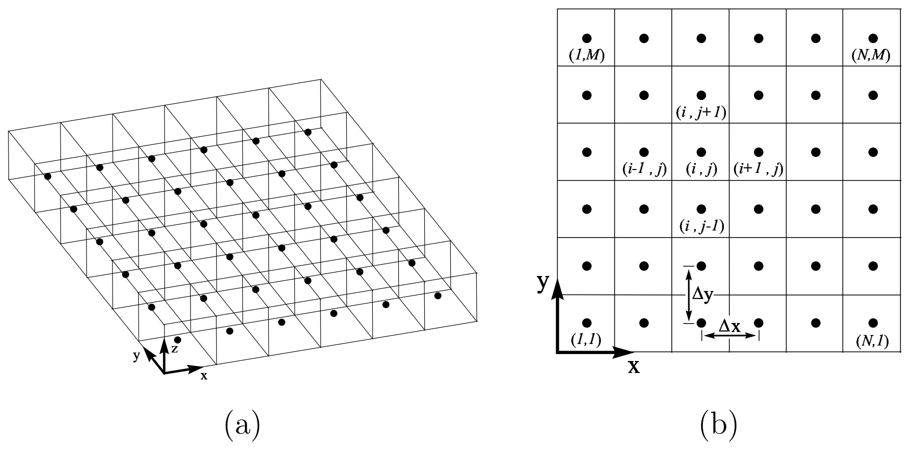


Figure 3.2: To reconstruct ρ on a slice, the third row of Eq. 3.13 is discretized by finite differences on the Cartesian grid consisting of center points of the slice hexahedrons. (a) 3D illustration of a slice with center points of its hexahedrons. (b) 2D illustration of a slice and indices of some hexahedrons used in Eq. 3.15.

like this. By rearranging and combining all the equation on a slice, matrix form of these equations can be constructed.

$$S\rho' = \Delta b \tag{3.16}$$

where S is the $NM \times NM$ coefficient matrix, ρ' is the $NM \times 1$ vector of all hexahedron values of conductivity distributions at that slice and Δb is the $NM \times 1$ vector of the right side of the Eq. 3.15.

Eq. 3.16 is constructed for only one current induction profile. For P different profiles, P different S and Δb will be obtained. In order to concatenate these matrices, a new non-square coefficient matrix $\Upsilon = [S_1^T S_2^T ... S_P^T]^T$ and $PMN \times 1$ vector $\Delta\beta = [\Delta b_1^T \Delta b_2^T ... \Delta b_P^T]^T$ can be defined that combines all the matrices. The new form of matrix equation is

$$\Upsilon\rho' = \Delta\beta. \tag{3.17}$$

After these calculations, the steps of iterative method can be defined as

Step 1 : Assume an initial ρ_0 for all slices. For the first iteration, it is taken as a uniform distribution. Set slice number k to 0.

Step 2 : Calculate Δb_ℓ vectors for $\ell = 1, 2, ..., P$ by using the calculated B_{pz} values for the k^{th} slice.

Step 3 : Measure B_{sz} and calculate $\nabla^2(B_{sz}/\mu_0)$ for the k^{th} slice. Calculate $\bar{J}_\ell$ using forward solver and obtain S_ℓ for $\ell = 1, 2, ..., P$.

Step 4 : Concatenate S_ℓ matrices and Δb_ℓ vectors to obtain combined system equation $\Upsilon \rho_i' = \Delta\beta$ of the k^{th} slice, where i is the iteration number.

Step 5 : Solve Eq. 3.17 to find the conductivity distribution of the k^{th} slice,ρ_i.

Step 6 : If k is not the maximum slice number, increase k by 1 and go to *Step2*. Else, check for stopping condition and stop if it is met. Else, go to *Step 1* and use the ρ_i found in *Step 5* as the initial vector.

In our simulation studies, 5 iterations are made. However, it is possible to select the stopping condition as the relative ℓ^2 norm error of the difference between the reconstructed image and the actual resistivity distribution instead of iteration count. Relative ℓ^2 norm error is defined as,

$$\rho_{\varepsilon_i} \triangleq 100 \frac{||\sigma_i - \sigma_{original}||_2}{||\sigma_{original}||_2} \tag{3.18}$$

where $\sigma_{original}$ is the original conductivity vector and σ_i is the conductivity contributed in i^{th} iteration. The the relative ℓ^2 norm errors of the simulations are given in the next chapter.

In the previous chapter, one can easily see that the eddy currents are directly related to conductivity values. Then, the secondary magnetic flux density is also related to conductivity distribution. At this manner, the coefficient matrix contains the information about conductivity distribution. In forward problem, we have to take a reference potential value to calculate the potential inside the object. However, there is no need for this type of modification in coefficient matrix. In this case, actually there must be no ill-conditioning problem in the coefficient matrix in the absence of numerical errors. However, some numerical errors exist in this solution methodology. In the next chapter, the singular value plots of simulation phantoms are given. Additionally, the solution methods to distract the ill-conditioning problem are discussed.

Chapter 4

Simulation Results

In this Chapter, computer simulation results will be shown for the algorithm given in previous chapter. No measurements were made for this thesis. All the required data are generated by computer. A rectangular prism shaped object is simulated with edge lengths $32 \times 32 \times 64 cm$. To assign a conductivity model, the conductivity of each hexahedron with 1 cm edge length is assumed to be constant.

To get better insight of the features of the algorithm, three different conductivity models are constructed and simulated on computer. In addition to this, the effect of the number of iterations and measurement noise are investigated for the algorithm. ρ distribution of all slices in z direction are reconstructed from the computer generated secondary magnetic flux density data except the 1^{st} and 64^{th} slices (bottom and top slices). The secondary magnetic flux densities at these two slices are inaccurate due to numerical errors made in the calculation of the Biot-Savart integral. Therefore, we did not calculate conductivity at these two slices. Instead, the conductivity values at 2^{nd} and 63^{rd} slices are taken as the conductivities of 1^{st} and 64^{th} slices, respectively.

Conductivity Region	Conductivity Model 1 (CM1)	Conductivity Model 2 (CM2)	Conductivity Model 3 (CM3)
Cylinder	0.6	$\times$	$\times$
Sphere (slices 21-36)	$\times$	0.4	$\times$
Sphere (slices 30-62)	$\times$	0.1	$\times$
Lungs	$\times$	$\times$	0.181-0.061
Heart	$\times$	$\times$	4.0-2.22
Backbone	$\times$	$\times$	0.143-0.055
Background	0.2	0.2	0.2

Table 4.1: Some conductivity values of the conductivity models (S/m).

4.1 Conductivity Models

Three conductivity models with different complexities are used for simulations. These three different conductivity models contain a cylinder (CM1), two spheres (CM2) and a human thorax model (CM3). In order to be close to the average body conductivity, the background conductivity is taken as $0.2S/m$. For each conductivity model, 30 slices are selected out of the 64 slices and these are given in Figures 4.1, 4.2 and 4.3, respectively.

The first model includes a simple cylinder centered at midpoint between the boundary of slice and the center of slice in the y direction. This model will give some information about how the solver behaves when the object is near the boundary and constant in z direction. In the second model, two spheres at different centers are used in the simulation. Since a sphere changes in all three dimensions, this simulation will give us some information about the effect of 3^{rd} dimension for the solution method. Finally, a computer generated human thorax model is used as third model to get a realistic simulation result. In thorax model, lungs and bones are less conductive and heart is more conductive than background conductivity. In Table 4.1, region conductivities are given for all three models.

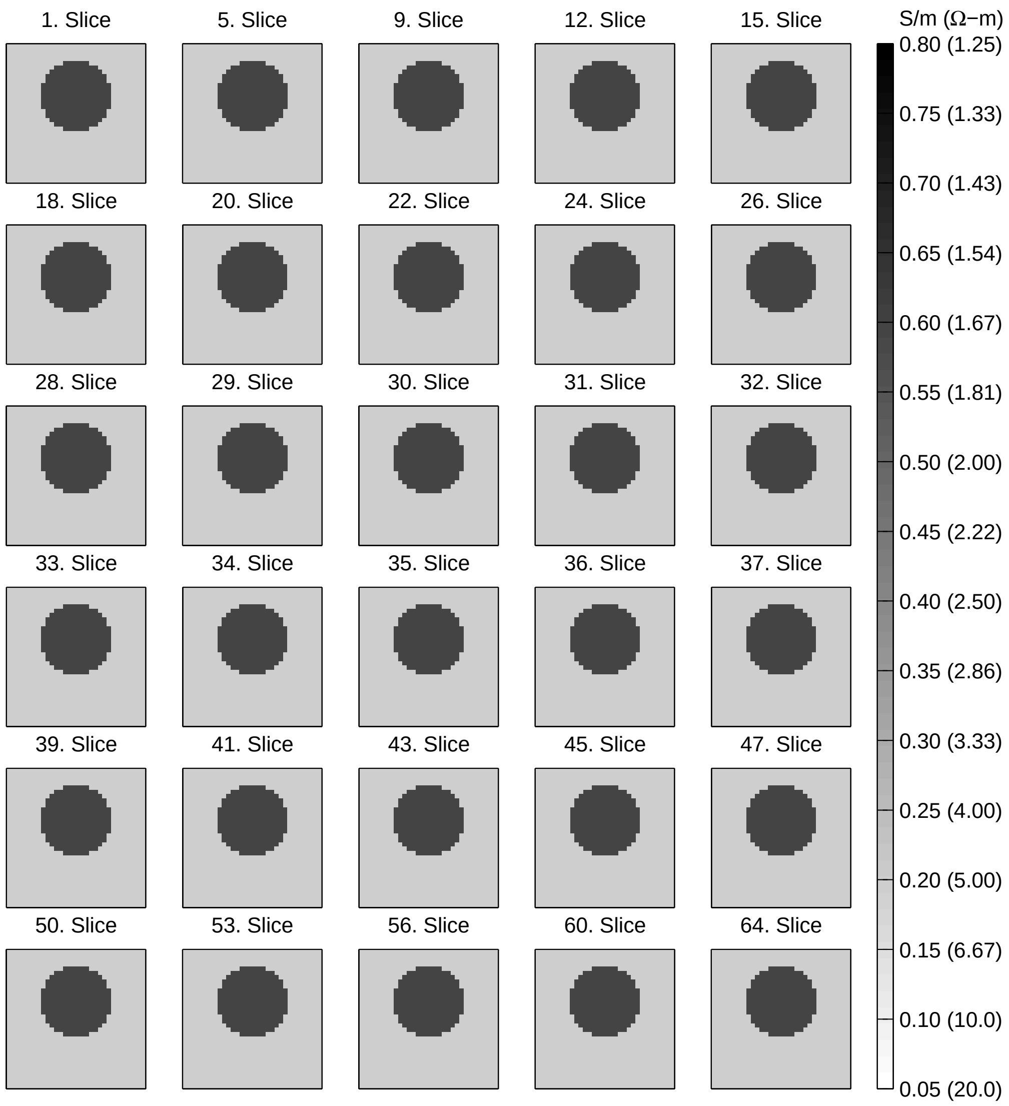


Figure 4.1: Some selected slices of the first conductivity model (CM1).

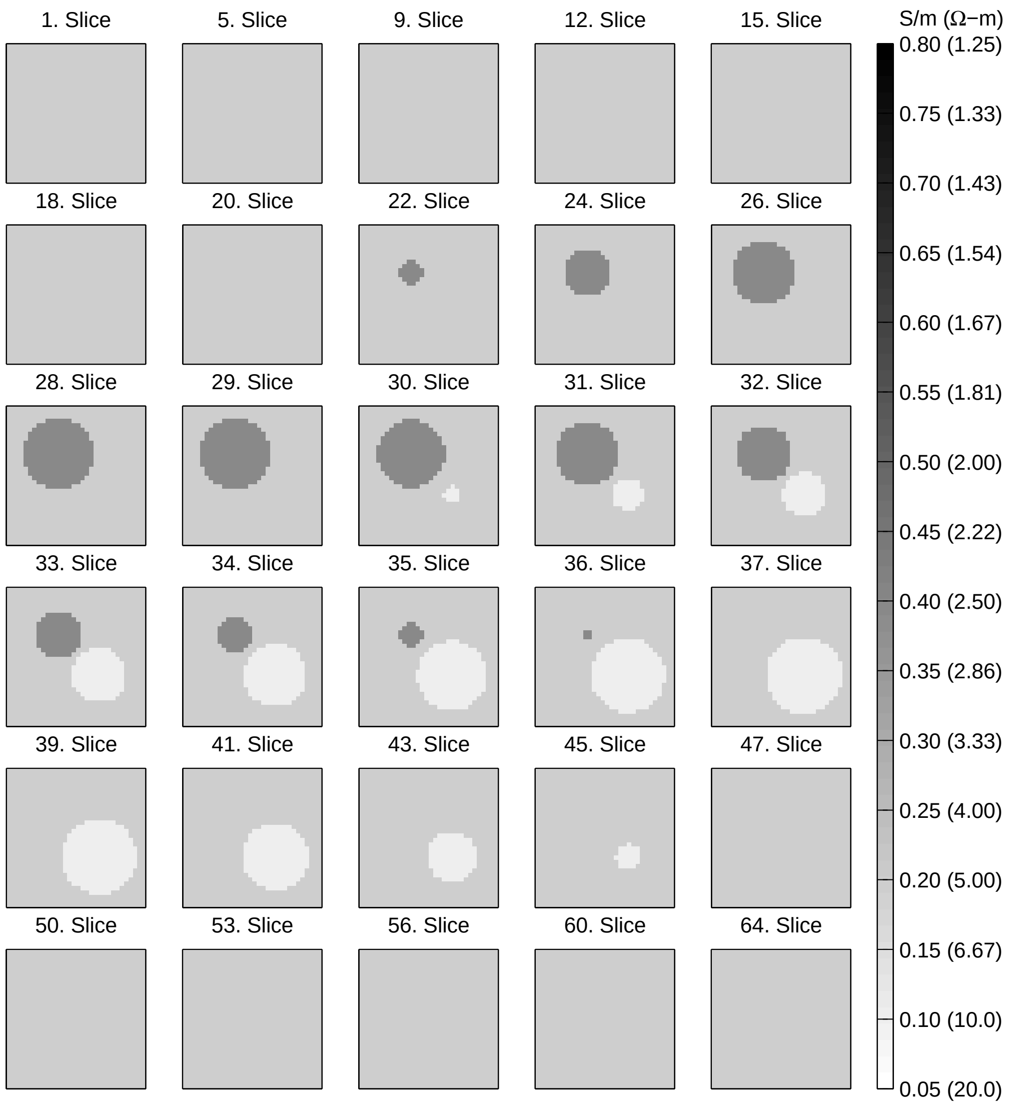


Figure 4.2: Some selected slices of the second conductivity model (CM2).

