Improving Forward Matrix Generation and Utilization for Time Domain Diffuse Optical Tomography

by

Damon Hyde

B.S., Worcester Polytechnic Institute, 2002

A thesis submitted to the

Department of Electrical and Computer Engineering of

Northeastern University

in partial fulfillment of the requirements

for the degree of

Master of Science

2004

This thesis entitled:
Improving Forward Matrix Generation and Utilization for Time Domain Diffuse Optical Tomography
written by Damon Hyde
has been approved for the Department of Electrical and Computer Engineering

Eric Miller

Prof. Dana Brooks

Dr. Vasilis Ntziachristos

Dr. Frederic Lesage

Dr. David Boas

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Hyde, Damon (M.S.E.E.)

Improving Forward Matrix Generation and Utilization for Time Domain Diffuse Optical Tomography

Thesis directed by Prof. Eric Miller

Methods of imaging of the body through the use of diffusive near-infrared light have gained significant attention in recent years. Methods using both time and frequency domain data collection methods have been developed, and show significant clinical promise.

One issue that arises through the use of time-dependant intensity measurements is the large number of data points utilized. The matrices associated with these large numbers of data points are quite large, possibly requiring hundreds or even thousands of megabytes of storage space. Explicitly calculating each value in these matrices requires days of computing time on even well equipped machines.

There are two primary contributions of this thesis to the area of diffuse optical tomography. The first details a method through which redundancies present in the source-detector geometry can intelligently be exploited to eliminate excess computations during the generation of the forward matrix. This minimizes both the time required to compute the forward matrix, as well as the space required to store it. This result is presented in a generalized fashion which applies to all source-detector configurations in a slab geometry, with potential applications in other areas.

The second contribution of this work is the use of a linear interpolation method to reduce both the initial forward matrix computation time, as well as the time required for each matrix-vector multiplication involved in the solution.

Results are then presented for a variety of interpolation levels, with both matched and mismatched simulated models. Comparisons are made with both the absolute truth and the solutions derived from a fully computed matrix.

Dedication

To My Parents

Acknowledgements

I would like to thank my advisor, Prof. Eric Miller, as well as Advanced Research Technologies, for the financial backing to pursue this project. Both Prof Miller and Prof Dana Brooks deserve many thanks for the numerous amounts of advice and guidance they have provided, as well as their helpful and insightful comments on the drafts of this thesis.

I would also like to thank my friends and family for their support throughout the last two years. Finally, I'd like to thank Clara for being understanding that this isn't the end, but only a step as I head forward towards a doctorate.

Contents

Tables

Figures

Figure

Chapter 1

Introduction

One area of current research in the field of biomedical optics involves the use of diffusive light for the noninvasive imaging of tissue optical properties. This method, often referred to as Diffuse Optical Tomography (DOT), uses near-infrared light with wavelengths in the range of 700-1100nm to estimate the spatial distribution of absorption and scattering parameters within the tissue [1]. These wavelengths are of particular interest because of the chromophore whose absorption and scattering coefficients dominate in most tissues of interest: hemoglobin. The primary oxygen carrier in blood, hemoglobin has two forms: deoxyhemoglobin, and its oxygen-saturated counterpart: oxyhemoglobin. These two variants have distinct oxygenation and wavelength dependent optical parameters [2] that, given measurements at two or more wavelengths, can provide an image of the blood concentration and oxygenation levels within the tissue.

These physiological sources of absorption and scattering information are potentially useful in several different clinical situations. Tumors tend to be highly vascularized, leading to an increase in blood density [3]. Areas of the brain see perturbed levels of blood oxygenation when active. Patients with aneurisms have bubbles in the walls of blood vessels, leading to high concentrations of blood. These perturbations, caused by multiple different physiological sources, can all potentially be imaged using a DOT system.

Several methods of approaching the DOT problem have been proposed. Some researchers use a continuous amplitude source approach, taking measurements at various spatial locations on the surface [4]. Others use a modulated source and approach the problem in the frequency domain [5][6][7]. Still others collect the time varying amplitude response to a pulsed laser [1].

For nearly all configurations, however, the DOT problem is an ill posed and difficult problem. For frequency domain techniques, the problem tends to be highly underdetermined due to restrictions on both source and sensor geometry. To noninvasively image living tissue, sources and detectors can only be placed on accessible surfaces, thereby restricting the number of usable locations. Despite being subject to these same restrictions, the time domain problem using a Born approximation is usually over-determined. While the number of source-detector pairs may be comparable to a frequency domain imager, the number of data samples collected (potentially thousands per source-detector pair), make for an overall over-determined system. This creates problems of its own however. Attempting to explicitly generate and manipulate the matrices necessary for time domain imaging severely taxes even well equipped machines. As such, intelligent methods of generating these matrices are necessary for efficient and timely use of these time domain models.

This efficient generation, storage, and use of the forward matrices is the primary goal of this thesis. This goal is achieved through two primary paths: exploitation of underlying symmetries in the source-detector configuration, and interpolation. The primary contribution of this thesis is an algorithm associated with these two techniques which can be applied to a wide variety of problems.

Based on an examination of the underlying symmetries present in many inverse scattering problems, a comprehensive method of analyzing these problems is presented. This method allows for the elimination of a large number of excess computations, resulting in a much shorter amount of time being required for the computation and application of the forward matrix. In the process of eliminating these excess computations, what begins as a very large forward matrix is decomposed into a single compact matrix, and a number of large, sparse, easily computed matrices. This allows for a large savings in storage versus computing the entire matrix explicitly.

Once the computations being performed are trimmed to a minimal set, interpolation is utilized to realize additional gains in both computational complexity and storage size. By using a linear interpolation method which can easily be represented in matrix form, the compact matrix previously developed is further decomposed into an even smaller compact matrix, and a single sparse one. By combining this new sparse matrix with the previously developed sparse matrices, the speed with which matrix vector products can be performed is increased. This increase is used to improve the speed with which a solution can be obtained from an iterative inverse solution.

For solutions to the inverse problem, multiple established methods are utilized. A traditional Tikhonov regularized least squares solution is used as the baseline solution. Additional solutions are generated using the LSQR algorithm. This latter algorithm is utilized due to its ability to exploit the sparse nature of the matrices involved.

1.1 Other Work

Important initial work in determining the time domain solutions to the diffusion equation was done by Patterson and Chance [8]. They developed equations for the time resolved transmittance of diffusive light through tissue. These equations form the basis of the system which defines time domain DOT. Of specific importance to this thesis, this article was utilized in the development of the time domain portion of the Photon Migration Imaging (PMI) toolbox, which was utilized for matrix generation in this thesis.

A significant amount of work on inversion of time domain DOT has been done by Simon Arridge and his group at University College in London. Working with a time domain system for brain imaging, they have developed TOAST (Time Resolved Optical Absorption and Scattering Tomography), a toolset designed for doing time domain DOT work. They described the theoretical basis for perturbative approaches [2] in both the time and frequency domains for a variety of different sensor geometries. Analytical forms of the associated Jacobians were then developed [9] for some of these geometries. These Jacobians are given in terms of the spatially varying absorption and scattering parameters, with the intention of being used as the kernel of the inverse problem. Using the methods outlined in their previous papers, the group developed their system for time domain imaging

1.2 Outline of Thesis

In this thesis, we concentrate on the problem of rapidly generating Born matrices for a time domain DOT problem in a transmission geometry. Results using the rapidly generated matrices are shown for several different cases and compared to results using matrices generated in a conventional manner. Comparisons are based upon the accuracy of the reconstruction, as well as the amount of computation required required to obtain that reconstruction.

Chapter 2 discusses and develops the forward model used for this research, as well as the inversion methods utilized. Chapter 3 covers the methods of rapid matrix generation we have developed. Results for a variety of test scenarios are presented in Chapter 4. These scenarios include simulated data using both matched and mismatched models, as well as clinically collected data. Conclusions and suggestions for further research are presented in Chapter 5.

Chapter 2

Linear DOT Models and Inversion

The goal of any inverse problem is, given a set of data, to determine the value of some parameter or parameters of interest which had an effect upon that data. In a DOT system, the data collected are some measurement of light transmitted through the tissue being imaged. The parameters to be recovered are the spatially varying absorption and scattering parameters. While the relationship between parameters and data can be determined analytically in imaging modalities such as CAT or MRI systems, this can not easily be done for DOT. Instead, some type of approximations or assumptions are made about the parameters to be determined, and about the model by which these parameters affect the data. These presumptions allow the problem to fit into a form which is mathematically easier to evaluate. With this new, simpler model, various techniques can then be applied to determine numerical solutions to the problem.

2.1 Forward Model

The basis of any inverse system is its forward model. This model needs to accurately represent the way in which the desired parameters interact with the inputs to affect the data collected at the detectors. This model also needs to be either analytically invertible, or suitable for some type of numerical inversion.

For this work, we have chosen to use a linearized model based around the first order Born approximation. In the following section, we start with the diffusion approximation. From there, we build up a linearized forward model which can then be used with a variety of numerical inversion techniques.