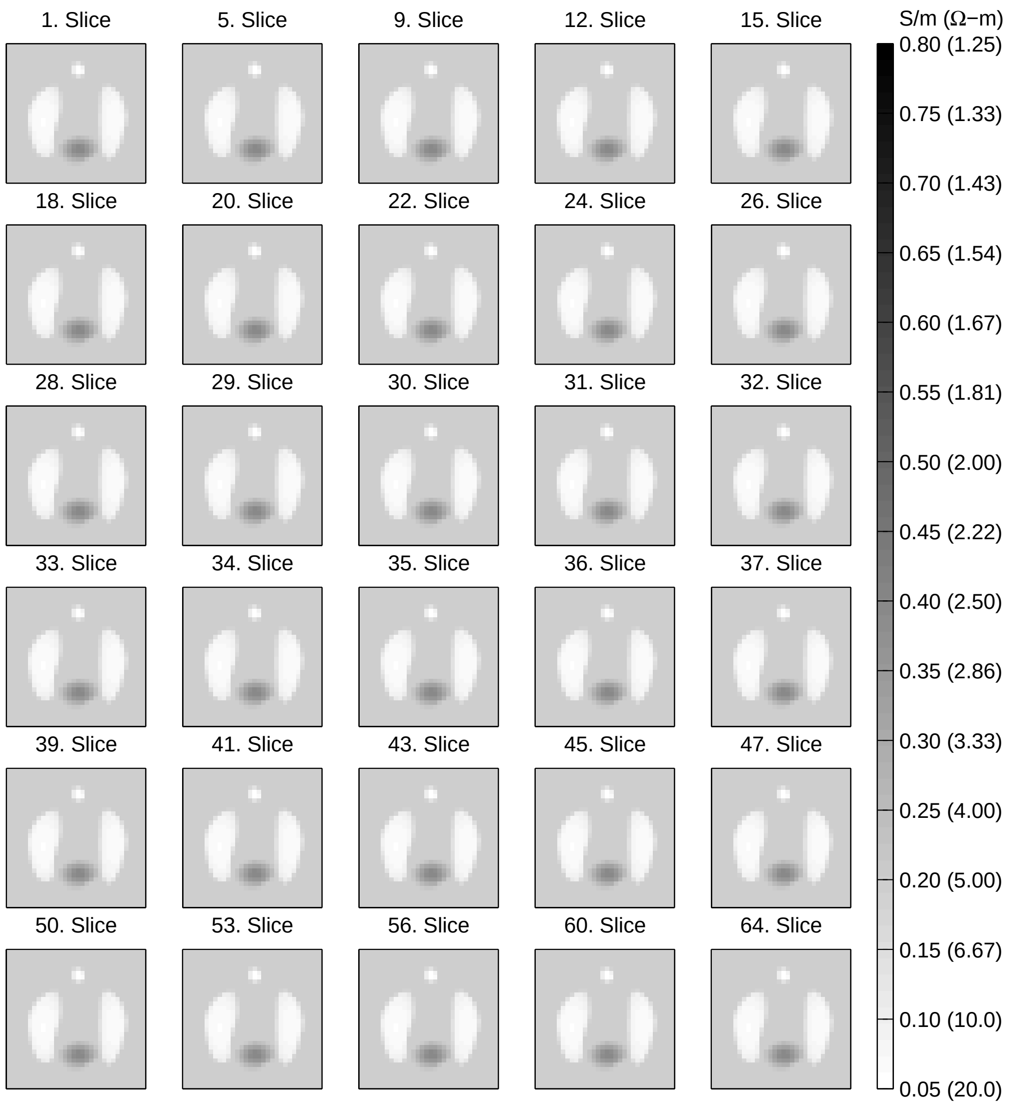


Figure 4.3: Some selected slices of the third conductivity model (CM3).

4.2 Measurement Noise

In the presence of measurement noise, the behavior of the algorithm is another issue to be understood. To be able to get some knowledge about this behavior, a generated Gaussian noise is added to the secondary magnetic flux density data. The standard deviation of noise for one component of secondary magnetic flux density is formulated as

$$\sigma_S = \frac{1}{2\gamma t_{curr} SNR} \tag{4.1}$$

where γ is gyro-magnetic ratio ($26.75 \times 10^7 rad/(sec \times Tesla)$), t_{curr} is the duration of spin echo experiment proposed for the solution of the solution method (Appendix D) and SNR is the signal-to-noise ratio of the MRI system [32]. The duration of spin echo experiment, t_{curr}, is taken as 100 ms. In order to be able to get better images, the experiment is repeated 100 times. Simulations are made for three different noise levels by assigning SNR value as 30, 60 and 90. For SNR values 30, 60 and 90, the corresponding standard deviations are approximately 62 pTesla, 31 pTesla and 21 pTesla by using Eq. 4.1. The calculated values are independent of magnetic flux density. As we increase the induced current over the coil, the SNR of the total system can be increased. However, there are some limitations over the magnitude level of the induced current.

The safety limit for induced/injected current inside the body is accepted as less than 1 mA at 1 kHz frequency [33]. To have less than 1 mA current inside the body, the maximum current density in the object must be about 4.88 mA/m^2. In order to achieve this, 18.5 A current must be applied to the coil. In this case, the average absolute magnetic flux density of induced current is 65 pTesla. For all SNR values 30, 60 and 90, the deviation of noise is comparable with magnetic flux density. However, after the laplacian operator, the magnetic flux density data becomes useless. If I = 2000 A current applied to the coil, an average of 0.5 A eddy current is induced in the object. In this case, secondary magnetic flux density is around 7 nTesla which is a better level with respect to the noise to be able to measure. However, this level of eddy current cannot be applied to human body. For the simulation purposes, this level is used. In the realization of the algorithm, some other numerical techniques such as singular value truncation, which decreases the effect of noise in the coefficient matrix, can be used.

4.3 Simulation Results for Algorithm

In this thesis, three different simulation phantoms are used. For each phantom, direct solutions of conductivity distributions without noise consideration are calculated. In Figures 4.4, 4.5 and 4.6, these results are presented for some selected slices for the first iteration except third model. Since it converges in the second iteration to a good result, the second iteration result is represented for third model. In addition, five iterations are made and the 32^{nd} slice (mid-slice) for each iteration are given in Figures 4.7(a), 4.7(b) and 4.7(c). The selection of iteration number is made from the calculated residual norm errors. In the first simulation model, 8 iterations are made and, after 5 iterations, the residual error converged to a constant value. Therefore, the iteration number for each three of the simulation models is selected as 5. As you can see from the residual norm errors of the other simulation models, 5 iteration is a good selection for all three models.

Calculation of magnetic flux density without noise is unrealistic. Therefore, a normally distributed noise with a standard deviation given in Equation 4.1 is added to secondary magnetic flux density for three different SNR values 30, 60 and 90. The simulation results for different SNR values are given in Figures 4.8(a), 4.8(b) and 4.8(c) for mid-slice.

In order to understand better how magnetic fields carry the conductivity information, the laplacian of calculated secondary magnetic flux densities for each simulation phantom at the mid-slice are given in Figures 4.9(a), 4.9(b) and 4.9(c). As you can see from the figures, the edge information of the conductivity is preserved in the secondary magnetic flux density. In the first simulation phantom, two constant magnetic flux density levels exist while the conductivity is constant. However, on the edges, the magnetic flux density information takes different values for each current induction profile. The combination of four current profiles

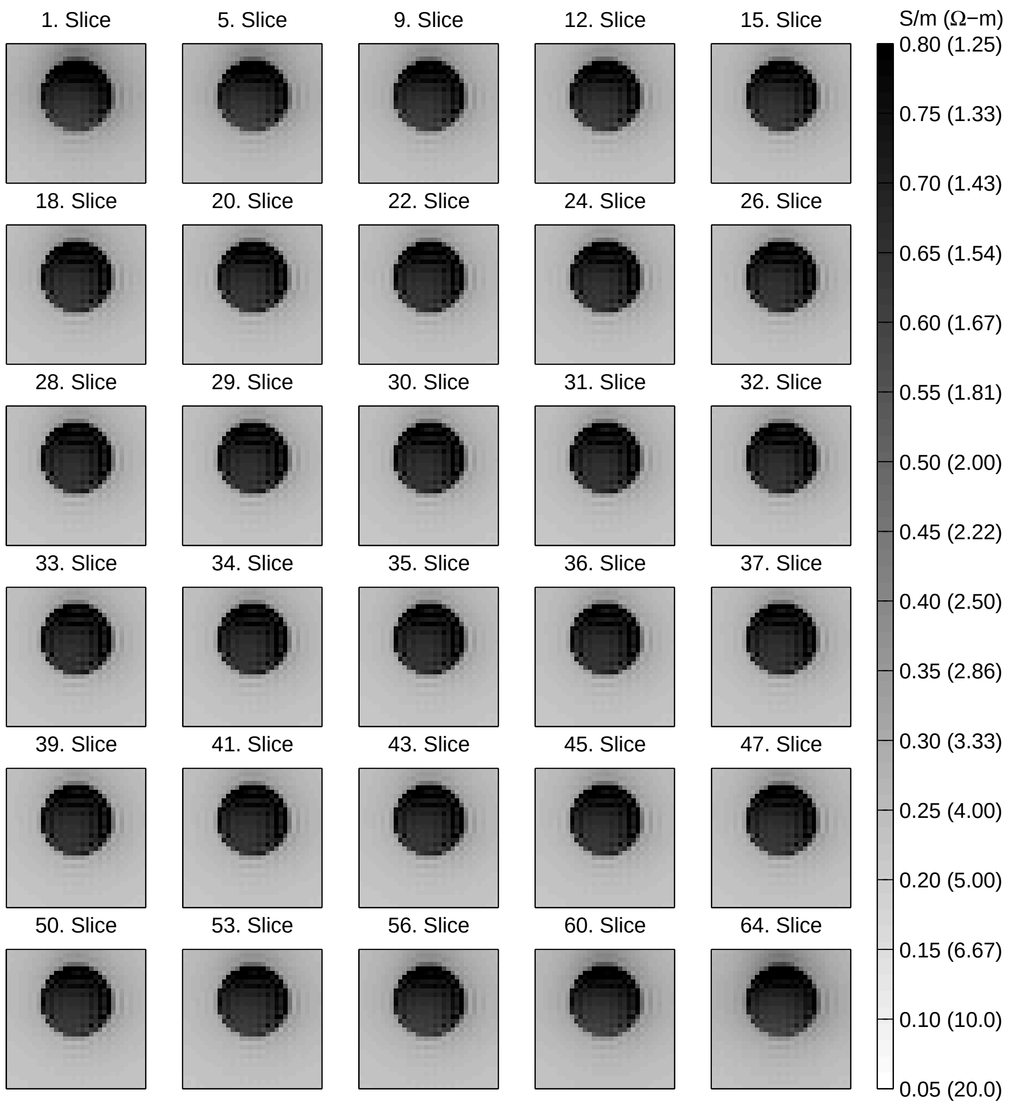


Figure 4.4: Some reconstructed slices of the first conductivity model (CM1).

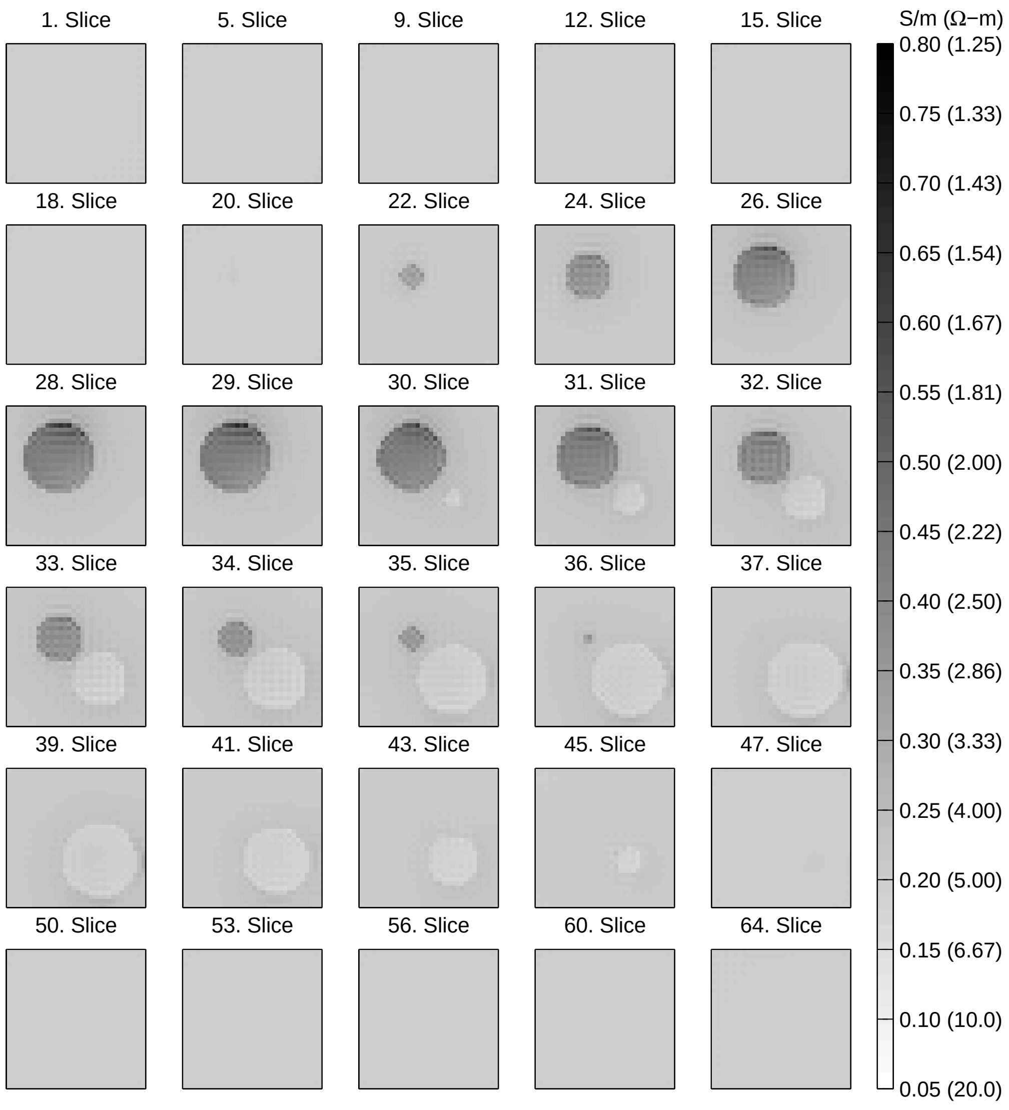


Figure 4.5: Some reconstructed slices of the second conductivity model (CM2).