2.1.1 Radiative Transport and the Diffusion Approximation

Due to the diffusive behavior of infrared radiation in tissue, deterministic modelling methods based on Maxwell's equations have such huge computational requirements associated with them that they are impractical for use. Because of this, methods based on stochastic relations have been developed to deal with these types of problems. Rather than treat each photon individually, they are viewed in more of a bulk fashion, given that the number of photons involved in a data collection experiment will be large. This resulting "radiance" or "photon density" can be thought of as being related to the number of photons in a given volume. The equation forming the basis of these methods is the Radiative Transport Equation (RTE), originally developed by Chandrasekhar [10]:

$$\frac{1}{v}\frac{\partial L(\mathbf{r},\hat{\Omega},t)}{\partial t} + \nabla \cdot L(\mathbf{r},\hat{\Omega},t) + \mu_t L(\mathbf{r},\hat{\Omega},t) = \mu_s \int L(\mathbf{r},\hat{\Omega}',t) f(\hat{\Omega},\hat{\Omega}')\, d\hat{\Omega}' + S(\mathbf{r},\hat{\Omega},t) \tag{2.1}$$

In (2.1), $L(\mathbf{r},\Omega,t)$ is the intensity of light at a position $\mathbf{r}$, in a direction Ω, at time t. The probability of a photon scattering from direction $\hat{\Omega}$ to direction $\hat{\Omega}'$ is given by $f(\hat{\Omega},\hat{\Omega}')$. The speed of light is c and the source is denoted as $S(\mathbf{r},\hat{\Omega},t)$. Additionally, μ_a and μ_s are the spatially varying absorption and scattering parameters, respectively, with μ_t equal to $\mu_s + \mu_a$. The terms on the left side represent radiance lost from a differential volume in a differential solid angle, while the terms on the right represent the radiance gained. Thus this equation is an expression of conservation of photons.

The RTE unfortunately also requires a great deal of computation to evaluate, and is thus unsuitable for use in general inverse problems. By making several assumptions about the structure of the source and the diffusive material, however, a further approximation can be derived. The assumptions that $f(\hat{\Omega},\hat{\Omega}')$ is dependent only on the angle between the directions, and also that the source is isotropic, lead to the diffusion approximation [5][2], which states that the photon density, $\Phi(r,t)$, satisfies the differential equation

$$(\gamma^2\nabla^2 - \mu_a - \frac{1}{v}\frac{\partial}{\partial t})\Phi(\mathbf{r},t) = -q(\mathbf{r},t) \tag{2.2}$$
$$\gamma^2 = \frac{1}{3[\mu_a + (1-\overline{p})\mu_s]}$$

where v is the speed of light in the tissue, μ_a and μ_s are the absorption and scattering coefficients, respectively, $\overline{p}$ is the mean cosine of the scattering angle, and $q(\mathbf{r}, t)$ is the source term.

Because $\Phi(\mathbf{r}, t)$ represents the density of photons at any one point in space, it is not a directly measurable quantity. In order to transform the density into a measurable variable it is necessary to take the gradient along the direction of a vector $\hat{\nu}$ which is normal to the surface at the point where the data are being collected. This results in the equation for the photon flux:

$$\Gamma(\mathbf{m}, t) = -\gamma^2 \hat{\nu} \cdot \nabla \Phi(\mathbf{r}, t) \tag{2.3}$$

This flux is the quantity measured at each detector.

2.1.2 Green's Functions

A closed form solution to (2.2) is not available in the general case. However, for the case of a homogeneous medium (with respect to the absorption and scattering parameters), a solution to the diffusion equation can be found in the form of a Green's function. In the time domain, the Green's function is a solution to the diffusion equation when the source function $q(\mathbf{r}, t)$ is an impulse at $t = t'$. The solution for any arbitrary input waveform can then be evaluated by convolution with the Green's function.

According to a proof provided by Arridge et al [2] the Green's function solutions to the diffusion equation are related to the Green's function solutions to the lossless heat conduction equation by a factor of $e^{-\mu_a c t}$. The Green's function solutions for the heat conduction equation for many geometries were solved for by Carslaw and Jaeger [11]. These solutions were then used by Arridge as a basis from which to construct the Green's functions for the diffusion equation. The extensive results provided by Carslaw and Jaeger could also potentially be used to solve for additional geometries, including those not solved for by Arridge [2].

One important consideration when dealing with realistic geometries is the boundary condition used in the solution. When (2.2) is solved for the case where the source is an impulse at $t = t'$, the resulting solution is:

$$\phi(\mathbf{r}, t) = v(4\pi\gamma^2 vt)^{-3/2} \exp(-\frac{\mathbf{r}^2}{4\gamma^2 vt} - \mu_a vt) \tag{2.4}$$
$$\gamma^2 = \frac{1}{3[\mu_a + (1-g)\mu_s]}$$

where g is the mean cosine of the scattering angle, and $\mathbf{r}$ is a the location of any point in space, presuming the source is located at the origin. This solution is only valid in an infinite medium, or in one where the boundaries are far enough from both the source and detector that the medium can be treated as infinite.

Given the restriction that sources and detectors must be noninvasive, and therefore located on the surface of the medium, an infinite geometry is unrealizable in any practical system. Because of this, solutions for other geometries must be obtained. In order to do this, a model must be developed which describes the behavior of the diffusing light at the boundary of the medium. This model must then be taken into account when solving (2.2).

Over the years, several different solutions have been proposed [12]. These different solutions vary in both their accuracy and their mathematical complexity. For this work, a zero boundary condition was chosen. Using an infinite slab geometry, and presuming that the slab spans the Z-axis from $z = 0$ to $z = z_d$, this amounts to placing the condition:

$$\phi(x, y, z, t) = 0 \quad z = \{0, z_d\} \tag{2.5}$$

on (2.2). While this presumption violates the diffusion approximation and does not accurately represent the physics, it is mathematically simple to implement. A more accurate solution can be obtained through the use of an extrapolated boundary condition, where $\phi(\mathbf{r}, t)$ is set equal to zero at some point near the boundary. Observations have been made that suggest that when the source-detector separation is large in comparison to the extrapolation length, the additional complexity added by the extrapolated boundary condition is unnecessary [8]. In such cases, it should suffice to stay with a simple zero boundary condition and gain the benefit of mathematical simplicity.

The solution to (2.2) for an infinite slab geometry with a zero boundary condition utilizes (2.4) and an infinite series of dipoles to satisfy the boundary condition. These dipoles are illustrated in Figure 2.1. By alternating positive and negative sources, and spacing these sets of sources every $2z_d$ along the Z-axis, the boundary condition is satisfied. This results in the Green's Function solution being in the form of an infinite summation. Ultimately, after some rearranging, the Green's function solution to (2.2) for an infinite slab geometry is given by Arridge et al [2] to be:

$$\begin{aligned} g^{\Phi}_{slab}(\mathbf{r}, \mathbf{r}', t, t') = & \frac{\exp\{-[\mu_a c(t-t') + \frac{\xi^2}{4\gamma^2(t-t')}]\}}{[4\pi\gamma^2(t-t')]^{3/2}} \\ & \times \sum_{n=-\infty}^{\infty} [\exp(-(z - 2z_d n - z_0)^2/4\gamma^2(t-t')) \\ & \qquad - \exp(-(z - 2z_d n + z_0)^2/4\gamma^2(t-t'))] \\ & \xi = \sqrt{x^2+y^2} \\ & z_o = [(1-\bar{\rho})\mu_s]^{-1}] \end{aligned} \tag{2.6}$$

Here, **r'** and **r** are the source and detector locations, respectively. The variable z_o is the source depth, which is set inside the upper boundary of the slab by one mean scattering length. This scattering length is the average distance a photon can be expected to travel along a path before being scattered in another direction. It enters into the solution because light from the source laser is presumed to travel, on average, this distance into the tissue before a scattering event occurs. Finally, z_d is the thickness of the slab being evaluated. If the vector $\mathbf{r} - \mathbf{r}'$ is represented in Cartesian coordinates as (x, y, z) then $\xi = \sqrt{x^2+y^2}$, is the distance between source and detector in the X, Y plane, and z is the depth at which the Green's function is being evaluated.

Arridge then takes the gradient of (2.6) and evaluates it at $z = z_d$ to give the Green's function for the flux at a point r:

$$\begin{aligned} g^{(\Gamma)}_{slab}(\xi, z_0, t, t') = & \frac{\exp\{-[\mu_a c(t-t') + \frac{\xi^2}{4\gamma^2(t-t')}]\}}{(4\pi\gamma^2)^{3/2}(t-t')^{5/2}} * \\ & \sum_{n=0}^{\infty} [z_{+n} \exp(\frac{-z_{+n}^2}{4\gamma^2(t-t')}) - z_{-n} \exp(\frac{-z_{-n}^2}{4\gamma^2(t-t')})] \\ & z_{+n} = (2n+1)z_d + z_0 \\ & z_{-n} = (2n+1)z_d - z_0 \end{aligned} \tag{2.7}$$

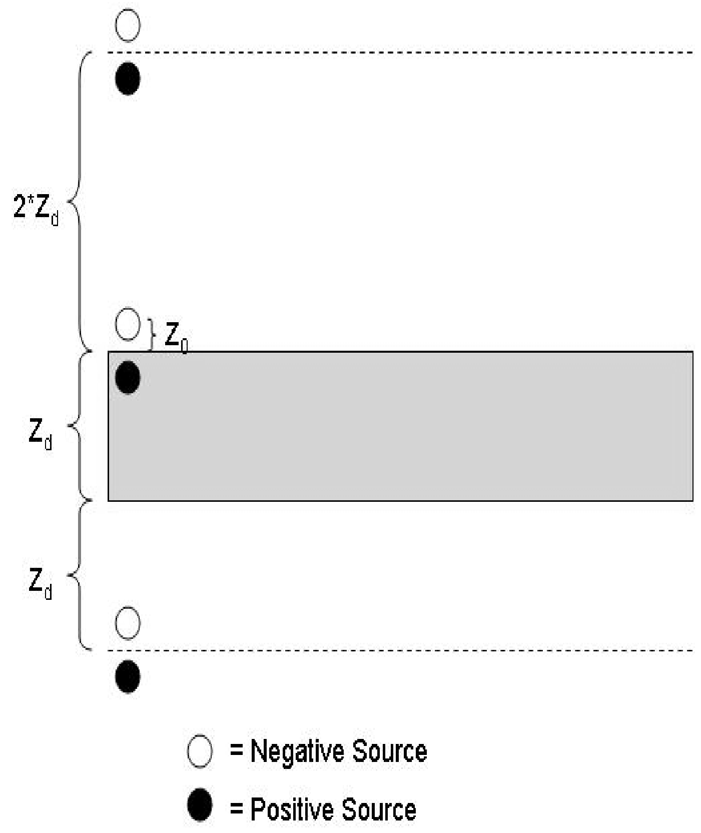

Figure 2.1: Illustration of positive and negative dipoles used to satisfy boundary condition of $\phi(x, y, z, t) = 0$ for $z = \{0, z_d\}$ to generate the Green's Function for an infinite slab geometry.