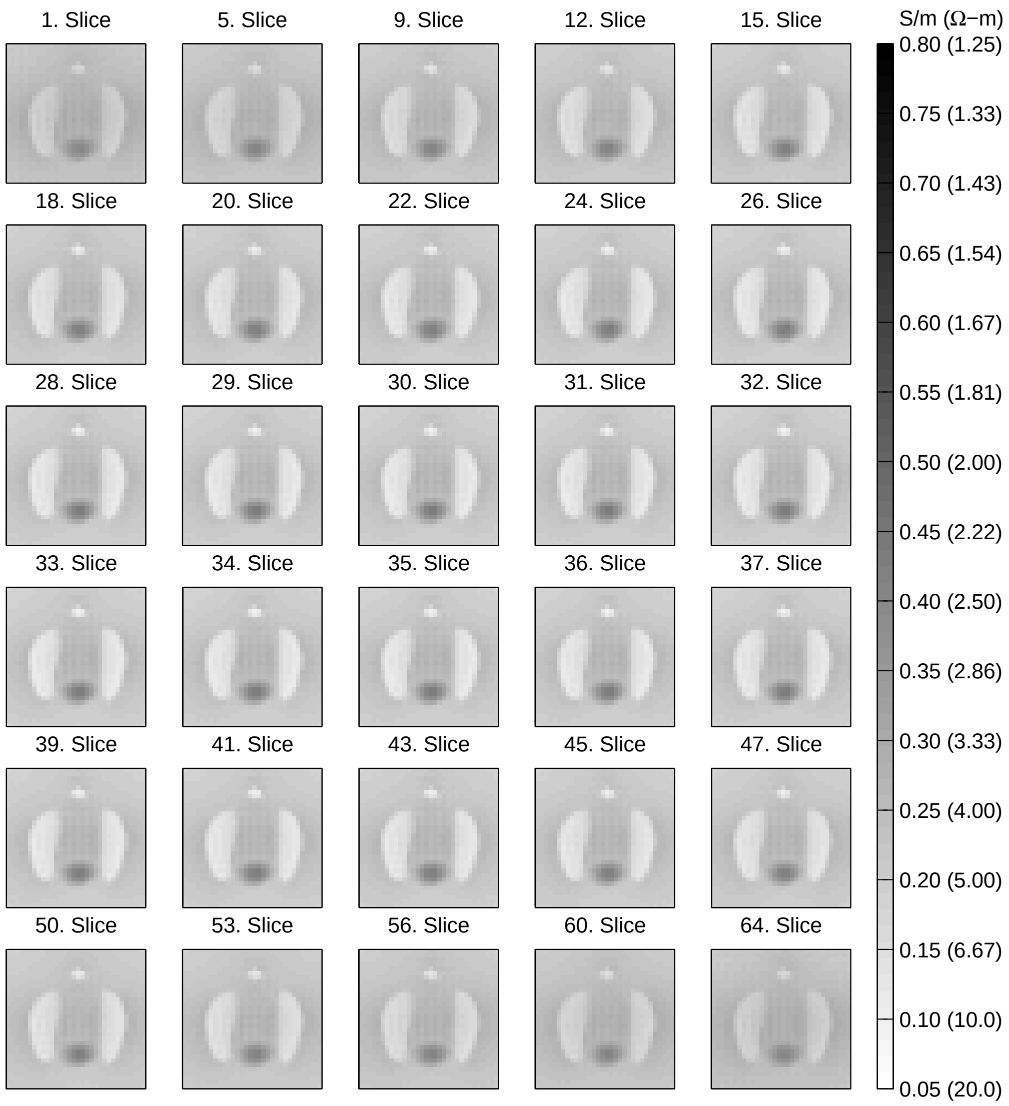
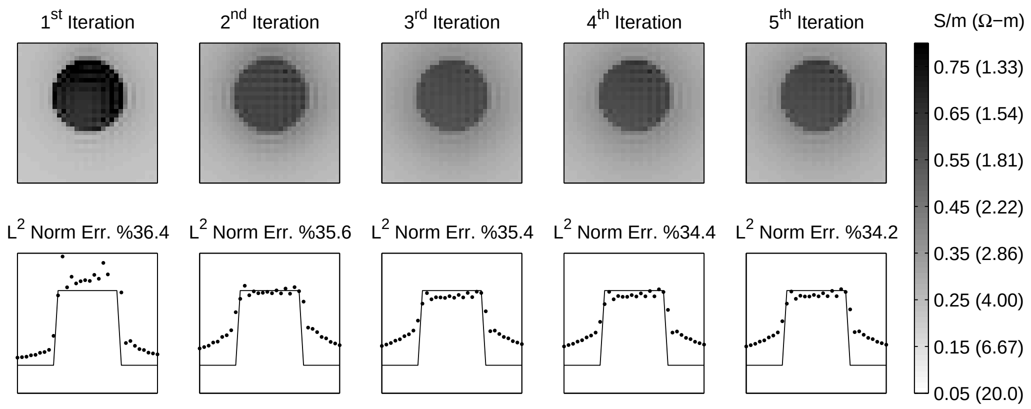
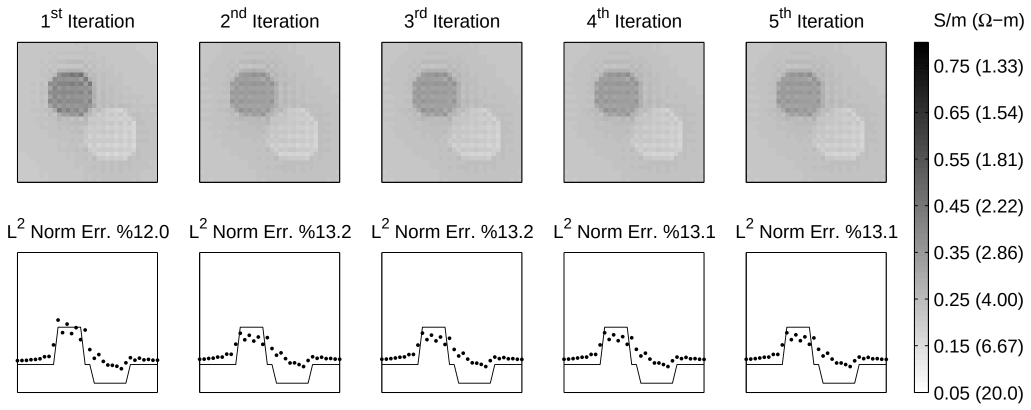


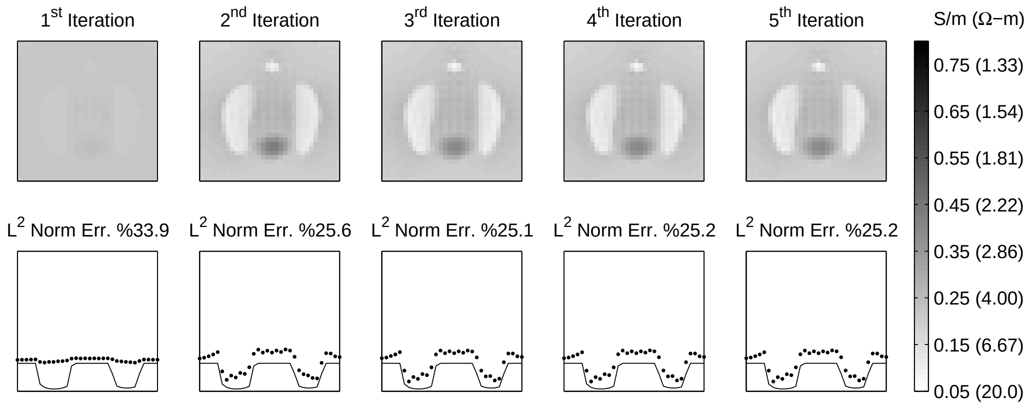
Figure 4.6: Some reconstructed slices of the third conductivity model (CM3).



(a)

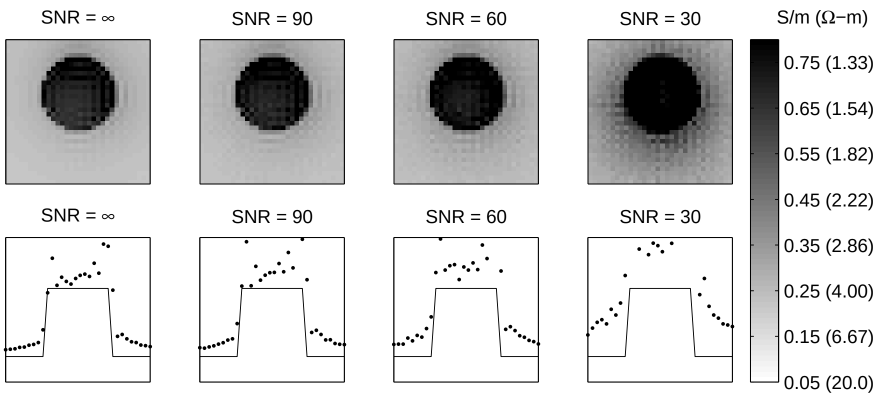


(b)

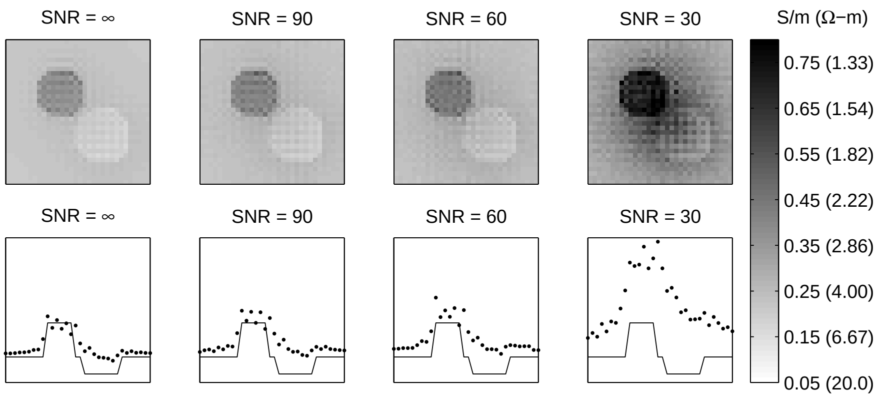


(c)

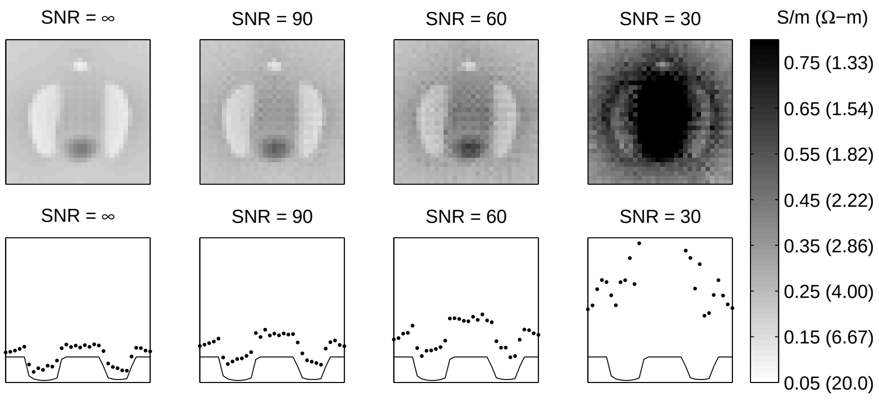
Figure 4.7: Effect of iteration over the reconstruction by the solution of linear equation system. Reconstructed images of 32^{nd} slice for three different conductivity models are given in (a) to (c) respectively for the first five iterations. No measurement noise is added to the secondary magnetic flux density. Also the profile plots of horizontal 16^{th} lines of images are at the bottom of each image. Relative L_2 norm errors indicate the difference between reconstructed and original images.



(a)



(b)



(c)

Figure 4.8: Reconstruction results of different SNR values for three conductivity models given in (a) to (c).

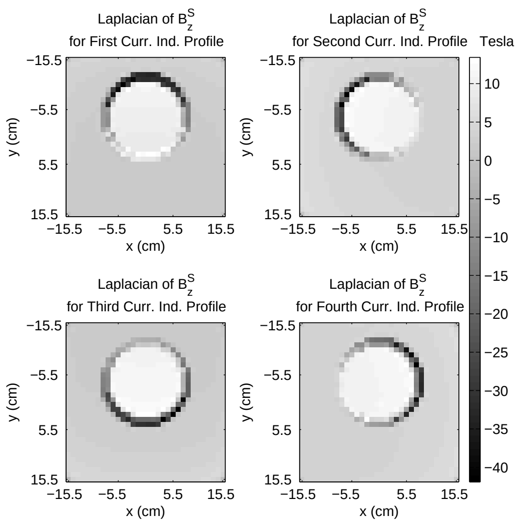
have the total edge information. If we use only one of the current profiles, the reconstructed image will contain only one edge information from one direction will be seen which is not a desired image of data.

Up to now, all the simulations are made for four current induction profiles. However, it is possible to use less current induction profiles in the solution. On the other hand, this brings some restrictions with itself. We have to consider the ill-conditioning problem of the matrix of linear system equation. In Figure 4.10, the singular value plots of the system matrices, which have one, two, three and four current induction profiles, of each phantom are given for the first iteration. From figures, it seems that at least two current induction profile is enough to be able to obtain good results. However, increasing the number of profiles shows better images.