Evaluating the function at $z = z_d$ is consistent with the presumption that the sources are located on the top of the slab at $z = 0$, while the detectors are located at $z = z_d$. This represents a transmission configuration for the sources and detectors, which is the system utilized for this work. These two Green's functions, (2.6) and (2.7), enable us to scatter light first from a source to a voxel within the volume, and then from that voxel to the detector in question. This use of first order scattering, when summed across the entire volume, yields the Born Approximation.

2.1.3 Born Approximation

Utilizing the two previously stated Green's functions, a perturbation analysis of the system can be performed. This type of analysis presumes that the total field at the detector can be expressed as the sum of a background field and a series of perturbations. Arridge [9] states that the change in the flux measurement Γ due to some perturbation $\eta(\mathbf{r}')$ of the absorption parameter μ_a at the location r, and some perturbation $\kappa(\mathbf{r}')$ of the parameter γ^2 is approximately equal to:

$$\begin{aligned} \triangle\Gamma(\xi,\zeta,t) \approx -\int_{\Omega} d^3\mathbf{r}' \int_{-\infty}^{\infty} dt' [g^{(\Gamma)}_{slab}(\xi,\mathbf{r}',t')\eta(\mathbf{r}')g^{(\Phi)}(\mathbf{r}',\zeta,t-t') - \\ \kappa(\mathbf{r}')\nabla_{\mathbf{r}'} g^{(\Gamma)}_{slab}(\xi,\mathbf{r}',t') \cdot \nabla_{\mathbf{r}'} g^{(\Phi)}(\mathbf{r}',\zeta,t-t')] \end{aligned} \tag{2.8}$$

where ζ and ξ are the source and detector locations, respectively.

This equation can be shown to be equal to the integral form of the perturbation component of the first order Born approximation. The Born approximation presumes that for a primarily homogeneous medium, the total flux at a detector is equal to the flux at that detector that would be seen were the material homogeneous, plus the sum of perturbations resulting from first order scattering for each of the perturbations.

The Jacobian is then taken to be the ratio of the change in measurement to the change in optical parameter in the limit as the perturbation magnitude goes to zero.

$$J^{(\Gamma)}_{p,slab} = lim_{\Delta p \to 0} \frac{\Delta\Gamma}{\Delta p} \tag{2.9}$$

Here, the perturbation Δp can be either the absorption perturbation $\eta(r)$ or the scattering perturbation $\kappa(r)$. The Jacobian needs to be computed separately for each of the two coefficients.

If the space is then voxelized into N_{xyz} discrete voxels, the above Jacobian(s) can be evaluated for each of the voxels individually, by presuming that the rest of the medium has no perturbations. (This is consistent with the presumption that only first order scattering contributes significantly to the overall measurement perturbation). If these Jacobians are evaluated at each of the desired time sampling points, for each of the source-detector pairs, what results is an $(N_S * N_D * N_T) \times N_{xyz}$ matrix where $\zeta = \{\zeta_1, \ldots, \zeta_{N_S}\}$ is the set of all source locations, $\xi = \{\xi_1, \ldots, \xi_{N_D}\}$ is the set of all detector locations, and $t = \{t_1, \ldots, t_{N_T}\}$ is the set of all sample timepoints, and N_{xyz} is the total number of voxels in the medium.

$$\begin{bmatrix}
J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_1;r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_1;r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_1;r'_{N_{XYZ}}) \\
J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_2;r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_2;r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_2;r'_{N_{XYZ}}) \\
\vdots & \ldots & \vdots & \ldots & \vdots \\
J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_{N_t};r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_{N_t};r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_1,t_{N_t};r'_{N_{XYZ}}) \\
J^{(\Gamma)}_{p,slab}(\xi_2,\zeta_1,t_1;r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_2,\zeta_1,t_1;r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_2,\zeta_1,t_1;r'_{N_{XYZ}}) \\
\vdots & \ldots & \vdots & \ldots & \vdots \\
J^{(\Gamma)}_{p,slab}(\xi_{N_D},\zeta_1,t_{N_t};r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_{N_D},\zeta_1,t_{N_t};r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_{N_D},\zeta_1,t_{N_t};r'_{N_{XYZ}}) \\
J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_2,t_1;r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_2,t_1;r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_1,\zeta_2,t_1;r'_{N_{XYZ}}) \\
\vdots & \ldots & \vdots & \ldots & \vdots \\
J^{(\Gamma)}_{p,slab}(\xi_i,\zeta_j,t_k;r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_i,\zeta_j,t_k;r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_i,\zeta_j,t_k;r'_{N_{XYZ}}) \\
\vdots & \ldots & \vdots & \ldots & \vdots \\
J^{(\Gamma)}_{p,slab}(\xi_{N_D},\zeta_{N_S},t_{N_T};r'_1) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_{N_D},\zeta_{N_S},t_{N_T};r'_q) & \ldots & J^{(\Gamma)}_{p,slab}(\xi_{N_D},\zeta_{N_S},t_{N_T};r'_{N_{XYZ}})
\end{bmatrix} \tag{2.10}$$

These matrices, obtained by evaluating (2.10) using the Jacobians for each of absorption and scattering parameters, are denoted $\mathbf{A}_\eta$ and $\mathbf{A}_\kappa$ respectively. Using the Green's function for photon flux ((2.7)) to determine the homogeneous response of the system, and using $\mathbf{A}$ to model the perturbation component, gives a total system equation of:

$$\mathbf{g}_{tot} = \underbrace{\mathbf{A}_\eta \mathbf{f}_\eta}_{\mathbf{g}_\eta} + \underbrace{\mathbf{A}_\kappa \mathbf{f}_\kappa}_{\mathbf{g}_\kappa} + \mathbf{g}_{hom} \tag{2.11}$$

where $\mathbf{g}_{tot}$ is a vector containing the total response for each timepoint at each source-detector pair. The matrices $\mathbf{A}_\eta$ and $\mathbf{A}_\kappa$ are as previously stated, while $\mathbf{f}_\eta$ and $\mathbf{f}_\kappa$ are vectors containing the perturbations to the background in the absorption and scattering parameters at each voxel. The two vectors $\mathbf{g}_\eta$ and $\mathbf{g}_\kappa$ represent the appropriate matrix-vector products, while $\mathbf{g}_{hom}$ is a vector of the homogeneous responses, computed using (2.7). In the course of this research, attention was paid exclusively to imaging of the absorption parameter. To do this, γ^2 was presumed to be perfectly homogeneous, making $\mathbf{f}_\kappa$ a vector of all zeros. The matrix $\mathbf{A}_\kappa$ then becomes irrelevant and is dropped from the above equation, leaving only $\mathbf{A}_\eta$. Because only one matrix and one perturbation vector are used, when the symbols $\mathbf{A}$ and $\mathbf{g}$ are used further in this thesis, they will refer to $\mathbf{A}_\eta$ and $\mathbf{g}_\eta$ respectively unless otherwise noted.

2.2 Inversion Methods

With a forward model established, the next step is to determine what method to use in solving the inverse problem. it. As we have constructed a linear forward model, we are looking at methods by which to invert the system:

$$\mathbf{A}\mathbf{f} = \mathbf{g} + \mathbf{n} \tag{2.12}$$

For some object $\mathbf{f}$, some data $\mathbf{g}$, noise $\mathbf{n}$, and a forward matrix $\mathbf{A}$ which is overdetermined, and very poorly conditioned.

We make use of two different algorithms to solve this system. The first is a classical Tikhonov regularized least squares solution. While this method offers a solid solution, its accuracy depends largely on the determination of the regularization parameter, and its computational requirements can become quite large.

The second method employed is LSQR. This is a Krylov subspace based iterative method which avoids one of the primary problems associated with least squares: the need to compute $\mathbf{A}^T\mathbf{A}$ and solve the associated linear system. By interacting with $\mathbf{A}$ only through matrix-vector products, LSQR is able to arrive at a solution much more rapidly than the Tikhonov method. The primary difficulty with LSQR is finding an appropriate stopping criterion. If an appropriate stopping criterion is not found, the solution can simply be examined after each iteration, adding additional iterations as necessary until an appropriate solution is found.

2.2.1 Obtaining Perturbation Data

Because the Born Approximation is a perturbation method, we are only concerned with that part of the total output due to the perturbations (i.e. - the $\mathbf{A}_\eta \mathbf{f}_\eta$ component of Eq (2.11)). This quantity is not, however, directly measurable. Only the values contained in $\mathbf{g}_{tot}$ are measurable. Because $\mathbf{f}_\kappa$ is zero, $\mathbf{g}_\kappa$ is also zero and can be disregarded, leaving only $\mathbf{g}_\eta$ and $\mathbf{g}_{hom}$. Determining the needed quantity $\mathbf{g}_\eta$ then becomes only an issue of determining $\mathbf{g}_{hom}$ and subtracting it from the total response.