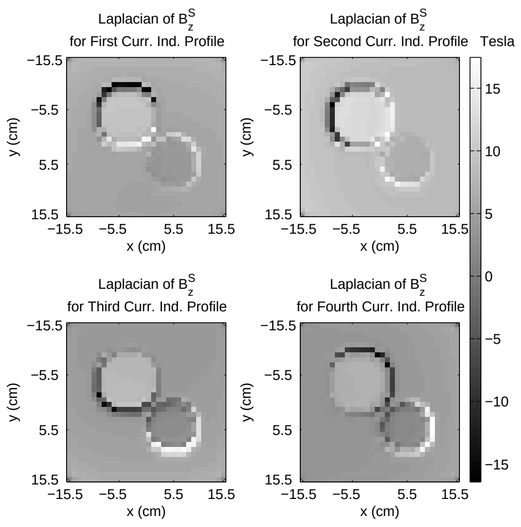
The condition numbers of the coefficient matrices for mid-slice of three different models are 37.36, 44.05 and 457.32 for CM1, CM2 and CM3,respectively. First two conductivity models, CM1 and CM2, seems to be well conditioned. However, it seems that the small ill-conditioning problem occurs in the third conductivity model. On the contrary, the ill-conditioning problem does not occur in the second iteration. It is possible to truncate some singular values of the coefficient matrix which is not covered in this thesis. The truncation can be made to achieve less noise dependent well-conditioned matrices to reconstruct and, as a result of this, better imaging with smaller current induction in the object. Since the condition numbers are not very large in our simulation phantoms, no truncation is applied in this thesis.

The matrix equation Eq. 3.17 is solved by taking the inverse of coefficient matrix and multiplicating with the right hand side vector. The inverse operation is done by using LU factorization with partial pivoting. Instead of using gaussian elimination for the inverse, first LU decomposition of the coefficient is found and the iteration given below is applied for each column of the matrix,

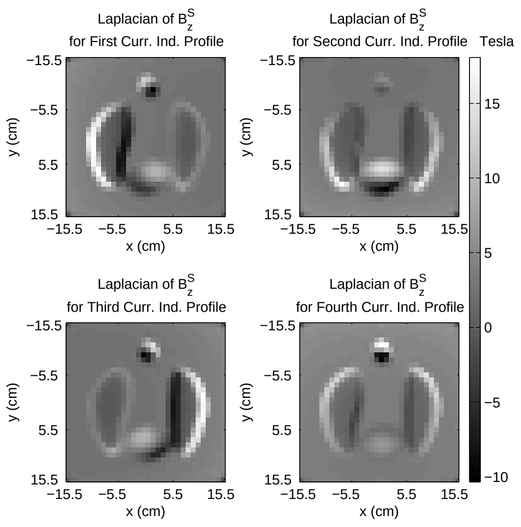
$$\begin{matrix} \text{Ly=P}b_k \\ \text{U}x_k\text{=y} \end{matrix} \quad \text{for k = 1,2,...,N} \tag{4.2}$$



(a)

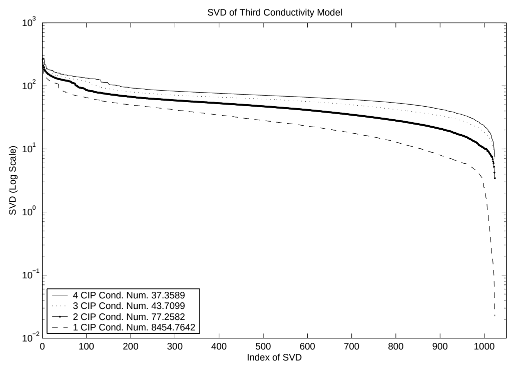


(b)

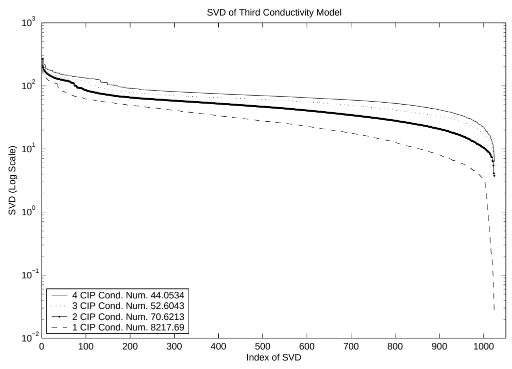


(c)

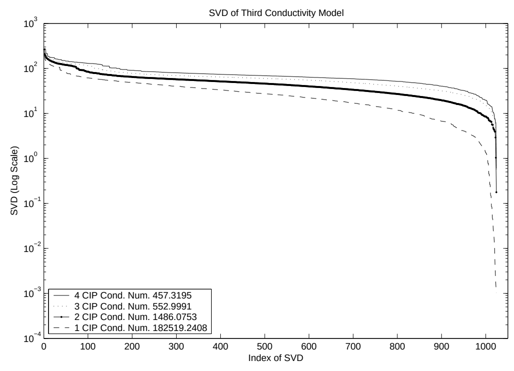
Figure 4.9: Laplacian of secondary magnetic flux density images for three conductivity models given in (a) to (c).



(a)



(b)



(c)

Figure 4.10: Singular values of combined system matrix for one, two, three and four current induction profiles (CIP) for three conductivity models given in (a) to (c).

where P is a permutation matrix, L is lower triangular with unit diagonal elements, and U is upper triangular. The resulting x_k vectors are the columns of the inverse matrix.

4.4 Computational Cost of Algorithm

The computation time of the algorithm and the required hardware sources such as memory and application specific software platform are important for the implementation of the system. There are many system possibilities that can be used to optimize the calculation time of the reconstructed data. However, it may be useful for the reader to give the system we used in this thesis to understand better the computational load of the algorithm.

For simulations, a general purpose PC with an hardware configuration of a Pentium IV CPU at 2.8 GHz, 1 GByte DDR-RAM and 500 MHz System Bus is used. The operating system on the PC is Windows XP Professional Edition and the interpreter used for simulations is MATLAB R13. Since MATLAB is just an interpreter, vectorization is made for better performance of the code. However, the magnetic flux density calculation part of the code cannot be vectorized and have many loops inside. In order to increase the performance of the code we have written, we used the ability of MATLAB to call C functions. The source codes of routines are given in Appendix F.

For the algorithm in this thesis, computation of single iteration takes 82 minutes approximately. In one iteration, 4 current densities for each current induction profiles are calculated and each one takes about 19 minutes. The primary and secondary magnetic flux densities and the vector magnetic potentials used in the simulations are calculated within 6 hours.

Chapter 5

Conclusion and Future Work

In this thesis, a new reconstruction algorithm is proposed for a novel imaging technique named as Induced Current Magnetic Resonance-Electrical Impedance Tomography. This algorithm is proposed for the reconstruction of conductivity distribution of objects in three dimensions. The algorithm is based on the induction of a conductive object by a coil and generating induced currents inside of the object. These induced currents will create a new magnetic flux density to reject the magnetic flux density generated by the coil. The relation between magnetic flux densities enables to visualize the conductivity distribution by using some image reconstruction techniques.

The advantage of this method is measurement noise due to electrode will not occur. However, finding a good coil is an essential problem since a bad configuration of coil will not give good results. In this thesis, we used a coil configuration which is just a loop of wire. Designing a better coil configuration is another concept to find.

In the future, it can be attempted to try to find better coil configurations, to improve our numerical results and to apply some other numerical methods to reconstruct the conductivity distribution. In addition, the MATLAB codes used in simulations can be converted to C/C++ routines to decrease the computational cost of the method. In addition, we plan to look at the Picard plots and L-curves

[34] of the coefficient matrices. It is possible to select a singular value level that will be a threshold for the singular values of coefficient matrix by using these two plots.

Bibliography

[1] Geddes L A and Baker L E, "The Specific Resistance of Biological Materials", *Med. Biol. Eng.*, vol. 5, pp. 271-293, 1967.

[2] Pethig R, "Dielectric Properties of Biological Materials", *John Wiley and Sons Ltd.*, New York, 1979.

[3] Pethig R, "Dielectric Properties of Body Tissues", *Clin. Phys. Physiol. Meas.*, vol. 8, suppl. A, pp. 5-12, 1987.

[4] Pethig R and Kell D B, "The Passive Electrical Properties of Biological Systems: Their Significance in Physiology, Biophysics and Biotechnology", *Phys. Med. Biol.*, vol. 32, pp. 933-970, 1987.

[5] Boone K, Barber D C and Brown B H, "Imaging with electricity: Report of the Europan concerned action on impedance tomography", *Journal of Medical Engineering and Technology*, vol. 21, no. 6, pp. 201-232, 1997.

[6] Sylvester J and Uhlmann G, "A uniqueness theorem for an inverse boundary value problem in electrical prospection", *Commun. Pure Appl. Math.*, vol. 17, pp. 91-112, 1986.

[7] Korjenevsky A and Sapetsky S C, "Magnetic induction tomography: experimental realization", *Physiol. Meas.*, vol. 21, pp. 8994, 2000.

[8] Kak A C, Slaney M, "Principles of computerized tomographic imaging", *SIAM*, New York, 1988.

[9] Wahl R L, Buchanan J W, "Principles and Practice of Positron Emission Tomography", *Lippincott Williams & Wilkins*, Philadelphia, 2002.

[10] Seagar A D, Baber D C and Brown B H "Theoretical Limits to Sensitivity and Resolution in Impedance Imaging", *Clin. Phys. Physiol. Meas.*, vol. 8, suppl. A, pp. 13-31, 1987. vol. 21, no. 6, pp. 201-232, 1997.

[11] Köksal A and Eyüboğlu B M, "Determination of optimum injected current patterns in electrical impedance tomography", *Clin. Phys. Physiol. Meas.*, vol. 16, suppl. A, pp. 99-109, 1995.

[12] Eyüboğlu B M, Köksal A and Demirbilek B M, "Distinguishability analysis of an induced current EIT system using discrete coils", *Phys. Med. Biol.* vol. 45, pp. 19972009, 2000.

[13] Zhang N, "Electrical impedance tomography based on current density imaging", Master's Thesis, University of Toronto, Dept. of Electrical and Electronics Eng., Toronto, Canada, 1992.

[14] Eyüboğlu B M, Reddy R and Leigh J S, "Magnetic resonance - electrical impedance tomography" *patent no: US6,397,095 B1, provisional patent application on Mar.1 1999*, 2002.

[15] Özdemir M and Eyüboğlu B M, "Novel fast reconstruction algorithm for magnetic resonance electrical impedance tomography", *Proc. of Biyomut* 8th *Annual Conference*, CD-ROM, Istanbul, Turkey, 2002.

[16] Kwon O, Lee J Y and Yoon J R, "Equipotential line method for magnetic resonance electrical impedance tomography" *Inverse Probl.*, vol. 18, pp. 1089-1100, 2002.

[17] Woo E J, Lee S Y , and Mun C W, "Impedance tomography using current density distribution measured by nuclear magnetic resosnance", *SPIE*, vol. 2299, pp. 377-385, 1994.

[18] Kwon O, Woo E J, Yoon J R, and Seo J K, "Magnetic Resonance Electrical Impedance Tomography (MREIT): Simulation Study of J-Substitution Algorithm", *IEEE Trans. on Biomed. Eng.*, vol. 49-2, pp. 160-167, 2002.

[19] Kim S, Kwon O, Seo J K, and Yoon J-R, "On a Nonlinear Partial Differential Equation arising in Magnetic Resonance Electrical Impedance Tomography" *SIAM J. MATH. ANAL.*, vol. 34, pp. 511-526, 2002.

[20] Khang H S, Lee B I, Oh S H, Woo E J, Lee S Y, Cho M H, Kwon O, Yoon J R and Seo J K, "J-substitution algorithm in magnetic resonance electrical impedance tomography (mreit): Phantom experiments for static resistivity images", *IEEE Trans. on Med. Imag.*, Vvol. 21, pp. 695-702, 2002.

[21] Eyüboğlu B M, Birgül Ö, and İder Y Z, "Dual modality system for high-resolution true conductivity imaging", *XI Int. Conf. Elec. Bioimpedance*, Oslo, Norway pp. 409-413, 2001.

[22] Birgül Ö, Eyüboğlu B M and İder Y Z, "Current constrained voltage scaled reconstruction (CCVSR) algorithm for MR-EIT and its performance with different probing current patterns", *Phys. Med. Biol.*, vol. 48 pp. 653-671, 2003.

[23] İder Y Z and Müftüler T, "Measurement of ac magnetic field distribution using magnetic resonance imaging", *IEEE Trans. Med. Imaging*, vol. 16, pp. 617-22, 1997.

[24] İder Y Z and Birgül Ö, "Use of magnetic field generated by the internal distribution of injected currents for electrical impedance tomography (mr-eit)", *Elektrik, Turkish J. of Elec. Eng. and Comp. Sci.*, vol. 6, pp. 215-225, 1998.

[25] Seo J K, Kwon O, Yoon J-R, Lee B I and Woo E J, "Single Component of Magnetic Flux Density can produce Resistivity Images in Magnetic Resonance Electrical Impedance Tomography (MREIT)", abstract, *First Mummy Range Workshop on Electrical Impedance Imaging*, Pingree Park, Colorado, 2002.

[26] İder Y Z, Onart S, Lionheart W R B, "Uniqueness and reconstruction in magnetic resonance-electrical impedance tomography (MR-EIT)", *Physiol. Meas.*, vol. 24, pp. 591-604, 2003.

[27] İder Y Z, Onart S, "Algebraic Reconstruction for 3D MR-EIT using one component of magnetic flux density", submitted to *Physiol. Meas.*, August, 2003.

[28] Marshall, S V, Skitek, G G,"Electromagnetic Concepts and Applications", *Prentice Hall Press*, Englewood Cliffs, N.J., 1990.

[29] Gencer N, Kuzuoğlu M, İder Y Z, "Electrical Impedance Tomography Using Induced Currents", *IEEE Transaction on Medical Imaging*, Vol. 13, No.2, pp.338-350, June 1994.

[30] Kriezis E E, Tsiobukis T D, Panas S M, Tegopoulos J A, "Eddy Currents: Theory and Applications", *Proc. IEEE*, Vol. 80, pp.1559-1589, Oct. 1992.

[31] David B, George C, "Basic partial differential equations", *Van Nostrand Reinhold Company*, New York, 1992.

[32] Scott G C, Joy M L G, Armstrong R L and Hankelman R M, "Sensitivity of magnetic resonance current density imaging", *Jour. of Mag. Res.*, vol. 97, pp. 235-254, 1992.

[33] Underwriters Laboratories, American National Standard for Leakage Current for Appliances, ANSI C101-1992, March. 1992.

[34] Hansen P C, "Regularization Tools", http://www.imm.dtu.dk/ pch, 2001.