If the exact background parameters are known, then computing $\mathbf{g}_{hom}$ is a matter of nothing more than evaluating (2.7) a number of times. When the exact background parameters are not known, another method must be utilized. The first option is to examine the data and extrapolate the appropriate background parameters. These parameters, combined with an appropriately estimated t_0 (the time at which the impulse occurs) and correct source and detector amplitudes, can then be used to generate the needed homogeneous response. This method is in theory the most accurate method, as it is customizable to the differences in amplitudes between sources and detectors, as well as being based in the fundamental physics. Additionally, the background absorption and scattering parameters need to be estimated in some manner before the sensitivity matrices $\mathbf{A}_\eta$ and/or $\mathbf{A}_\kappa$ can be computed, and in the course of implementing this method, those values are indeed estimated.

In practice, some additional knowledge is necessary to obtain this type of solution. Source amplitudes and detector sensitivities can vary, leading to data sets with widely varying amplitudes. This forces one to additionally solve for the amplitude of each of the sources and detectors in order to obtain some level of normalization. To do this, one must know exactly which sources and detectors are unique, and where each is being utilized. In our case, however, these data were unavailable, and additionally, the simulated data set was prenormalized. Because of this, the method of generating the background field used here is significantly different. Due to a large number of repetitions in the source-detector geometry, the background fields would be identical for a large number of source-detector pairs, given perfect homogeneity and normalization. This relation is exploited by simply taking the mean response across all similar source-detector pairs, and using that mean as the homogeneous response. This method is similar to methods previously utilized successfully by others [13], and is significantly faster than attempting to match on a parameter level. The background parameters used in the generation of $\mathbf{A}$ are then estimated from this mean value.

2.2.2 Least Squares

In the previous section we obtained a discretized model for the perturbed system characterized by the equation:

$$\mathbf{g} = \mathbf{A}\mathbf{f} \tag{2.13}$$

where $\mathbf{f}$ is a vector containing the values of the absorption perturbations at each voxel, and $\mathbf{g}$ is a vector giving the perturbations in measurement values. Solving this system using a least squares approach amounts to solving the following error minimization problem:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} ||\mathbf{g} - \mathbf{A}\mathbf{f}||_2^2 \tag{2.14}$$

This gives an overall solution of:

$$\hat{\mathbf{f}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{g} \tag{2.15}$$

The poor conditioning of $\mathbf{A}^T\mathbf{A}$, however, necessitates the use of some type of regularization to stabilize the result. To do this, we used Tikhonov regularization. The form of this minimization problem is:

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}}[||\mathbf{g} - \mathbf{A}\mathbf{f}||_2^2 + \lambda||\mathbf{R}\mathbf{f}||_2^2] \tag{2.16}$$

The addition of the second term in the minimization problem constrains the solution, depending on how the regularization matrix $\mathbf{R}$, and corresponding regularization parameter λ, are chosen [14]. Common choices of $\mathbf{R}$ include the identity matrix, which constrains the size of the result, and various types of derivative operators, all of which act to smooth the result in some manner. Solving the above equation using a method similar to the previous minimization system gives a result of:

$$\hat{\mathbf{f}} = (\mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{R}^T\mathbf{R})^{-1}\mathbf{A}^T\mathbf{g} \tag{2.17}$$

Here, initially, an identity matrix was used for $\mathbf{R}$. In that case, the choice of λ is directly related to the SVD of the forward matrix. It acts as a scaling coefficient, reducing the relative contribution of those singular values less than λ, and increasing the relative contribution of those greater than λ. The choice of λ also dictates the SNR that can be tolerated by the system. For a full discussion of this topic, see [14]. For a smaller $\mathbf{A}$ with a well defined split in singular values, doing a full singular value decomposition and examining the singular values enables one to accurately determine the value of λ necessary for a particular SNR. This is not possible in this case, however, due to the size of the matrix $\mathbf{A}$. As such, selection of the appropriate value of λ has been based primarily on L-Curve type methods

Another regularization matrix we used was a three dimensional gradient. Three separate gradient matrices were generated, one for each dimension. Each of these matrices is a first-order gradient matrix, simply taking the gradient to be equal to the difference between the value at a voxel and each of its neighbors in the corresponding directions. This gives a solution of:

$$(\mathbf{A}^T\mathbf{A} + \lambda_x^2\mathbf{L}_x^T\mathbf{L}_x + \lambda_y^2\mathbf{L}_y^T\mathbf{L}_y + \lambda_z^2\mathbf{L}_z^T\mathbf{L}_z)\mathbf{f} = \mathbf{A}^T\mathbf{g} \tag{2.18}$$

where $\mathbf{L}_x$, $\mathbf{L}_y$ and $\mathbf{L}_z$ are the gradient matrices along the X, Y, and Z directions, respectively. If we make all three regularization parameters identical, the equation reduces to:

$$(\mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{R}^T\mathbf{R})\mathbf{f} = \mathbf{A}^T\mathbf{g} \tag{2.19}$$

where $\mathbf{R}^T\mathbf{R}$ is the sum of the inner products of the three individual gradient matrices. It was found through experimentation on simulated data that this method, utilizing the gradient matrix for regularization, gave superior results to the use of the identity. This makes sense intuitively, because while the absolute value of the perturbation may not necessarily be small, it is a reasonable assumption that the objects we are trying to image will have some level of smoothness. All of our simulated results utilize compact buried objects, so the gradient matrix was used as the regularizer for all least squares inversions.

2.2.3 LSQR

A second inversion method utilized in this research was LSQR. LSQR is an algorithm which solves the problem

$$\hat{\mathbf{f}} = \arg\min_{\mathbf{f}} \|\mathbf{g} - \mathbf{A}\mathbf{f}\|_2^2 \tag{2.20}$$

through a method similar to the conjugate gradient method [15]. An interesting feature of this algorithm is that it interacts with the matrix $\mathbf{A}$ only through the matrix-vector products $\mathbf{A}\mathbf{v}$ and $\mathbf{A}^T\mathbf{v}$, for various vectors $\mathbf{v}$. Thus, when $\mathbf{A}$ has sparse structure, LSQR becomes highly attractive for its computational efficiency when compared to a direct solution of traditional least squares. This is due in part to the fact that computing $\mathbf{A}^T\mathbf{A}$ can require large amounts of computational overhead. Also, the number of voxels in a given solution becomes somewhat limited by the necessity of solving the system defined by $\mathbf{A}^T\mathbf{A}$ or some regularized version thereof. As the number of voxels increases, the size of $\mathbf{A}^T\mathbf{A}$ and the computation required for Gaussian elimination both increase much more rapidly than with LSQR. The derivation of LSQR given here is similar to the original derivation given by Page and Saunders [15].

Solving a regularized version of the above system is equivalent to a least squares solution to the system:

$$\begin{bmatrix} \mathbf{A} \\ \lambda\mathbf{R} \end{bmatrix} \mathbf{f} = \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix} \tag{2.21}$$

Using $\mathbf{A}$ to denote the augmented matrix on the left side of (2.21), we have the matrix upon which LSQR operates.

The LSQR algorithm is based on two bidiagonalization procedures, one for generating a lower bidiagonal matrix, the other for generating an upper bidiagonal matrix. The first of these starts by generating a series of vectors and corresponding scalar quantities such that:

$$\begin{aligned}\beta_1\mathbf{u}_1 &= \mathbf{g}, \qquad \alpha_1 v_1 = \mathbf{A}^T\mathbf{u}_1 \\ \beta_{i+1}\mathbf{u}_{i+1} &= \mathbf{A}\mathbf{v}_i - \alpha_i\mathbf{u}_i \\ \alpha_{i+1}\mathbf{v}_{i+1} &= \mathbf{A}^T\mathbf{u}_{i+1} - \beta_{i+1}\mathbf{v}_1\end{aligned} \tag{2.22}$$

In the above equations, α_i and β_i are chosen such that $\|\mathbf{u}_i\| = \|\mathbf{v}_i\| = 1$.

The second bidiagonalization is quite similar to the first. The major differences are the exchange of $\mathbf{A}^T$ and $\mathbf{A}$, and the use of $\mathbf{A}^T\mathbf{g}$ instead of $\mathbf{g}$ as a starting vector. This results in the iterative procedure:

$$\begin{aligned}\theta_1\mathbf{v}_1 = \mathbf{A}^T\mathbf{g}, \qquad \rho_1\mathbf{p}_1 &= \mathbf{A}^T\mathbf{v}_1 \\ \theta_{i+1}\mathbf{v}_{i+1} &= \mathbf{A}^T\mathbf{p}_i - \rho_i\mathbf{v}_i \\ \rho_{i+1}\mathbf{p}_{i+1} &= \mathbf{A}\mathbf{v}_{i+1} - \theta_{i+1}\mathbf{p}_1\end{aligned} \tag{2.23}$$

with ρ_i and θ_i once again being chosen such that $\|\mathbf{p}_i\| = \|\mathbf{v}_i\| = 1$.

If the vector and scalar quantities generated by these two bidiagonalizations are placed into matrices according to a particular order, the above iterative procedures can be rewritten as a series of matrix equations. The definitions associated with the first bidiagonalization (which reduces $\mathbf{A}$ to lower bidiagonal form), are:

$$\begin{aligned}\mathbf{U}_k &= [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k] \\ \mathbf{V}_k &= [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]\end{aligned} \qquad \mathbf{B}_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix} \tag{2.24}$$

while the matrix definitions for the second bidiagonalization (which results in an upper bidiagonal matrix), are:

$$\mathbf{P}_k = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k] \qquad \mathbf{R}_k = \begin{bmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \end{bmatrix} \tag{2.25}$$
$$\mathbf{V}_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

These two sets of definitions enable the two bidiagonalizations to be rewritten. This results in the first bidiagonalization appearing as:

$$\mathbf{U}_{k+1}(\beta_1 \mathbf{e}_1) = \mathbf{g} \tag{2.26}$$

$$\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{B}_k \tag{2.27}$$

$$\mathbf{A}^T\mathbf{U}_{k+1} = \mathbf{V} - k\mathbf{B}_k^T + \alpha_{k+1}\mathbf{v}_{k+1}\mathbf{e}_{k+1}^T \tag{2.28}$$

Here, $\mathbf{e}_N$ represents a vector of arbitrary length, consisting of all zeros, except in the N^{th} location, where there is a 1. Given infinite precision, the equivalencies $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$ would hold, but this orthogonality rapidly disappears in practice [15]. Applying the definitions to the second bidiagonalization gives the following system of equations:

$$\mathbf{V}_{k+1}(\theta_1 \mathbf{e}_1) = \mathbf{A}^T\mathbf{g} \tag{2.29}$$

$$\mathbf{A}\mathbf{V}_k = \mathbf{P}_k\mathbf{R}_k \tag{2.30}$$

$$\mathbf{A}^T\mathbf{P}_k = \mathbf{V}_k\mathbf{R}_k^T + \theta_{k+1}\mathbf{v}_{k+1}\mathbf{e}_k^T \tag{2.31}$$

which, as with the first system, would yield $\mathbf{P}^T\mathbf{P} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$ with infinite precision.

Furthermore, these two bidiagonalizations are interrelated. Given that the $\mathbf{V}_k$ produced by each bidiagonalization will be identical, it can be shown that:

$$\mathbf{B}_k^T\mathbf{B}_k = \mathbf{R}_k^t\mathbf{R}_k \tag{2.32}$$

and that an orthogonal transformation $\mathbf{Q}_k$ exists such that:

$$\mathbf{Q}_k\mathbf{B}_k = \begin{bmatrix} \mathbf{R}_k \\ \mathbf{p} \end{bmatrix} \tag{2.33}$$

Using these two bidiagonalizations, it is possible to develop a method by which to solve the original system of equations. Using the values generated by the first bidiagonalization, we define a series of values in terms of a vector $\mathbf{y}_k$.

$$\mathbf{f}_k = \mathbf{V}_k\mathbf{y}_k \tag{2.34}$$

$$\mathbf{r}_k = \mathbf{g} - \mathbf{A}\mathbf{f}_k \tag{2.35}$$

$$\mathbf{t}_{k+1} = \beta_1\mathbf{e}_1 - \mathbf{B}_k\mathbf{y}_k \tag{2.36}$$

It can then easily be shown that $\mathbf{r}_k$, the residual error vector given some potential solution $\mathbf{x}_k$, satisfies the equation

$$\mathbf{r}_k = \mathbf{U}_{k+1}\mathbf{t}_{k+1}. \tag{2.37}$$

Looking back at (2.35), we can see that minimizing $\mathbf{r}_k$ is equivalent to solving the original error minimization problem. Because $\mathbf{U}_k$ is theoretically orthonormal and bounded, a reasonable method of solution would be to minimize $\|\mathbf{t}_{k+1}\|$. This yields the least squares problem at the heart of LSQR:

$$\hat{\mathbf{y}}_k = \arg\min_{\mathbf{y}} \|\beta_1\mathbf{e}_1 - \mathbf{B}_k\mathbf{y}_k\|_2^2. \tag{2.38}$$

This equation is best solved through the use of a QR factorization, such that:

$$\mathbf{Q}_k \begin{bmatrix} \mathbf{B}_k & \beta_1\mathbf{e}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_k & \mathbf{b}_k \\ \mathbf{0} & \phi_{k+1} \end{bmatrix} = \left[\begin{array}{ccccc|c} \rho_1 & \theta_2 & & & & \phi_1 \\ & \rho_2 & \theta_3 & & & \phi_2 \\ & & \ddots & \ddots & & \vdots \\ & & & \rho_{k-1} & \theta_k & \phi_{k-1} \\ & & & & \rho_k & \phi_k \\ \hline & & & & & \bar{\phi}_{k+1} \end{array}\right] \tag{2.39}$$

Where $\mathbf{Q}_k$ is a product of plane rotations which eliminate the subdiagonals of $\mathbf{B}_k$. Noting that $[\mathbf{R}_k\mathbf{f}_k]$ is nothing more than $[\mathbf{R}_{k-1}\mathbf{f}_{k-1}]$ with an added row and column, and using $\mathbf{R}_k\mathbf{y}_k = \mathbf{f}_k$, we get:

$$\mathbf{g}_k = \mathbf{V}_k\mathbf{R}_k^{-1}\mathbf{f}_k = \mathbf{D}_k\mathbf{b}_k, \tag{2.40}$$

for which the columns of D_k can be found successively from $R_k^T D_k^T = V_k^T$. Setting initial values at $D_0 = x_0 = 0$ gives:

$$\mathbf{d}_k = \frac{1}{\rho_k}(\mathbf{v}_k - \phi_k\mathbf{d}_{k-1}) \tag{2.41}$$

$$\mathbf{f}_k = \mathbf{f}_{k-1} + \phi_k\mathbf{d}_k, \tag{2.42}$$

which gives us a method of iteratively updating an estimate of the object values $\mathbf{f}$ in $\mathbf{Af} = \mathbf{g}$.

To summarize, the resulting iterative solution method is characterized by the following steps:

(1) Initialize
$$\begin{array}{rclrclrcl} \beta_1 u_1 &=& \mathbf{g}, & \alpha_1 v_1 &=& \mathbf{A}^T u_1, & w_1 &=& v_1 \\ \bar{\phi}_i &=& \beta_1, & \bar{\rho}_1 &=& \alpha_1 & x_0 &=& 0 \end{array}$$

(2) For $i = 1, 2, 3, \ldots$ do steps 3-6

(3) Bidiagonalization

$$\begin{aligned} \beta_{i+1} u_{i+1} &= \mathbf{A} v_i - \alpha_i u_i \\ \alpha_{i+1} v_{i+1} &= \mathbf{A}^T u_{I+1} - \beta_{i+1} v_i \end{aligned}$$

(4) Orthogonal Transformation

$$\begin{aligned} \rho_i &= (\bar{\rho}_i^2 + \beta_{i+1}^2)^{1/2} \\ c_i &= \bar{\rho}_i / \rho_i \\ s_i &= \beta_{i+1} / \rho_i \\ \theta_{i+1} &= s_i \alpha_{i+1} \\ \bar{\rho}_{i+1} &= -c_i \alpha_{i+1} \\ \phi_i &= c_i \bar{\phi}_i \\ \bar{\phi}_{i+1} &= s_i \bar{\phi}_i \end{aligned}$$

(5) Update

$$\begin{aligned} \mathbf{f}_i &= \mathbf{f}_{i-1} + (\phi_i / \rho_i) w_i \\ w_{i+1} &= v_{i+1} - (\theta_{i+1} / \rho_1) w_i \end{aligned}$$

(6) Check For Convergence

The final point of interest involves step #6 in the preceding sequence. The question of how to test for convergence is an important question when solving iterative systems.

It is necessary to find a way through which it can be determined when a suitable solution has been reached. With too few iterations, the solution will be a smoothed version of the actual solution, somewhat similar to overregularizing with Tikhonov inversion. Likewise, too many iterations will potentially result in a significant amount of error entering the system, which will dominate the solution and render it useless.

The most straightforward method of checking for convergence is to simply monitor the size of the residual error $\|\mathbf{g} - \mathbf{Af}\|$. While this is theoretically the optimal way of solving for a system in which the matrix is an accurate representation of the forward model, it does not perform as well in cases where the matrix is overdetermined or simply a non-ideal representation of the system. In these cases, the residual error may converge to a number greater than zero. Unless this value to which it converges is known (which is rather unlikely), it is difficult if not impossible to know when the solution has converged sufficiently. In such cases, another method of determining convergence is to use $\|\mathbf{A}^T\mathbf{r}\|_2^2 < \varepsilon$ [16] instead of the norm of the residual. In the course of this work, however, it was found that neither criterion converged sufficiently well to serve as an indicator of a good solution. Because of this, results with LSQR were simply computed for a large range of iteration values (by simply saving the estimate at each step), and then observing and selecting an appropriate result. This is a sufficient result for this work as we are interested more in the comparison of our interpolated results against fully computed results than we are in the absolute error involved. Were the methods outlined in this thesis to be used in a clinical situation, an appropriate convergence parameter would need to be determined to give a numerical method of selecting the optimal solution.

Chapter 3

Born Matrix Generation

3.1 System Description

The configuration under consideration in this project is a time-domain DOT system in a transmission configuration. A grid of sources is located on the surface of the tissue at $z = 0$. For each source, a series of detectors is located in the plane $z = z_d$, arranged symmetrically around the source, with an additional detector located immediately underneath the source location.

A near infrared laser is pulsed for a time period on the order of picoseconds for a single source location at $t = 0$ and data are then collected for the associated detectors. This process is then repeated for each of the source locations. This creates a system where the source-detector geometry is unchanging from location to location, as both the source and the associated detectors are raster scanned across the X-Y plane.

This work deals primarily with the inversion of the full first Born approximation matrix. Other solutions have been suggested, such as a method based on the Mellin transform (the n^{th} time weighted integral) of the data [17]. These methods simplify the computation by operating on some statistic of the information, rather than the information itself. The work presented here instead attempts to use all available spatio-temporal information, and optimizes the methods by which that information is processed. Future work may incorporate the use of some type of statistical information to further increase the speed of computation beyond what is gained through the methods presented here.

3.2 Matrix Generation

Looking at (2.6), (2.7) and (2.8), it is clear that a significant amount of computation is required to explicitly calculate every value in the matrix **A**. The primary goal of this research was to determine a method by which **A**, or some reasonable approximation of **A**, could be generated with significantly less computation. The two main methods by which this was done were symmetry exploitation and interpolation.