[35] Micac U, Demsar F, Beravs K, Sersa I, "Magnetic Resonance Imaging of Alternating Currents", *Magnetic Resonance Imaging 19*, pp.845-856, 2001.

[36] Scott G C, Joy L G, Armstrong R L, Henkelman R M, "Measurement of Nonuniform Current Density by Magnetic Resonance", *IEEE Transaction on Medical Imaging*, Vol. 10, No.3, pp.362-374, September 1991.

Appendix A

Potentials and Fields due to Coil

A.1 Magnetic Flux Density

Let *(x',y',z')* be the source point on a coil and *(x,y,z)* be the observation point, the solution for the magnetic flux density $\bar{B}$ can be found from Biot-Savart rule:

$$\bar{B}(x,y,z) = \frac{\mu_0}{4\pi} \oint_C \frac{\bar{J} \times \bar{R}}{R^3} dl' \tag{A.1}$$

where $\bar{J} = I\hat{\phi}$, I is the current flowing through the wire, $R = \sqrt{(x-x')^2 + (y-y')^2 + (z-z')^2}$ and dl' represents the differential element on the curve C. In this study, the integration given in the above equation is approximated numerically for B_x, B_y and B_z by

$$B_x(x,y,z) = \frac{\mu_0 I a}{2N} \sum_{n=1}^{N} \frac{z sin(n\frac{2\pi}{N})}{R^3} \tag{A.2}$$

$$B_y(x,y,z) = \frac{\mu_0 I a}{2N} \sum_{n=1}^{N} \frac{z cos(n\frac{2\pi}{N})}{R^3} \tag{A.3}$$

$$B_z(x,y,z) = -\frac{\mu_0 I a}{2N} \sum_{n=1}^{N} \frac{x cos(n\frac{2\pi}{N}) + y sin(n\frac{2\pi}{N}) - a}{R^3} \tag{A.4}$$

where N denotes the number of integration points and a is the coil radius.

A.2 Vector Magnetic Potential

Let *(x',y',z')* be the source point on a coil and *(x,y,z)* be the observation point, the solution for the magnetic vector potential $\bar{A}$ is

$$\bar{A}(x, y, z) = \frac{\mu_0}{4\pi} \oint_C \frac{I}{R} d\bar{l}' \tag{A.5}$$

where $d\bar{l}'$ represents the differential vector element on the curve C. In this study, the integration given in the above equation is approximated numerically for A_x and A_y by

$$A_x(x, y, z) = \frac{\mu_0 I a}{2N} \sum_{n=1}^{N} \frac{-sin(n\frac{2\pi}{N})}{R} \tag{A.6}$$

$$A_y(x, y, z) = \frac{\mu_0 I a}{2N} \sum_{n=1}^{N} \frac{cos(n\frac{2\pi}{N})}{R} \tag{A.7}$$

Appendix B

Some Properties of a Tetrahedron

In this Appendix, some properties of tetrahedrons are represented, which are used in the solution of the forward problem.

B.1 Surface Normals of a Tetrahedron

In order to find the surface normals of a tetrahedron, defining two vectors over the surface of that normal to be found is enough. These two vectors can be the vectors from first corner to second corner and first corner to third corner. The normalized version of the cross product of these two vectors is equal to normal. However, this normal is in-directed or out-directed to the tetrahedron. For convenience, all normals must be in-directed or out-directed. In this thesis, normals is taken to be out-directed. To ensure that the normals are out-directed, a dot product between normal vector and a vector from the center of tetrahedron to the center of the surface is taken. If result is positive then normal is out-directed and no modification is required. If result is negative then normal is in-directed and must be multiplied with -1.

B.2 Surface Areas of a Tetrahedron

Surface Area of a tetrahedron is equal to the half length of the vector created by the cross product defined above.

Appendix C

Finite Difference Formulation

The discretized version of the derivative is finite difference. If a function is sampled with a distance of small h at P points, then new sampled values define a new discrete function related to the first function with the below relation.

$$f[p] \equiv f(x_0 + ph) \quad \text{for } p = 1, 2, ..., P. \tag{C.1}$$

Here, it is possible to define lots of finite difference formulas. We will define three of them. First is the *finite forward difference* of function f[p] defined as,

$$\Delta f_p = f[p+1] - f[p] \quad \text{for } p = 1, 2, ..., P-1. \tag{C.2}$$

Finite backward difference is another method to find the finite difference of function f[p] and is defined as,

$$\Delta f_p = f[p] - f[p-1] \quad \text{for } p = 2, 3, ..., P. \tag{C.3}$$

Note that, the first index of f[p] for backward finite difference and the last index of f[p] for forward finite difference are note defined. Third and last of the defined method for finite difference is central finite difference of function f[p] is defined as,

$$\delta f_p = f[p+1] - f[p-1] \quad \text{for } p = 2, 3, ..., P-1. \tag{C.4}$$

First two method have a sample distance of *h*, but third one have a sample distance of *2h*. A derivative is defined as the limit value between two points over the distance between these points. Starting from here, a first order approximation to the derivative *f'x* is,

$$f'(x) = \frac{df(x)}{dx} = \frac{\Delta f_p}{h} + O(h) \tag{C.5}$$

with forward and backward finite differences. In both cases, the order of approximation error is order of *h*. *Central finite difference* is a more accurate discrete approximation for *f'(x)*.

$$f'(x) = \frac{\delta f_p}{2h} + O(h^2) \tag{C.6}$$

Approximation error is in order of h^2 for central difference formulation. This means that halving *h* decreases the error to a quarter. Central finite difference formulation is not defined for edges, it can only be useful for points between first and last points.

Appendix D

Proposed Pulse Sequence for Measuring Secondary Magnetic Flux Density Using MRI

Measurement of B_{S_z} can be achieved using MR-CDI techniques. In this study, the magnetic flux density generated by the AC eddy current density is to be measured. In [35], an MRI pulse sequence for measuring the magnetic flux density of currents obtained by applied square wave voltages of about 1 KHz frequency is developed. This MR pulse sequence can be adapted to the particular case in this study as shown in Figure D.1. During AC excitation, phase of the MR image accumulates due to B_{S_z} provided that N RF pulses are applied at the times of the zero crossings of B_{S_z}. Since B_{S_z} and B_{P_z} are out-of-phase, phase accumulation due to B_{P_z} in a positive half cycle cancels with the accumulation during a negative half cycle, and only the contribution of B_{S_z} remains. This is a fundamental assumption which may be difficult to achieve in an experimental setting.

In order to get the magnetic flux density data, two experiments are done with different current [36]. First, an in-phase current to the secondary magnetic flux density is applied and a complex image is gathered. Then, a current, which has

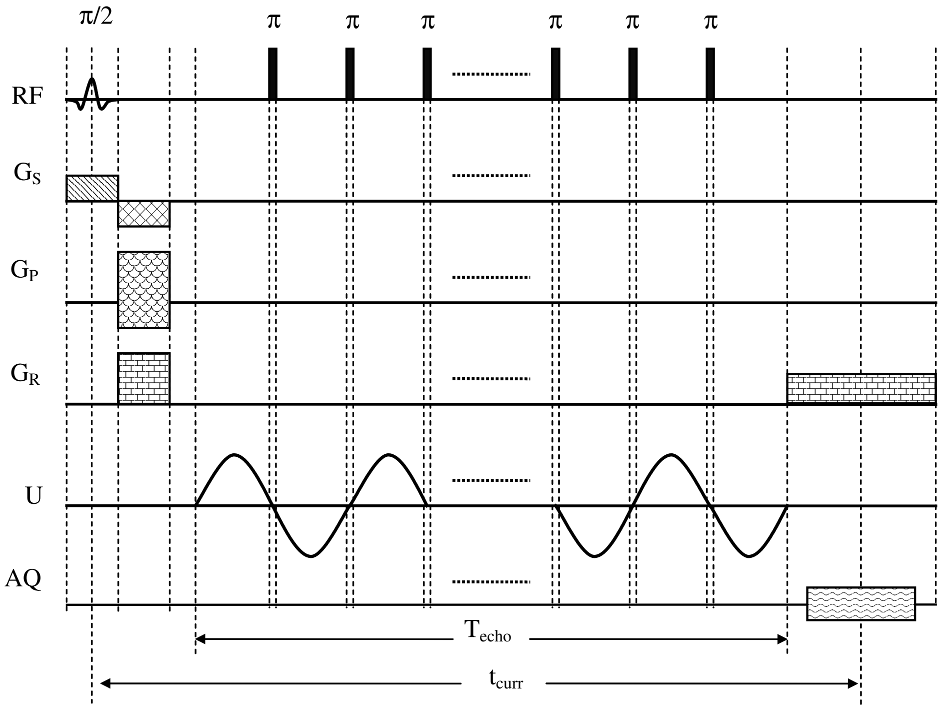


Figure D.1: Spin Echo Experiment for Induced Current MR-EIT.

a 180^o phase from the secondary magnetic flux density, is applied and another complex image is gathered. These two images have an extra phase term from the no-current case image as given below,

$$M_{c+}(x,y,z) \simeq M(x,y,z)\ \ exp\ \ [j(-1)^N(N+1)\int_0^{\frac{T_{echo}}{N+1}} \gamma B_{S_z}(x,y,z)dt] \tag{D.1}$$

$$M_{c-}(x,y,z) \simeq M(x,y,z)\ \ exp\ \ [-j(-1)^N(N+1)\int_0^{\frac{T_{echo}}{N+1}} \gamma B_{S_z}(x,y,z)dt] \tag{D.2}$$

where M(x,y,z) is the complex image that is gathered from standard the spin echo experiment with no current,M_{c+} and M_{c-} are the complex images gathered from in-phase and opposite-phase current applications and T_{echo} is the duration of total AC excitation. The integral term can be calculated easily due to the fact that it is only an integral of sine function. In this case, the equations become

$$M_{c+}(x,y,z) \simeq M(x,y,z)\ \ exp\ \ [j(-1)^N\gamma \left\langle B_{S_z}(x,y,z)\right\rangle \frac{2T_{echo}}{\pi}] \tag{D.3}$$

$$M_{c-}(x,y,z) \simeq M(x,y,z)\ \ exp\ \ [-j(-1)^N \gamma \left\langle B_{S_z}(x,y,z) \right\rangle \frac{2T_{echo}}{\pi}]. \tag{D.4}$$

where $\langle B_{S_z} \rangle$ is the average value of B_{S_z} during the half period of AC current. If we look at the ratios of these two images, it gives a phase term. From Eqs. D.3 and D.4, it is easy to see that this phase term is a constant times secondary magnetic flux density itself. The accumulated phase shift is then obtained as

$$\varphi_S = 2(-1)^N \gamma \left\langle B_{S_z}(x,y,z) \right\rangle \frac{2T_{echo}}{\pi} \tag{D.5}$$

Here, T_{echo} can be calculated easily since we know the frequency of the AC current and N. The duration of the applied AC current is

$$T_{echo} = \frac{N+1}{2} T_{AC} \tag{D.6}$$

where T_{AC} is the period of AC current.

Appendix E

Error Analysis for Calculation of Secondary Magnetic Flux Density

In this thesis, magnetic flux densities are calculated numerically. However, these calculations have some numerical errors. In the calculation of primary magnetic flux density, the error can be less than a desired tolerance by increasing the calculation number of points over the coil (N) given in Eqs. A.2, A.3, A.4. When we apply the Biot-Savart integral for the calculation of primary magnetic flux density, the approximation given in Eq. 2.30 can be easily applied. Since the coil is enough far from the object, the error will be negligible. However, in the case of secondary magnetic flux density, this is not exactly true. As we mentioned before, the error due to close points, i.e. the points sharing the same cube, will be negligible. On the other hand, the cubes, which are placed on the upper, lower, right and left part of the current cube, will also create an error term. In the middle of the object, the points in the upper cube will destroy the effect of the points in the lower cube. In the same way, the points in the left cube will do the same thing for the points of right one. However, as we get closer to the edges, especially wedges, the error will increase. While we are in one of the wedges that has only lower and left cube neighbors, the effect of lower and left cube will not

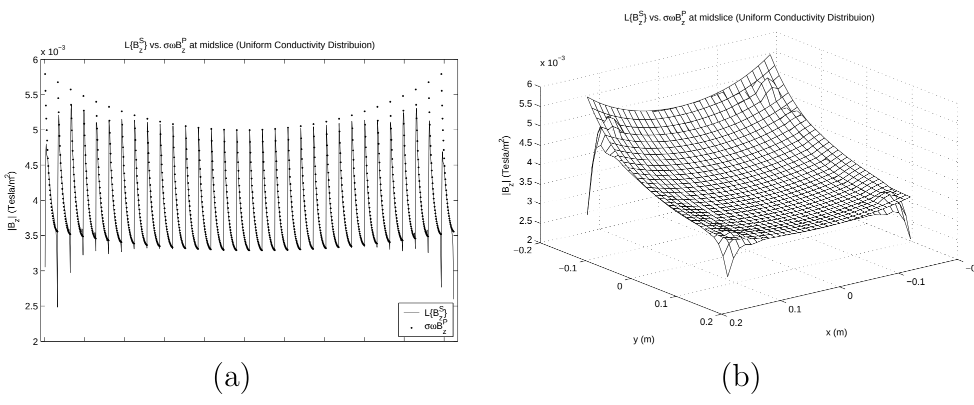


Figure E.1: Error due to Biot-Savart Integral (a) 2D illustration of laplacian of B_{S_z} versus $\sigma_0 \omega B_{P_z}$. (b) 3D laplacian of B_{S_z} versus $\sigma_0 \omega B_{P_z}$.

be canceled by an upper or right cube. These errors will be seen in the magnetic flux density data.

The effect of these errors can be easily seen in the uniform conductivity distribution case. Since the conductivity distribution is constant, the gradient of ρ will be equal to 0. In Eq. 3.12, if we replace 0 instead of gradient term and ρ_0 instead of ρ, the equation becomes

$$\rho_0 \nabla^2 \frac{\bar{B}_{S_I}}{\mu_0} = w \bar{B}_{P_R}. \tag{E.1}$$

The laplacian term is a constant times primary magnetic flux density for the uniform distribution case, from the equation. In the inverse problem, we only consider the z component of both fields. In Figure E.1, the primary magnetic flux density and the laplacian term is computed for the uniform distribution case.