Symmetry exploitation takes advantage of specific features of the system configuration to minimize the number of times that the Green's functions need to be explicitly calculated. Based on a combination of sensor geometry and symmetries in the solution to the diffusion equation, excess computations are eliminated. In their place, a series of sparse, easily constructed matrices are developed which allow for the decomposition of **A** into a series of multiplications utilizing these sparse matrices, along with a core matrix of highly reduced size. This decomposition and reduction of computations allows for a large decrease in the computational complexity associated with computing **A**, as well as a marked reduction in the amount of memory required for both storage and matrix utilization.

To further improve the speed of computation beyond what is possible with symmetry exploitation, interpolation was used. Rather than compute all of the points dictated by the symmetry exploitation, a small number of values are computed, appropriately spread throughout the reduced-dimensional solution space. The continuity and smoothness of the true values are then exploited to extrapolate the remaining values using a linear interpolation method. The potential for use of other interpolation methods is also discussed.

Some notation standards which are used throughout this work are presented in Table 3.1. In most cases, these are also described in text prior to being used. Table 3.1 is intended primarily to be a reference source, and a single collection of all notation used.

N_X	Number of Voxels along X-axis
N_Y	Number of Voxels along Y-axis
N_Z	Number of Voxels along Z-axis
N_{XYZ}	Total number of voxels in medium Equal to $N_X * N_Y * N_Z$
N_S	Number of Sources
N_D	Number of detectors per source location
N_{min}	Number of (D_1, D_2, Z) coordinates required to construct $\mathbf{A}$
N_{pt}	Number of explicitly computed points
N_T	Number of timepoints per source-detector pair
(X_s, Y_s, Z_s)	Cartesian Coordinates of Source
(X_d, Y_d, Z_d)	Cartesian Coordinates of Detector
(X_v, Y_v, Z_v)	Cartesian Coordinates of Voxel
Z_d	Thickness of Slab

Table 3.1: Notation Standards

3.2.1 Symmetry Exploitation

The first step that was taken to reduce the amount of time required for computation of the forward matrix was to minimize the total number of computations being performed. The specific structure present in the device configuration used for this work results in a large number of repeated calculations in explicitly generating the forward Born matrix. Instead of executing these calculations multiple times, what was done was to calculate the values once, and reuse them whenever required.

When the volume over which (2.8) is integrated is discretized into uniform voxels, the equation reduces to:

$$\begin{aligned} J^{(\Gamma)}_{p,slab}(\xi,\zeta,t;\mathbf{r}') \approx -\, dV \int_{-\infty}^{\infty} & dt' [g^{(\Phi)}_{slab}(\mathbf{r}',\zeta,t-t')\eta(\mathbf{r}')g^{(\Gamma)}_{slab}(\xi,\mathbf{r}',t') - \\ & \kappa(\mathbf{r}')\nabla_{\mathbf{r}'} g^{(\Gamma)}_{slab}(\xi,\mathbf{r}',t') \cdot \nabla_{\mathbf{r}'} g^{(\Phi)}_{slab}(\mathbf{r}',\zeta,t-t')] \end{aligned} \tag{3.1}$$

In (3.2), $\mathbf{r}'$ is the center of the voxel in question, while dV is the volume of the voxel. The source location is indicated by ξ, and ζ is the detector location. It should also be noted that this research was concerned only with perturbations in the absorption parameter. Because of this, $\kappa(\mathbf{r}')$ is presumed to be zero, and the second term inside the integral can be dropped. The resulting equation,

$$J^{(\Gamma)}_{p,slab}(\xi,\zeta,t;\mathbf{r}') \approx -\, dV \int_{-\infty}^{\infty} dt' g^{(\Phi)}_{slab}(\mathbf{r}',\zeta,t-t')\eta(\mathbf{r}')g^{(\Gamma)}_{slab}(\xi,\mathbf{r}',t') \tag{3.2}$$

is used to compute each of the individual matrix values, and is a convolution of two Green's functions. The first Green's function scatters the light from the source to the voxel in question, while the second scatters the light from the voxel to the detector.

Looking at (2.6) and (2.7), it can be seen that the X-Y coordinates of the source, detector, and voxels only enter into the equations as a radial distance $\sqrt{X^2 + Y^2}$. Because of this, the absolute X-Y locations involved become irrelevant, and it becomes possible to change from Cartesian to a dual cylindrical coordinate system as illustrated in Fig 3.1.

However, because the Green's functions are only dependent on the radial distances involved, the kernel of the problem is invariant to the angles θ_1 and θ_2. Thus we can disregard them and instead look at a different space defined by the following relations:

$$\begin{aligned}
D_1 &= ((X_v - X_s)^2 + (Y_v - Y_s)^2)^{1/2} \\
Z_1 &= Z_v - Z_s \\
D_2 &= ((X_v - X_d)^2 + (Y_v - Y_d)^2)^{1/2} \\
Z_2 &= Z_d - Z_v
\end{aligned} \tag{3.3}$$

Here, (X_s, Y_s, Z_s), (X_d, Y_d, Z_d), (X_v, Y_v, Z_v) are the absolute Cartesian coordinate locations of the source, detector, and voxel in question, respectively. While there are four variables listed above, Z_2 and Z_1 are dependent on one another, so only one value, $Z = Z_1$, is needed, which reduces the number of degrees of freedom from four to three. Additionally, Z_s is taken to be equal to zero, which amounts to presuming that the sources are all placed on the same plane, which is then used as the origin of the coordinate system. This gives us a three dimensional space over which the solution to (3.2) exists. These equations define a mapping $\Re^9 \rightarrow \Re^3$ that is onto. The fact that this mapping is not one to one can be exploited to reduce the number of computations necessary to determine the matrix $\mathbf{A}$.

If all of the source-voxel-detector combinations required to construct $\mathbf{A}$ are converted into (D_1, D_2, Z) notation, one can obtain a set of $(N_S * N_D * N_x * N_y * N_z)$ ordered triplets, which we define as S_{pts}. However, there are a large number of repeated elements in S_{pts}. If a second set, S_{min}, is defined such that it contains all of the unique values in S_{pts}, then we can obtain a minimal set of points which includes all points required

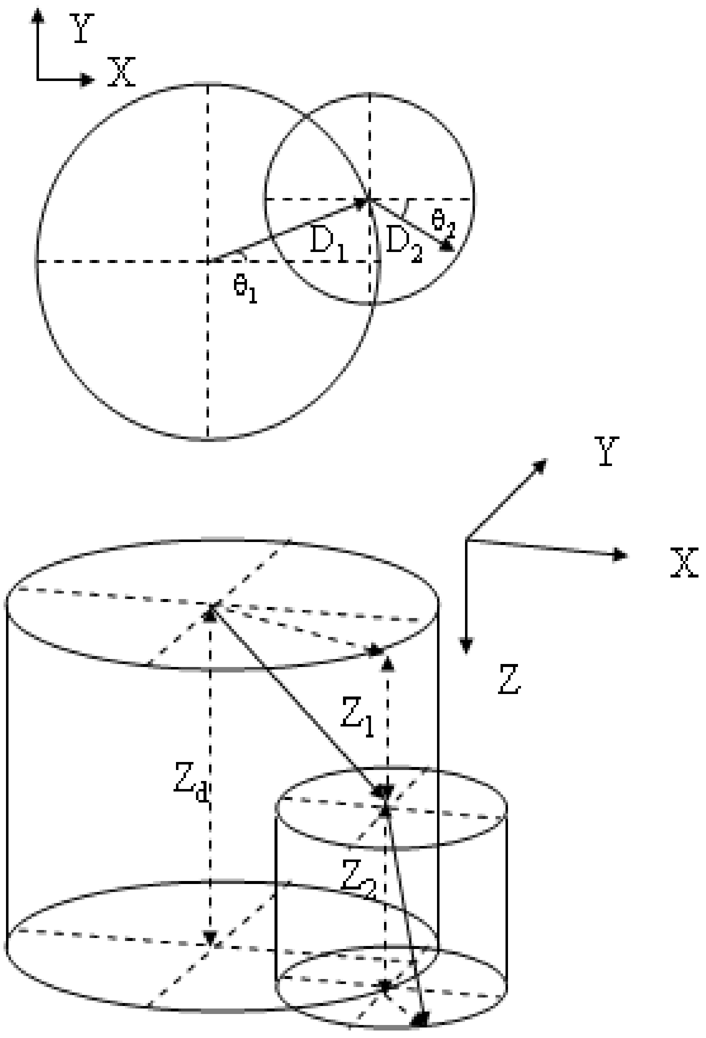

Figure 3.1: Dual Cylindrical Coordinate System

to determine the forward matrix $\mathbf{A}$. This set, by definition, contains exactly N_{min} ordered triplets. By only computing the elements of $\mathbf{A}$ for each of the coordinates in this minimal set once, we can eliminate a large number of excess computations that would otherwise be executed were the system to be viewed in the original nine dimensional solution space. Once this minimal set is computed, the results can be repeatedly used to quickly build the forward matrix $\mathbf{A}$. Instead of requiring additional computations for each source-detector pair, a simple referencing scheme is all that is needed to access the already computed values.

Using the above method of computation across the minimal set, a series of matrices can be developed which express the full forward matrix $\mathbf{A}$ as a product of sparse matrices and a matrix containing the minimal set of data. Starting with the equation

$$\mathbf{Af} = \mathbf{g} \tag{3.4}$$

and working backwards, this series of matrices can be developed, leading to an overall system for this method of data reuse.