Appendix F

MATLAB Source Codes

In this section, source codes used in the simulations for both forward solver and inverse solver are given. The codes are written in MATLAB. The forward solver uses the routines SolverForward, magvecpot3d, SolverPlotter, SolverBody1 and findnodeBfield2. The inverse solver uses the routines SolverForward, SolverBody2 and SolverInverse.

Listing F.1: SolverBody1

```
close all;
warning on;
% Load Mesh
Nodenum = 70785;
load(['mesh3dtet' num2str(Nodenum) '.mat']);
%load Org_Sigma.mat;
% Define globals
% Number of Nodes
global nodes
% Number of Triangles (Elements)
global nelmts
% X coordinates of nodes
global x
% Y coordinates of nodes
global y
% Y coordinates of nodes
global z
% Triangle element nodes
global elmn
global sigma
global radius
% Define default values for variables
w = 2 * pi * 1000; % radian/sec
K = floor(xmax/4);
Time0 = clock;
radius = 3*max(x)
```

```
abcd = zeros(nelmts,4,4);

% Calculate Vector Magnetic Potential, abcd and Volume for Each Tetrahedron
n = zeros(nelmts,4,3);
for i = 1:6
    % Coordinate Matrix
    matrix1 = [1 x(elmn(i,1)) y(elmn(i,1)) z(elmn(i,1))
        1 x(elmn(i,2)) y(elmn(i,2)) z(elmn(i,2))
        1 x(elmn(i,3)) y(elmn(i,3)) z(elmn(i,3))
        1 x(elmn(i,4)) y(elmn(i,4)) z(elmn(i,4))];% m
    % Coefficients of Nji
    abcd1(i,:,:) = inv(matrix1);% 1/m
    % Volume of Element (Tetrahedron)
    Volume1(i) = abs(det(matrix1)/6);% m^3
    % Calculate the normal to the surface of tetrahedron
    % Node coordinates
    node1(i,:) = [x(elmn(i,1)) y(elmn(i,1)) z(elmn(i,1))];% m
    node2(i,:) = [x(elmn(i,2)) y(elmn(i,2)) z(elmn(i,2))];% m
    node3(i,:) = [x(elmn(i,3)) y(elmn(i,3)) z(elmn(i,3))];% m
    node4(i,:) = [x(elmn(i,4)) y(elmn(i,4)) z(elmn(i,4))];% m
    % Vectors defining the edges and their lengths
    P1(i,:) = node1(i,:) - node2(i,:);% m
    P1l(i) = norm(P1(i,:));% m
    P2(i,:) = node1(i,:) - node3(i,:);% m
    P2l(i) = norm(P2(i,:));% m
    P3(i,:) = node1(i,:) - node4(i,:);% m
    P3l(i) = norm(P3(i,:));% m
    P4(i,:) = node2(i,:) - node3(i,:);% m
    P4l(i) = norm(P4(i,:));% m
    P5(i,:) = node2(i,:) - node4(i,:);% m
    P5l(i) = norm(P5(i,:));% m
    P6(i,:) = node3(i,:) - node4(i,:);% m
    P6l(i) = norm(P6(i,:));% m
    % Mid points of Surfaces
    mid1(i,:) = (node1(i,:) + node2(i,:) + node3(i,:))/3;% m
    mid2(i,:) = (node1(i,:) + node2(i,:) + node4(i,:))/3;% m
    mid3(i,:) = (node1(i,:) + node4(i,:) + node3(i,:))/3;% m
    mid4(i,:) = (node4(i,:) + node2(i,:) + node3(i,:))/3;% m
    % Cross Products to find normals of 4 surface of tetrahedron
    Pc1(i,:) = cross(P1(i,:),P2(i,:));% m
    Pc2(i,:) = cross(P1(i,:),P3(i,:));% m
    Pc3(i,:) = cross(P2(i,:),P3(i,:));% m
    Pc4(i,:) = cross(P4(i,:),P5(i,:));% m
    % Decide signs of normal vectors
    Signs(i,:) = sign([Pc1(i,:)*(mid1(i,:) - node4(i,:)).'
                       Pc2(i,:)*(mid2(i,:) - node3(i,:)).'
                       Pc3(i,:)*(mid3(i,:) - node2(i,:)).'
                       Pc4(i,:)*(mid4(i,:) - node1(i,:)).']).';
    % Normals of surfaces
    n1(i,:,:) = [Signs(i,1)*Pc1(i,:)/norm(Pc1(i,:))
                 Signs(i,2)*Pc2(i,:)/norm(Pc2(i,:))
                 Signs(i,3)*Pc3(i,:)/norm(Pc3(i,:))
                 Signs(i,4)*Pc4(i,:)/norm(Pc4(i,:))];
    % Surface areas of tetrahedron
    S11(i) = norm(Pc1(i,:))/2;% m^2
    S21(i) = norm(Pc2(i,:))/2;% m^2
    S31(i) = norm(Pc3(i,:))/2;% m^2
    S41(i) = norm(Pc4(i,:))/2;% m^2
end
n = repmat(n1,nelmts/6,1);
disp('Surface Normals Calculated!');
S1 = repmat(S11,1,nelmts/6);% m^2
S2 = repmat(S21,1,nelmts/6);% m^2
S3 = repmat(S31,1,nelmts/6);% m^2
S4 = repmat(S41,1,nelmts/6);% m^2
disp('Surface Areas Calculated!');
abcd = repmat(abcd1,nelmts/6,1);% 1/m
disp('Basis Function Coefficients Calculated!');
```

```
Volume = repmat(Volume1,1,nelmts/6);% m^3
disp('Volumes of Tetrahedrons Calculated!');

Time1 = clock;

Ant1 = zeros(nelmts,4,3);
Ant2 = zeros(nelmts,4,3);
Ant3 = zeros(nelmts,4,3);
Ant4 = zeros(nelmts,4,3);
Ann1 = zeros(nelmts,4);
Ann2 = zeros(nelmts,4);
Ann3 = zeros(nelmts,4);
Ann4 = zeros(nelmts,4);
midpoints = zeros(nelmts,3);
midsurfs = zeros(nelmts,4,3);
Bp1 = zeros((xmax-1)*(ymax-1)*(zmax-1),3);
Bp2 = zeros((xmax-1)*(ymax-1)*(zmax-1),3);
Bp3 = zeros((xmax-1)*(ymax-1)*(zmax-1),3);
Bp4 = zeros((xmax-1)*(ymax-1)*(zmax-1),3);

% Calculate Midpoints of Tetrahedrons
midpoints = [sum(x(elmn(:,:)),2) sum(y(elmn(:,:)),2) sum(z(elmn(:,:)),2)]/4;
midsurfs(:,1,:) = [x(elmn(:,1))+x(elmn(:,2))+x(elmn(:,3))
                   y(elmn(:,1))+y(elmn(:,2))+y(elmn(:,3))
                   z(elmn(:,1))+z(elmn(:,2))+z(elmn(:,3))].';
midsurfs(:,2,:) = [x(elmn(:,1))+x(elmn(:,2))+x(elmn(:,4))
                   y(elmn(:,1))+y(elmn(:,2))+y(elmn(:,4))
                   Z(elmn(:,1))+z(elmn(:,2))+z(elmn(:,4))].';
midsurfs(:,3,:) = [x(elmn(:,1))+x(elmn(:,3))+x(elmn(:,4))
                   y(elmn(:,1))+y(elmn(:,3))+y(elmn(:,4))
                   z(elmn(:,1))+z(elmn(:,3))+z(elmn(:,4))].';
midsurfs(:,4,:) = [x(elmn(:,2))+x(elmn(:,3))+x(elmn(:,4))
                   y(elmn(:,2))+y(elmn(:,3))+y(elmn(:,4))
                   z(elmn(:,2))+z(elmn(:,3))+z(elmn(:,4))].';
disp('Mid-Points of Volume and Surfaces Calculated!');

for i = 1:nelmts
    if i == 1
        disp(nelmts)
    elseif mod(i,10000)==0
        disp(i)
    end
    xp = radius/3;
    yp = 0;
    zp = 0;% m
    [Ax,Ay,Az] = magvecpot3d(i,xp,yp,zp);% mWb/m
    Ant1(i,:,:) = [Ax Ay Az];%x-y plane% mWb/m
    Ann1(i,1:4) = (Ax.'.*n(i,:,1)+Ay.'.*n(i,:,2)+Az.'.*n(i,:,3));% mWb/m
    xp = 0;
    yp = radius/3;
    zp = 0;% m
    [Ax,Ay,Az] = magvecpot3d(i,xp,yp,zp);% mWb/m
    Ant2(i,:,:) = [Ax Ay Az];%x-z plane% mWb/m
    Ann2(i,1:4) = (Ax.'.*n(i,:,1)+Ay.'.*n(i,:,2)+Az.'.*n(i,:,3));% mWb/m
    xp = -radius/3;
    yp = 0;
    zp = 0;% m
    [Ax,Ay,Az] = magvecpot3d(i,xp,yp,zp);% mWb/m
    Ant3(i,:,:) = [Ax Ay Az];%x-y plane% mWb/m
    Ann3(i,1:4) = (Ax.'.*n(i,:,1)+Ay.'.*n(i,:,2)+Az.'.*n(i,:,3));% mWb/m
    xp = 0;
    yp = -radius/3;
    zp = 0;% m
    [Ax,Ay,Az] = magvecpot3d(i,xp,yp,zp);% mWb/m
    Ant4(i,:,:) = [Ax Ay Az];%x-z plane% mWb/m
    Ann4(i,1:4) = (Ax.'.*n(i,:,1)+Ay.'.*n(i,:,2)+Az.'.*n(i,:,3));% mWb/m
end
Ap1 = reshape(sum(Ant1,2)/4,nelmts,3);% mWb/m
```

```
Ap2 = reshape(sum(Ant2,2)/4,nelmts,3);% mWb/m
Ap3 = reshape(sum(Ant3,2)/4,nelmts,3);% mWb/m
Ap4 = reshape(sum(Ant4,2)/4,nelmts,3);% mWb/m
disp('Vector Magnetic Potentials Calculated!');

Time2 = clock;

% Cylinder
for q =1:6:nelmts-5
    dist = sum(midpoints(q:q+5,:))/6;
    P = max(x)/2;
    if sqrt(((dist(1)+radius/12)/P)^2 + (dist(2)/P)^2)<= 1
        sigma(q:q+5,1) = 0.6;
    else
        sigma(q:q+5,1) = 0.2;
    end
end

Org_Sigma = sigma;
save(['OrgSigma' num2str(Nodenum) '.mat'],'Org_Sigma');
disp('Sigma Created!');

save(['MagVec' num2str(Nodenum) '.mat'],'Ann1','Ant1','Ap1','Ann2','Ant2','Ap2','Ann3','Ant3','Ap3','Ann4','Ant4','Ap4');
save (['Imports' num2str(Nodenum) '.mat'],'midpoints','abcd','Volume','S1','S2','S3','S4','n');
Time3 = clock;
Ant = Ant1;
Ann = Ann1;
Ap = Ap1;
SolverForward;
J1 = J;
phical1 = phiCalculated;
disp('SolverForward 1');

Time4 = clock;
xp = radius/3;
yp = 0;
zp = 0;
SolverPlotter;

B1 = B;
Bp1 = Bp;
disp('SolverPlotter 1');

Time5 = clock;
Ant = Ant2;
Ann = Ann2;
Ap = Ap2;
SolverForward;
J2 = J;
phical2 = phiCalculated;
disp('SolverForward 2');

Time6 = clock;
xp = 0;
yp = radius/3;
zp = 0;
SolverPlotter;

B2 = B;
Bp2 = Bp;
disp('SolverPlotter 2');

Time7 = clock;
Ant = Ant3;
Ann = Ann3;
Ap = Ap3;
SolverForward;
J3 = J;
```

```
phical3 = phiCalculated;
disp('SolverForward 3');

Time8 = clock;
xp = -radius/3;
yp = 0;
zp = 0;
SolverPlotter;

B3 = B;
Bp3 = Bp;
disp('SolverPlotter 3');

Time9 = clock;
Ant = Ant4;
Ann = Ann4;
Ap = Ap4;
SolverForward;
J4 = J;
phical4 = phiCalculated;
disp('SolverForward 4');

Time10 = clock;
xp = 0;
yp = -radius/3;
zp = 0;
SolverPlotter;

B4 = B;
Bp4 = Bp;
disp('SolverPlotter 4');

Time11 = clock;
save(['Bsz' num2str(Nodenum) '.mat'],'B1','B2','B3','B4');
save(['Bpz' num2str(Nodenum) '.mat'],'Bp1','Bp2','Bp3','Bp4');

K = floor(etime(Time1,Time0)/3600);
L = floor((etime(Time1,Time0)-3600*K)/60);
M = etime(Time1,Time0) - 3600*K - 60*L;
disp(['Normals,Surface Areas : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time2,Time1)/3600);
L = floor((etime(Time2,Time1)-3600*K)/60);
M = etime(Time2,Time1) - 3600*K - 60*L;
disp(['Calculate VMPs : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time3,Time2)/3600);
L = floor((etime(Time3,Time2)-3600*K)/60);
M = etime(Time3,Time2) - 3600*K - 60*L;
disp(['Calculate Primary B Field : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time4,Time3)/3600);
L = floor((etime(Time4,Time3)-3600*K)/60);
M = etime(Time4,Time3) - 3600*K - 60*L;
disp(['Forward Solver 1 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time5,Time4)/3600);
L = floor((etime(Time5,Time4)-3600*K)/60);
M = etime(Time5,Time4) - 3600*K - 60*L;
disp(['Solver Plotter 1 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time6,Time5)/3600);
L = floor((etime(Time6,Time5)-3600*K)/60);
M = etime(Time6,Time5) - 3600*K - 60*L;
disp(['Forward Solver 2 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time7,Time6)/3600);
L = floor((etime(Time7,Time6)-3600*K)/60);
M = etime(Time7,Time6) - 3600*K - 60*L;
disp(['Solver Plotter 2 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time8,Time7)/3600);
L = floor((etime(Time8,Time7)-3600*K)/60);
M = etime(Time8,Time7) - 3600*K - 60*L;
disp(['Forward Solver 3 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
```

```
K = floor(etime(Time9,Time8)/3600);
L = floor((etime(Time9,Time8)−3600∗K)/60);
M = etime(Time9,Time8) − 3600∗K − 60∗L;
disp(['Solver Plotter 3 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time10,Time9)/3600);
L = floor((etime(Time10,Time9)−3600∗K)/60);
M = etime(Time10,Time9) − 3600∗K − 60∗L;
disp(['Forward Solver 4 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time11,Time10)/3600);
L = floor((etime(Time11,Time10)−3600∗K)/60);
M = etime(Time11,Time10) − 3600∗K − 60∗L;
disp(['Solver Plotter 4 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time11,Time0)/3600);
L = floor((etime(Time11,Time0)−3600∗K)/60);
M = etime(Time11,Time0) − 3600∗K − 60∗L;
disp(['Total : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
```

Listing F.2: SolverForward

```
% Number of Nodes
global nodes
% Number of Triangles (Elements)
global nelmts
% X coordinates of nodes
global x
% Y coordinates of nodes
global y
% Y coordinates of nodes
global z
% Triangle element nodes
global elmn
global sigma

b = sparse(nodes,1);
S = sparse(nodes,nodes);
% Iteration number to find Ax and Ay
for i = 1:nelmts
    SmallS = sigma(i) ∗ Volume(i)∗[abcd(i,2,1)∗abcd(i,2,1)+abcd(i,3,1)∗abcd(i,3,1)+abcd(i,4,1)∗abcd(i,4,1)...
              abcd(i,2,1)∗abcd(i,2,2)+abcd(i,3,1)∗abcd(i,3,2)+abcd(i,4,1)∗abcd(i,4,2)...
              abcd(i,2,1)∗abcd(i,2,3)+abcd(i,3,1)∗abcd(i,3,3)+abcd(i,4,1)∗abcd(i,4,3)...
              abcd(i,2,1)∗abcd(i,2,4)+abcd(i,3,1)∗abcd(i,3,4)+abcd(i,4,1)∗abcd(i,4,4)
                        abcd(i,2,2)∗abcd(i,2,1)+abcd(i,3,2)∗abcd(i,3,1)+abcd(i,4,2)∗abcd(i,4,1)...
              abcd(i,2,2)∗abcd(i,2,2)+abcd(i,3,2)∗abcd(i,3,2)+abcd(i,4,2)∗abcd(i,4,2)...
              abcd(i,2,2)∗abcd(i,2,3)+abcd(i,3,2)∗abcd(i,3,3)+abcd(i,4,2)∗abcd(i,4,3)...
              abcd(i,2,2)∗abcd(i,2,4)+abcd(i,3,2)∗abcd(i,3,4)+abcd(i,4,2)∗abcd(i,4,4)
                     abcd(i,2,3)∗abcd(i,2,1)+abcd(i,3,3)∗abcd(i,3,1)+abcd(i,4,3)∗abcd(i,4,1)...
              abcd(i,2,3)∗abcd(i,2,2)+abcd(i,3,3)∗abcd(i,3,2)+abcd(i,4,3)∗abcd(i,4,2)...
              abcd(i,2,3)∗abcd(i,2,3)+abcd(i,3,3)∗abcd(i,3,3)+abcd(i,4,3)∗abcd(i,4,3)...
              abcd(i,2,3)∗abcd(i,2,4)+abcd(i,3,3)∗abcd(i,3,4)+abcd(i,4,3)∗abcd(i,4,4)
                     abcd(i,2,4)∗abcd(i,2,1)+abcd(i,3,4)∗abcd(i,3,1)+abcd(i,4,4)∗abcd(i,4,1)...
              abcd(i,2,4)∗abcd(i,2,2)+abcd(i,3,4)∗abcd(i,3,2)+abcd(i,4,4)∗abcd(i,4,2)...
              abcd(i,2,4)∗abcd(i,2,3)+abcd(i,3,4)∗abcd(i,3,3)+abcd(i,4,4)∗abcd(i,4,3)...
              abcd(i,2,4)∗abcd(i,2,4)+abcd(i,3,4)∗abcd(i,3,4)+abcd(i,4,4)∗abcd(i,4,4)];% 1/m^2

    Smallb = −w ∗sigma(i) ∗ [S1(i)∗Ann(i,1)+S2(i)∗Ann(i,2)+S3(i)∗Ann(i,3);
                                             S1(i)∗Ann(i,1)+S2(i)∗Ann(i,2)+S4(i)∗Ann(i,4);
                                                     S1(i)∗Ann(i,1)+S3(i)∗Ann(i,3)+S4(i)∗Ann(i,4);
                                                     S2(i)∗Ann(i,2)+S3(i)∗Ann(i,3)+S4(i)∗Ann(i,4)]/6;% Siemens∗mV
    % Calculate Total S Matrix and b vector
    S(elmn(i,:),elmn(i,:)) = S(elmn(i,:),elmn(i,:)) + SmallS;
    b(elmn(i,:)) = b(elmn(i,:)) + Smallb;
end
disp('S Matrix and b Vector Calculated!');
C = S;
```

```
nz = ceil(nodes/2);
b(nz) = 0;
C(nz,:) = zeros(1,length(S));
C(nz,nz) = 1;
phiCalculated = qmr(C,b,1e−6,200);% mV
% Calculate Current values from calculated Phi values
Jelmn = zeros(nelmts,3);
gradPhi = zeros(nelmts,3);
for i = 1:nelmts
   phi = [phiCalculated(elmn(i,1)) phiCalculated(elmn(i,2))...
          phiCalculated(elmn(i,3)) phiCalculated(elmn(i,4))];% mV
   % Current calculated in a tetrahedron
   gradPhi(i,:) = [phi(1)∗abcd(i,2,1)+phi(2)∗abcd(i,2,2)+phi(3)∗abcd(i,2,3)+phi(4)∗abcd(i,2,4)...
              phi(1)∗abcd(i,3,1)+phi(2)∗abcd(i,3,2)+phi(3)∗abcd(i,3,3)+phi(4)∗abcd(i,3,4)...
              phi(1)∗abcd(i,4,1)+phi(2)∗abcd(i,4,2)+phi(3)∗abcd(i,4,3)+phi(4)∗abcd(i,4,4)];% mV/m
   Jelmn(i,:) = −sigma(i)∗(gradPhi(i,:)+w∗Ap(i,:));% mA/mˆ2
end

for d = 1:6:nelmts−5
   J(floor(d/6)+1,:) = sum(Jelmn(d+(0:5),:),1)/6;
end
disp('Currents Calculated!');
```

Listing F.3: SolverPlotter

```
% Define the points t ocalculate B field
x1 = −(xmax/2−1):1:(xmax/2−1);% cm
y1 = −(ymax/2−1):1:(ymax/2−1);% cm
z1 = −(zmax/2−1):1:(zmax/2−1);% cm
x2 = reshape(repmat(x1,1,(ymax−1)∗(zmax−1)),(xmax−1)∗(ymax−1)∗(zmax−1),1)/100;% m
y2 = reshape(repmat(y1,(xmax−1),(zmax−1)),(xmax−1)∗(ymax−1)∗(zmax−1),1)/100;% m
z2 = reshape(repmat(z1,(xmax−1)∗(ymax−1),1),(xmax−1)∗(ymax−1)∗(zmax−1),1)/100;% m
r = [x2 y2 z2];% m

% Calculate B field
[Bx By Bz] = findNodeBfield2(Jelmn(:,1),Jelmn(:,2),Jelmn(:,3),x2,y2,z2,midpoints(:,1),midpoints(:,2),midpoints(:,3));
B = [Bx,By,Bz]/(6∗1e6);

% Arrange B field solution to show up
N = 360;
phi11 = linspace(0,2∗pi,N+1);
phi1 = phi11(1:N);
deltaphi = phi1(2);% radian
a = radius;% m
I0 = 1;% A
mu0 = 4∗pi∗1e−7;

Bp = zeros(length(x1)∗length(y1)∗length(z1),3);
for k = 1:length(z1)
    for i = 1:length(x1)
        for j = 1:length(y1)
            u = i + ((j−1) + (k−1)∗length(y1))∗length(x1);
            R = sqrt((x2(u) − a∗cos(phi1) − xp).ˆ2+(y2(u) − a∗sin(phi1) − yp).ˆ2+(z2(u) − zp).ˆ2);% m
            Bxn = cos(phi1).∗(z2(u) − zp)./R.ˆ3;%1/mˆ2
            Byn = sin(phi1).∗(z2(u) − zp)./R.ˆ3;%1/mˆ2
            Bzn = −((x2(u) − xp).∗cos(phi1) + (y2(u) − yp).∗sin(phi1) − a)./R.ˆ3;%1/mˆ2
            Bp(u,:) = mu0∗I0∗a/(2∗N) ∗ [sum(Bxn) sum(Byn) sum(Bzn)];% mWb/mˆ2 = mTesla
        end
    end
end
```

Listing F.4: findnodeBfield2

```
#include "mex.h"
#include <math.h>

void findNodeBfield (double *Bz,
                     double *Jx, double *Jy, double *Jz,
                     double *x, double *y, double *z,
                     double *Tcentx, double *Tcenty, double *Tcentz,
                     int NodeNum, int TelmnNum, double suspend)
{
int i,j,k;
double dist,Xdiff,Ydiff,Zdiff,PercOne;

PercOne = (double) (NodeNum-1)/100;
k = 1;
for (i=0; i<NodeNum; i++) {

    *(Bz+i) = 0;

    for (j=0; j<TelmnNum; j++) {

        Xdiff = *(x+i) - *(Tcentx+j);
        Ydiff = *(y+i) - *(Tcenty+j);
        Zdiff = *(z+i) - *(Tcentz+j);

        dist = sqrt(Xdiff * Xdiff + Ydiff * Ydiff + Zdiff * Zdiff);
        dist = dist * dist * dist;
        //*(distmat+TelmnNum*i+j) = dist;

        *(Bz+i) = *(Bz+i) + ( Ydiff * *(Jx+j) - Xdiff * *(Jy+j)) / dist;

    }

    *(Bz+i) = *(Bz+i) / 1.0e7;

    if (i>=PercOne*k & suspend == 0)
    {
     printf("%d%% completed.\n",k);
     mexEvalString("drawnow;"); // to dump string.
     k = k++;
    }
}
}

void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[] )
{
double *Bz,*Jx,*Jy,*Jz,*x,*y,*z,*Tcentx,*Tcenty,*Tcentz,suspend;

int NodeNum,TelmnNum;

/* Check for proper number of arguments. */

if(nrhs!=10)
mexErrMsgTxt("Ten inputs required.");
if(nlhs!=1)
mexErrMsgTxt("One output required.");

/*Get input vector adresses*/
Jx = mxGetPr(prhs[0]);
Jy = mxGetPr(prhs[1]);
Jz = mxGetPr(prhs[2]);

x = mxGetPr(prhs[3]);
y = mxGetPr(prhs[4]);
z = mxGetPr(prhs[5]);

Tcentx = mxGetPr(prhs[6]);
Tcenty = mxGetPr(prhs[7]);
Tcentz = mxGetPr(prhs[8]);

suspend = mxGetScalar(prhs[9]);

/* Find number of tetrahedron elmts. */
TelmnNum = mxGetM(prhs[0]);

/* Find number of nodes. */
NodeNum = mxGetM(prhs[3]);

/* Set the output pointer to the output matrix. */
plhs[0] = mxCreateDoubleMatrix(NodeNum,1,mxREAL);
//plhs[1] = mxCreateDoubleMatrix(TelmnNum,NodeNum,mxREAL);

/* Create a C pointer to a copy of the output matrix. */
Bz = mxGetPr(plhs[0]);
//distmat = mxGetPr(plhs[1]);

/* Call the C subroutine. */
findNodeBfield (Bz,Jx,Jy,Jz,x,y,z,Tcentx,Tcenty,Tcentz,NodeNum,TelmnNum,suspend);
}
```

Listing F.5: MagVecPot3D

```
function [Ax,Ay,Az] = magvecpot(i,xp,yp,zp)

% Number of Nodes
global nodes
% Number of Triangles (Elements)
global nelmts
% X coordinates of nodes
global x
% Y coordinates of nodes
global y
% Z coordinates of nodes
global z
% Triangle element nodes
global elmn
global sigma
global radius

N = 360;
% Permeability of Media
mu0 = 4*pi*1e-7; % H/m
% Radius of coil
a = radius; % m
I0 = 1; % mA
% Calculate constant parts of Magnetic Vector Potential
% ----------------------------------------------------
phi = linspace(0,2*pi-(2*pi/N),N); % radian
x1 = (x(elmn(i,1)) + x(elmn(i,2)) + x(elmn(i,3)) + x(elmn(i,4)))/4;% m
y1 = (y(elmn(i,1)) + y(elmn(i,2)) + y(elmn(i,3)) + y(elmn(i,4)))/4;% m
z1 = (z(elmn(i,1)) + z(elmn(i,2)) + z(elmn(i,3)) + z(elmn(i,4)))/4;% m
R = sqrt((x1 - xp - a*cos(phi)).^2+(y1 - yp - a*sin(phi)).^2+(z1 - zp).^2);% m
Axn = -mu0*I0*a*sin(phi)./(2*R*N);% mWb/m
Ayn = mu0*I0*a*cos(phi)./(2*R*N);% mWb/m

Ax = [sum(Axn);sum(Axn);sum(Axn);sum(Axn)];
Ay = [sum(Ayn);sum(Ayn);sum(Ayn);sum(Ayn)];
Az = [0;0;0;0];
```