We start by defining a matrix $\mathbf{A}_{sm}$, of size $N_{min} \times N_T$. Here, the subscript $\mathbf{A}_{sm}$ indicates that this is the "small" version of the matrix $\mathbf{A}$. All distinct values present in $\mathbf{A}$ are present in $\mathbf{A}_{sm}$; they simply need to be properly selected/permuted to obtain $\mathbf{A}$. Each column of this matrix contains the values associated with a particular time for each of the N_{min} (D_1, D_2, Z) triplets in S_{min}. Looking across each row yields all of the time values for a single (D_1, D_2, Z) triplet. Thus, for each source-detector combination involved in the reconstruction, the forward matrix associated with that pair can be constructed by rearranging a subset of the rows of $\mathbf{A}_{sm}$. This means that $\mathbf{A}$ can now be represented in block form as:

$$\mathbf{A}^T = [\mathbf{S}_{11}\mathbf{A}_{sm} \quad \ldots \quad \mathbf{S}_{1D}\mathbf{A}_{sm} \quad \mathbf{S}_{21}\mathbf{A}_{sm} \quad \ldots \quad \mathbf{S}_{ND}\mathbf{A}_{sm}] \tag{3.5}$$

Each matrix $\mathbf{S}_{nd}$ is of size $(N_X * N_Y * N_Z) \times (N_{min})$, and is the matrix responsible for selecting the appropriate rows from $\mathbf{A}_{sm}$. Each of these selection matrices also has a significant amount of structure. If we order the rows of $\mathbf{A}_{sm}$ such that, as you step from row to row, you step first across all values of D_1 and D_2, then to the next Z value, the matrices $\mathbf{S}_{nd}$ will all have a block diagonal structure. If $\mathbf{A}_{sm}$ is thought of as having block structure:

$$\mathbf{A}_{sm} = \begin{bmatrix} \mathbf{Z}_1 \\ \mathbf{Z}_2 \\ \vdots \\ \mathbf{Z}_{N_z} \end{bmatrix} \tag{3.6}$$

where N_z is the number of voxels along the Z-dimension, then each of the $\mathbf{Z}_j$ blocks corresponds to the pieces of $\mathbf{A}_{sm}$ required to build the matrix for a single X-Y slice. Because the values selected from each X-Y slice are going to be the same for a single source-detector pair, the selection matrix will be identical as well. Thus,

$$\mathbf{S}_{ij}\mathbf{A}_{sm} = \begin{bmatrix} \tilde{\mathbf{S}}_{ij}Z_1 \\ \tilde{\mathbf{S}}_{ij}Z_2 \\ \vdots \\ \tilde{\mathbf{S}}_{ij}Z_N \end{bmatrix} \tag{3.7}$$

or, alternately:

$$\mathbf{S}_{ij} = \mathbf{I}_{N_z} \otimes \tilde{\mathbf{S}}_{ij} \tag{3.8}$$

Each of the $\tilde{\mathbf{S}}_{ij}$ is a selection matrix; it consists entirely of ones and zeros, and each of size $(N_X * N_Y) \times (N_{min}/N_Z)$. Presuming that a slab geometry is used, N_{min}/N_Z will always be an integer because N_{min} will consist of N_z sets of points, each varying only in their Z value. Each row consists of exactly one non-zero value, which is a one. Every column has either all zeros or one nonzero value.

If one so desired, it would be possible to generate the entire forward matrix as a series of sparse matrices and Kronecker products. Factoring out the $\mathbf{A}_{sm}$ from (3.5) and replacing it with a Kronecker product yields:

$$\mathbf{A}^T = [\mathbf{S}_{11} \quad \ldots \quad \mathbf{S}_{1N_D} \quad \mathbf{S}_{21} \quad \ldots \quad \mathbf{S}_{N_S N_D}] * [\mathbf{I}_{N_S * N_D} \otimes \mathbf{A}_{sm}] \tag{3.9}$$

In practice, however, this computation is unnecessary. Due to the fact that the data from each source-detector pair can be worked with separately, and in any desired order, it is much more efficient to store $\mathbf{A}_{sm}$ only once, and generate the selection matrices as needed.

3.2.2 Interpolation

The next step taken to improve the computation time was to look at how $\mathbf{A}_{sm}$ is computed. Explicitly computing every value in this matrix was still time consuming, although significantly less so than computing all of the values in $\mathbf{A}$. Our goal was to interpolate the majority of the values in $\mathbf{A}_{sm}$ from a small number of explicitly calculated ones. Presuming that the method of interpolation chosen requires less computation than calculating the values outright, this will result in a further increase in the speed with which the forward matrix can be computed.

Looking at $\mathbf{A}_{sm}$ as a matrix reveals that it too has a high degree of structure. Each column of $\mathbf{A}_{sm}$ consists of the forward matrix values for every possible relative voxel location at one specific time. If these values could be obtained through interpolation from a smaller number of points, and that interpolation operator represented as a matrix, $\mathbf{A}_{sm}$ could be reduced to a product of two matrices. The first would implement the interpolation, while the second would contain the required input values. Hence, $\mathbf{A}_{sm}$ could be represented as:

$$\mathbf{A}_{sm} = \mathbf{Q}_{interp}\mathbf{V}_{comp} \tag{3.10}$$

where, $\mathbf{Q}_{interp}$ is the matrix responsible for the linear interpolation, while $\mathbf{V}_{comp}$ are the explicitly computed values. The sizes of these matrices are $N_{min} \times N_{pt}$ and $N_{pt} \times N_T$ respectively, where N_{pt} is the number of points explicitly computed.

For this work, a linear interpolation method was chosen to be implemented. This choice was based on the fact that a linear interpolation method is easily implemented as a matrix, and only requires that (3.2) be evaluated at a few sample points. An additional benefit of using a simple linear interpolation is the low computational complexity associated with multiplying by such sparse matrices.

The method of linear interpolation used for this research was based around a De-Launay tessellation of the 3-D space created by D_1, D_2 and Z, and utilizes two sets of points. The first set is the set of sample points. The values at these points are explicitly computed and are the input values to the interpolation. The second set is a set of interpolation points. These are the points whose values will be the output of the interpolation. The DeLaunay tessellation uses the sample points as vertices to describe a set of tetrahedrons. These tetrahedrons are defined in such a way that no sample point is circumscribed by any tetrahedron; they exist only at the vertices. To determine the value at each point in the set of interpolation points, one first determines inside which tetrahedron the point lies. This tetrahedron will be defined by four sample points, each of which contributes its value to the interpolation. To determine the weights associated with each input value, barycentric coordinates are utilized. The three dimensional barycentric coordinates of the point to be interpolated are computed with respect to the four sample points which comprise the encircling tetrahedron. These weights are used directly as the weights of the interpolation. Multiplying each of these weights by the value at the associated sample point and summing the result yields the value at the point being interpolated. This method of interpolation is the same as utilized in the Matlab *griddata()* function. If precision is required beyond what is achievable with this method of linear interpolation, another method of interpolation could easily be substituted in its place. For interpolation methods which can be represented in sparse matrix format, the gains seen in §(3.4.2) would still apply, although some reformulation of the exact gain value would be needed. Methods which cannot be represented in matrix format are still viable as methods of reducing the time required to compute $\mathbf{A}_{sm}$, but will not have the gains involved in matrix-vector multiplications, detailed in §(3.4.2).

In looking to utilize a basic linear interpolation such as we have used, one aspect of the function which needs to be examined is its smoothness. If the function does not have at least some degree of smoothness, a linear interpolation method is likely to result in a high degree of error in the output points. While this can potentially be countered by higher sampling densities, an increased number of sample points can drastically decrease the computational gains that are the purpose of utilizing the interpolation.

In this particular case, the critical question is whether the function is smooth across the (D_1, D_2, Z) space. Looking at the triplets in S_{min} and taking all of those associated with a single Z value, a slice in the D_1-D_2 plane can be studied. Plotting the values of $D1$ along the X-axis, $D2$ along the Y-axis, and the associated values along the Z-axis, a smooth surface is obtained. Plots of this surface are shown in Fig 3.2 and Fig 3.3.

Looking at the first plot, it can easily be seen that (3.2) smoothly varies as D_1 and D_2 are changed. Equally evident, however, is the exponential relationship that the values of both D_1 and D_2 have with the associated matrix value. This exponential relationship requires some additional thought in order to accurately interpolate the values. It should be noted that while values are only shown for a strip around the $D_1 = D_2$ line, they exist for all D_1-D_2 combinations. The strip arises from the use of only those points associated with a raster scanned source-detector combination. Because the detectors are always at a fixed X-Y distance from the source, the value of $|D_1 - D_2|$ had a fixed maximum, which leads to the use of only those values lying inside the strip.

The other question of smoothness which needs to be posed involves the values along the Z dimension. Fixing the value of D_1, and plotting the matrix values for all D_2-Z pairs, a plot can be generated to show the smoothness along these dimensions. Figures 3.4 and 3.5 show two different views of this surface. In Fig 3.4, the curve of values along the Z-axis can clearly be seen. While the value is rapidly changing across the Z-axis, the change is much more linear than along the D_1 and D_2 axes. Because of this, linear sampling along the Z-axis is likely to be sufficient to obtain reasonable results. Fig 3.5 one again shows the exponential relationship between D_2 and the resulting matrix value.

Having examined the plots along two slices of the (D_1, D_2, Z) space in which the solution to (3.2) lies, it is clear that the solution is smooth along all three dimensions. In examining these plots, however, an additional issue has arisen. Due to the exponential relationship both D_1 and D_2 have with the associated matrix values, it is unlikely that simply linearly sampling the space and then interpolating will yield sufficient results. When several test results were done using straight linear sampling, this presumption was proven correct. Error between the interpolated and fully computed matrices range was far out of the range that would be considered acceptable. A different approach was clearly necessary to be able to interpolate effectively.