Listing F.6: SolverBody2

```
diary('diary.txt');
% close all;clear all;
% warning off;
% Load Mesh
Time1 = clock;
Nodenum = 4913;
load(['mesh3dtet' num2str(Nodenum) '.mat']);
load(['MagVec' num2str(Nodenum) '.mat']);
load(['Imports' num2str(Nodenum) '.mat']);
load(['Bpz' num2str(Nodenum) '.mat']);
load(['Bsz' num2str(Nodenum) '.mat']);
load(['OrgSigma' num2str(Nodenum) '.mat']);
disp('Parameters Loaded!');

% SNR = 90;
% Tc = 100e-3;
% Noise = randn(length(B1),1)/(2*42.6e6*Tc*SNR);
% Define globals
% Number of Nodes
global nodes
% Number of Triangles (Elements)
global nelmts
% X coordinates of nodes
global x
% Y coordinates of nodes
global y
% Y coordinates of nodes
global z
% Triangle element nodes
global elmn
global sigma

% Define default values for variables
w = 2 * pi * 1000;
sigma = 0.2*ones(nelmts,1);
%old_sigma = zeros(nelmts,1);
new_sigma = 0.2*ones(nelmts,1);
Org_Sigma = Org_Sigma;
Time2 = clock;
Q = 1;
RecSigma = zeros(nelmts,Q);
RecError = zeros(nelmts,Q);

%-----------------------------------------------
for iter = 1:Q
   Timer1 = clock;
   % Calculate phi values for a given sigma
   Ant = Ant1;
   Ann = Ann1;
   Ap = Ap1;
   SolverForward;
   disp('First Forward Solver');
   J1 = J;

   Timer2 = clock;
   Ant = Ant2;
   Ann = Ann2;
   Ap = Ap2;
   SolverForward;
   disp('Second Forward Solver');
   J2 = J;

   Timer3 = clock;
   % Calculate phi values for a given sigma
```

```
    Ant = Ant3;
    Ann = Ann3;
    Ap = Ap3;
    SolverForward;
    disp('Third Forward Solver');
    J3 = J;

    Timer4 = clock;
    % Calculate phi values for a given sigma
    Ant = Ant4;
    Ann = Ann4;
    Ap = Ap4;
    SolverForward;
    disp('Fourth Forward Solver');
    J4 = J;

    Timer5 = clock;
    % Calculate new sigma values for a given phi and B field
    SolverInverse;
    disp('Inverse Solver');

    % Replace new sigma values with old ones
%   old_sigma = sigma;
    sigma = new_sigma;
    RecSigma(:,iter) = new_sigma;
    %RecRho = Rsig
    % Calculate Error between new sigma and old one
    %ErrorSig = ((OrgR - reshape([Rsig;Rsig;Rsig;Rsig;Rsig;Rsig],1,nelmts))./OrgR);
    ErrorSig = ((Org_Sigma - sigma)./Org_Sigma);
    ErrorTotal(iter) = 100*(norm(Org_Sigma - sigma)./norm(Org_Sigma));
    %ErrorSig = 0;
    RecError(:,iter) = ErrorSig;
    Timer6 = clock;

    disp(['Iteration Step (' num2str(iter) ')' 'Error % ' num2str(ErrorTotal)]);
    disp(['Max Rsig ' num2str(max(Rsig)) ' Min Rsig ' num2str(min(Rsig))]);
    disp(['Max sigma ' num2str(max(sigma)) ' Min sigma ' num2str(min(sigma))]);

    K = floor(etime(Timer2,Timer1)/3600);
    L = floor((etime(Timer2,Timer1)-3600*K)/60);
    M = etime(Timer2,Timer1) - 3600*K - 60*L;
    disp(['Solver Forward 1 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
    K = floor(etime(Timer3,Timer2)/3600);
    L = floor((etime(Timer3,Timer2)-3600*K)/60);
    M = etime(Timer3,Timer2) - 3600*K - 60*L;
    disp(['Solver Forward 2 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
    K = floor(etime(Timer4,Timer3)/3600);
    L = floor((etime(Timer4,Timer3)-3600*K)/60);
    M = etime(Timer4,Timer3) - 3600*K - 60*L;
    disp(['Solver Forward 3 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
    K = floor(etime(Timer5,Timer4)/3600);
    L = floor((etime(Timer5,Timer4)-3600*K)/60);
    M = etime(Timer5,Timer4) - 3600*K - 60*L;
    disp(['Solver Forward 4 : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
    K = floor(etime(Timer6,Timer5)/3600);
    L = floor((etime(Timer6,Timer5)-3600*K)/60);
    M = etime(Timer6,Timer5) - 3600*K - 60*L;
    disp(['Inverse Solver   : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
    K = floor(etime(Timer6,Timer1)/3600);
    L = floor((etime(Timer6,Timer1)-3600*K)/60);
    M = etime(Timer6,Timer1) - 3600*K - 60*L;
    disp(['Total Iter. Time : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);

end
%-------------------------------------------------
Time3 = clock;
save(['RecValue' num2str(Nodenum) '_1.mat'],'RecSigma','RecError');
```

```
%SolverPlotter;
K = floor(etime(Time2,Time1)/3600);
L = floor((etime(Time2,Time1)−3600∗K)/60);
M = etime(Time2,Time1) − 3600∗K − 60∗L;
disp(['Calculate VMPs : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time3,Time2)/3600);
L = floor((etime(Time3,Time2)−3600∗K)/60);
M = etime(Time3,Time2) − 3600∗K − 60∗L;
disp(['Iterative Method : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);
K = floor(etime(Time3,Time1)/3600);
L = floor((etime(Time3,Time1)−3600∗K)/60);
M = etime(Time3,Time1) − 3600∗K − 60∗L;
disp(['Total : ',num2str(K),' hour(s) ',num2str(L),' minute(s) ',num2str(M),' second(s)']);

diary;
```

Listing F.7: SolverInverse

```
global nodes
global nelmts
global x
global y
global z
global elmn

% OrgR = −log(Org_Sigma);

% Distance Between Two Nodes
deltax = 0.01;
deltay = 0.01;
% Length of Maximum Number of Nodes
Lx = (xmax−1);
Ly = (ymax−1);
Lz = (zmax−1);
% Total Node Number in a Slice
SqSize = Lx∗Ly;
% Sigma Values in the middle of a Square
sigmamid = sum(reshape(sigma,nelmts/6,6),2)/6;
Rsig = zeros(size(sigmamid));
% Calculate del2B/mu0
mu0 = 4∗pi∗1e−7;

gama = 26.75e7;
SNR = 30;
tcurr = 0.1;
Noise1 = 0;Noise2 = 0;Noise3 = 0;Noise4 = 0;
for i = 1:100
    Noise1 = Noise1 + randn(size(B1(:,3)))/(2∗gama∗tcurr∗SNR)/100;
    Noise2 = Noise2 + randn(size(B1(:,3)))/(2∗gama∗tcurr∗SNR)/100;
    Noise3 = Noise3 + randn(size(B1(:,3)))/(2∗gama∗tcurr∗SNR)/100;
    Noise4 = Noise4 + randn(size(B1(:,3)))/(2∗gama∗tcurr∗SNR)/100;
end;
LapNoise1 = 6∗reshape(del2(reshape(Noise1/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);
LapNoise2 = 6∗reshape(del2(reshape(Noise2/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);
LapNoise3 = 6∗reshape(del2(reshape(Noise3/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);
LapNoise4 = 6∗reshape(del2(reshape(Noise4/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);

LapHz1 = 6∗reshape(del2(reshape((B1(:,3))/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);
LapHz2 = 6∗reshape(del2(reshape((B2(:,3))/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);
LapHz3 = 6∗reshape(del2(reshape((B3(:,3))/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);
LapHz4 = 6∗reshape(del2(reshape((B4(:,3))/mu0,Lx,Ly,Lz),0.01,0.01,0.01),SqSize∗Lz,1);

delB1 = −w ∗ Bp1(:,3);
delB2 = −w ∗ Bp2(:,3);
```

```
delB3 = −w ∗ Bp3(:,3);
delB4 = −w ∗ Bp4(:,3);

%−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
for Slice = 8:8%2:Lz−1
   % Define Offset value for ith Slice
   ofs = (Slice−1)∗SqSize;
   % Define Profiles
   for Profile = 1:4
      if Profile == 1
         J = J1;
         LapHz = LapHz1+LapNoise1;
      elseif Profile == 2
         J = J2;
         LapHz = LapHz2+LapNoise2;
      elseif Profile == 3
         J = J3;
         LapHz = LapHz3+LapNoise3;
      elseif Profile == 4
         J = J4;
         LapHz = LapHz4+LapNoise4;
      end
      % Initialize Sparse Matrix C
      clear C;
      C=sparse(SqSize,SqSize);
      % Inner Side of The Matrix
      for i=2:Lx−1
         for j=2:Ly−1
            k = i+(j−1)∗Lx;
            n = k+ofs;
            C(k,k−1) = C(k,k−1) − J(n,2)/(2∗deltax);
            C(k,k+1) = C(k,k+1) + J(n,2)/(2∗deltax);
            C(k,k+Lx) = C(k,k+Lx) − J(n,1)/(2∗deltay);
            C(k,k−Lx) = C(k,k−Lx) + J(n,1)/(2∗deltay);
         end
      end

      % First Row of The Matrix
      i=1;
      for j=2:Ly−1
         k = i+(j−1)∗Lx;
         n = k+ofs;
         C(k,k+1) = C(k,k+1) + J(n,2)/deltax;
         C(k,k) = C(k,k) − J(n,2)/deltax;
         C(k,k+Lx) = C(k,k+Lx) − J(n,1)/(2∗deltay);
         C(k,k−Lx) = C(k,k−Lx) + J(n,1)/(2∗deltay);
      end

      % Last Row of The Matrix
      i=Lx;
      for j=2:Ly−1
         k = i+(j−1)∗Lx;
         n = k+ofs;
         C(k,k) = C(k,k) + J(n,2)/deltax;
         C(k,k−1) = C(k,k−1) − J(n,2)/deltax;
         C(k,k+Lx) = C(k,k+Lx) − J(n,1)/(2∗deltay);
         C(k,k−Lx) = C(k,k−Lx) + J(n,1)/(2∗deltay);
      end

      % First Column of The Matrix
      j=1;
      for i=2:Lx−1
         k = i+(j−1)∗Lx;
         n = k+ofs;
         C(k,k−1) = C(k,k−1) − J(n,2)/(2∗deltax);
         C(k,k+1) = C(k,k+1) + J(n,2)/(2∗deltax);
         C(k,k+Lx) = C(k,k+Lx) − J(n,1)/deltay;
         C(k,k) = C(k,k) + J(n,1)/deltay;
```

```
        end

        % Last Column of The Matrix
        j=Ly;
        for i=2:Lx-1
           k = i+(j-1)*Lx;
           n = k+ofs;
           C(k,k-1) = C(k,k-1) - J(n,2)/(2*deltax);
           C(k,k+1) = C(k,k+1) + J(n,2)/(2*deltax);
           C(k,k) = C(k,k) - J(n,1)/deltay;
           C(k,k-Lx) = C(k,k-Lx) + J(n,1)/deltay;
        end

        % Upper Left Corner of The Matrix
        i=1;j=1;
        k = i+(j-1)*Lx;
        n = k+ofs;
        C(k,k) = C(k,k) - J(n,2)/deltax + J(n,1)/deltay;
        C(k,k+1) = C(k,k+1) + J(n,2)/deltax;
        C(k,k+Lx) = C(k,k+Lx) - J(n,1)/deltay;

        % Lower Left Corner of The Matrix
        i=1;j=Ly;
        k = i+(j-1)*Lx;
        n = k+ofs;
        C(k,k) = C(k,k) - J(n,2)/deltax - J(n,1)/deltay;
        C(k,k+1) = C(k,k+1) + J(n,2)/deltax;
        C(k,k-Lx) = C(k,k-Lx) + J(n,1)/deltay;

        % Lower Right Corner of The Matrix
        i=Lx;j=Ly;
        k = i+(j-1)*Lx;
        n = k+ofs;
        C(k,k) = C(k,k) + J(n,2)/deltax - J(n,1)/deltay;
        C(k,k-1) = C(k,k-1) - J(n,2)/deltax;
        C(k,k-Lx) = C(k,k-Lx) + J(n,1)/deltay;

        % Upper Right Corner of The Matrix
        i=Lx;j=1;
        k = i+(j-1)*Lx;
        n = k+ofs;
        C(k,k) = C(k,k) + J(n,2)/deltax + J(n,1)/deltay;
        C(k,k-1) = C(k,k-1) - J(n,2)/deltax;
        C(k,k+Lx) = C(k,k+Lx) - J(n,1)/deltay;

         for i=1:Lx
           for j=1:Ly
              k = i+(j-1)*Lx;
              n = k+ofs;
              C(k,k) = C(k,k) - LapHz(n);
           end
        end

        if Profile == 1
           smat1 = C;
        elseif Profile == 2
           smat2 = C;
        elseif Profile == 3
           smat3 = C;
        elseif Profile == 4
           smat4 = C;
        end
     end
     %-----------------------------------------------------------------
     % Initial Condition
     Scon = [smat1;smat2;smat3;smat4];
     dcon = [delB1(ofs+(1:SqSize));delB2(ofs+(1:SqSize));delB3(ofs+(1:SqSize));delB4(ofs+(1:SqSize))];
     % Calculate the natural logartihm of new sigma values
```

```
    %Rsig(ofs + [2:Lx−1 Lx+1:(Lx∗(Ly−1)) (Lx∗(Ly−1)+(2:Lx−1))]) = pinv(full(Scon))∗dcon;
    L = get_l(SqSize,1);
    [UU,sm,VV] = cgsvd(Scon,L);
    lambda = gcv(UU,sm,dcon);
    [X_tikh,rho,eta] = tikhonov(UU,sm,VV,dcon,lambda);

    corner = l_curve(UU,sm,dcon,'tsvd');
    close all;
    X_L = tgsvd(UU,sm,VV,dcon,corner);
%   Rsig(ofs + (1:SqSize)) = X_tikh;
    Rsig(ofs + (1:SqSize)) = inv(Scon'∗Scon)∗(Scon'∗dcon);
    figure;
    plot(1./Org_Sigma(6∗8∗SqSize+(1:6:6∗SqSize)));
    hold on;plot(X_tikh,'−−');plot(X_L,':');plot(Rsig(ofs + (1:SqSize)),'−.');
    title('Results in midslice');
    legend('Original Resistivity','Tikhonov Regularization','TGSVD Regularization','Direct Inverse Solution');
    figure;
    subplot(221);imagesc(reshape(1./Org_Sigma(6∗8∗SqSize+(1:6:6∗SqSize)),Lx,Ly));colorbar;
    title('Original Resistivity');
    subplot(222);imagesc(reshape(X_tikh,Lx,Ly));colorbar;title('Tikhonov Regularization');
    subplot(223);imagesc(reshape(X_L,Lx,Ly));colorbar;title('TGSVD Regularization');
    subplot(224);imagesc(reshape(Rsig(ofs + (1:SqSize)),Lx,Ly));colorbar;title('Direct Inverse Solution');
    %colormap gray;
end
Rsig(1:SqSize) = Rsig(SqSize + (1:SqSize));
Rsig((Lz−1)∗SqSize + (1:SqSize)) = Rsig((Lz−2)∗SqSize + (1:SqSize));
cnt1 = 0;
for cnt = 1:6:nelmts
    cnt1 = cnt1+1;
    new_sigma(cnt:cnt+5) = 1/Rsig(cnt1);
end
```