Given that there is an exponential relationship between the two axes D_1 and D_2 and the output value of the function, it would seem that a reasonable method to improve the solution would be through the use of a logarithm. By taking the natural logarithm of each of the source data points, the dynamic range is compressed, and simultaneously

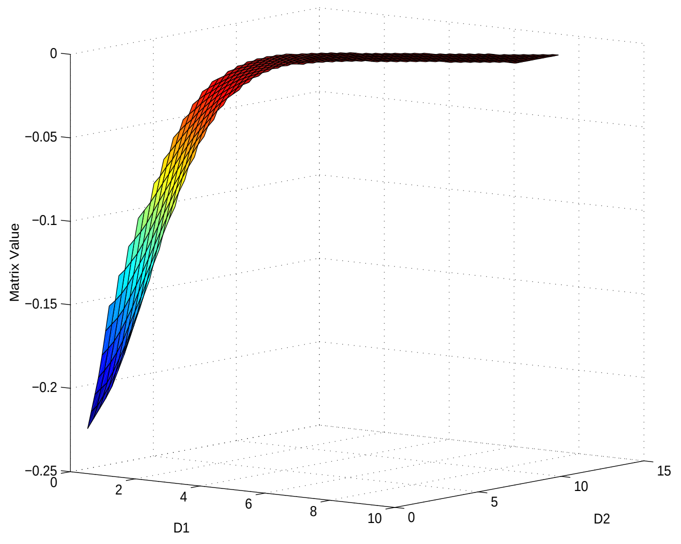

Figure 3.2: View #1 of values for all D_1-D_2 pairs for a fixed Z value

This view gives a clear profile of the exponential drop in value that (3.2) undergoes as D_1 and D_2 approach the origin.

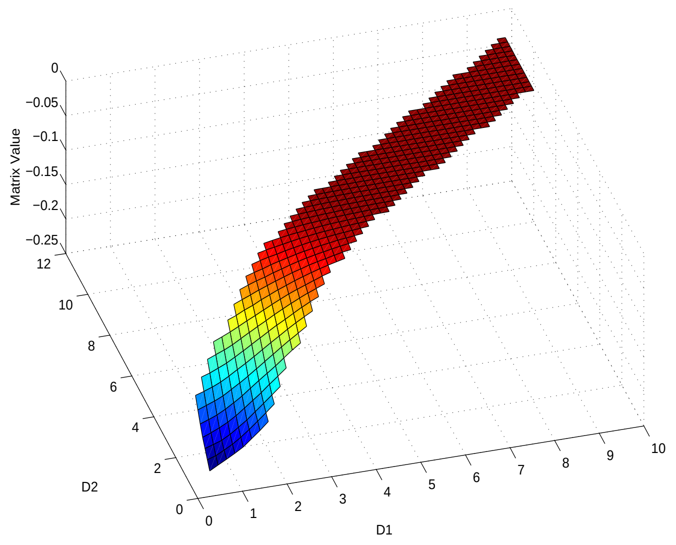

Figure 3.3: View #2 of values for all D_1-D_2 pairs for a fixed Z value

This view shows more direct downward view of the D_1-D_2 plane, showing the narrow strip of the plane that is actually utilized, as well as more clearly showing the actual D_1-D_2 values associated with the rapidly changing parts of the graph.

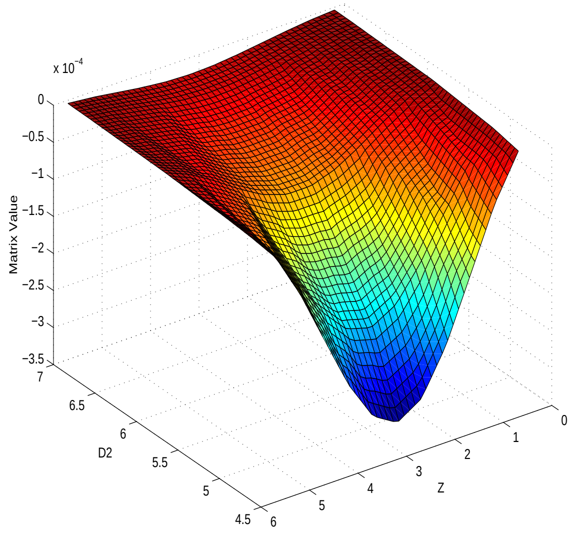

Figure 3.4: View #1 of values for all D_2-Z pairs for a fixed D_1 value.

D_2 values do not reach zero because the fixed value of D_1 limits the range of values which D_2 can take on. This restriction is dependent upon the source-detector configuration.

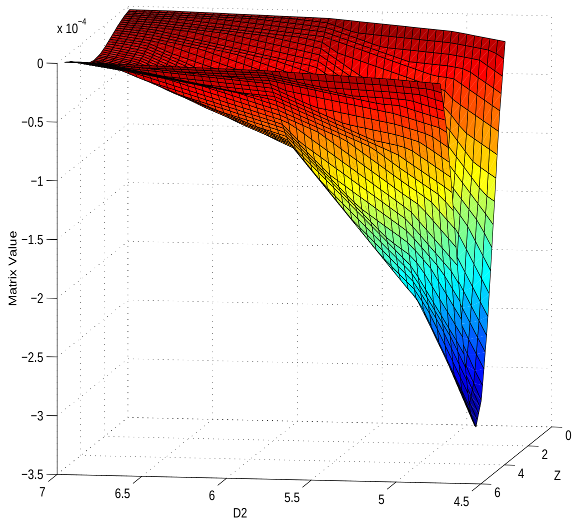

Figure 3.5: View #2 of values for all D_2-Z pairs for a fixed D_1 value.

Here, the exponential relationship between D_2 and matrix values in clearly evident.

the relationship between the matrix value and the two axes values D_1 and D_2 can be transformed into something more closely approximating a linear relationship. By applying the interpolation matrix to this transformed source data, and then taking the inverse logarithm of the resulting output, much more accurate results could be obtained. This method of logarithmically compressing the dynamic range will now be referred to as LCDR. This is as opposed to the use of the full dynamic range, now referred to as FDR.

Using the natural logarithm to compress the dynamic range yields an overall procedure for obtaining A_{sm} of:

(1) Compute $\mathbf{V}_{comp}$

(2) Compute $\mathbf{Q}_{interp}$

(3) Take ln() of each value in $\mathbf{V}_{comp}$

(4) Multiply resulting matrix by $\mathbf{Q}_{interp}$

(5) Take exp() of product to obtain interpolated version of $\mathbf{A}_{sm}$

The primary issue with this method is that while it provides accurate results, it provides absolutely no performance gain. Due to the need to evaluate so many logarithms and exponentials, it is fastest to simply compute $\mathbf{A}_{sm}$ (As $\mathbf{Q}_{interp}\mathbf{V}_{comp}$) once and reuse it for every source-detector pair. While this still provides an improvement over computing the entire $\mathbf{A}_{sm}$, it does not provide a reduced computational complexity for matrix-vector products involving $\mathbf{A}$. Given that such an improvement was a goal of this work, a method was sought which would eliminate the need to transform the data into a more linear relationship.

If the full dynamic range (FDR) is instead used, a different procedure is necessary to obtain $\mathbf{A}_{sm}$. It becomes only a matter of two steps, both of which were part of the previous procedure:

(1) Compute $\mathbf{V}_{comp}$

(2) Compute $\mathbf{Q}_{interp}$

For each source-detector pair, the appropriate $\mathbf{S}_{ij}$ is multiplied by $\mathbf{Q}_{interp}$, yielding a very large sparse matrix. Data is projected onto $\mathbf{V}_{comp}$, and the resulting vector is multiplied by the $\mathbf{S}_{ij}\mathbf{V}_{comp}$ product. This series of operations reduces the computational complexity by a level proportional to the amount of interpolation being performed. These gains are detailed in §(3.4.2).

The ability to represent the interpolation operation as a sparse matrix $\mathbf{Q}_{interp}$ is at the core of the computational gains this method is intended to obtain. By multiplying the sparse $\mathbf{Q}_{interp}$ by each of the $\mathbf{S}_{ij}$, which are also sparse, a significant amount of computation can be saved. In order for this to be able to occur, no operations can be applied between $\mathbf{S}_{ij}$ and $\mathbf{Q}_{interp}$. While step #5 in the previous method could be performed after multiplication with $\mathbf{S}_{ij}$, the repeated evaluation of so many exponents would negate any computational gains. Instead, in order to be able to exploit the sparse matrices for computational gains, our options are limited to using the FDR with a properly selected set of sample points.

In order for this interpolation to operate well, the (D_1, D_2, Z) solution space needed to be efficiently sampled. Given that the data which have had their dynamic range logarithmically compressed have an approximately linear relationship with the variables D_1 and D_2, a linear sampling method was chosen using a dual grid pattern. This method utilizes two grids: A and B. Grid A is simply a grid in the D_1-D_2 plane with linear spacing along each axis. Grid B has the same spacing as Grid A, but is shifted by one half the grid spacing along each axis, with the top row and rightmost column both eliminated. These two grids are alternated for each Z value, which were linearly sampled from the range of Z values, making sure that Grid A is used for both the top and bottom layers of the slab. In the case of an even number of layers in the slab, this will result in Grid A being used for two layers in a row. This condition is necessary in order to ensure that all of the points to be interpolated fall within the volume defined by the source points. This pattern is shown in Fig 3.6, and will be referred to as a linear sampling pattern. When used with the FDR, however, this sampling method was found to give very poor results, detailed in §(3.3.2). Using the LCDR, however, provided for a significant improvement in the error levels. In order to be able to use the FDR, a different sampling method was

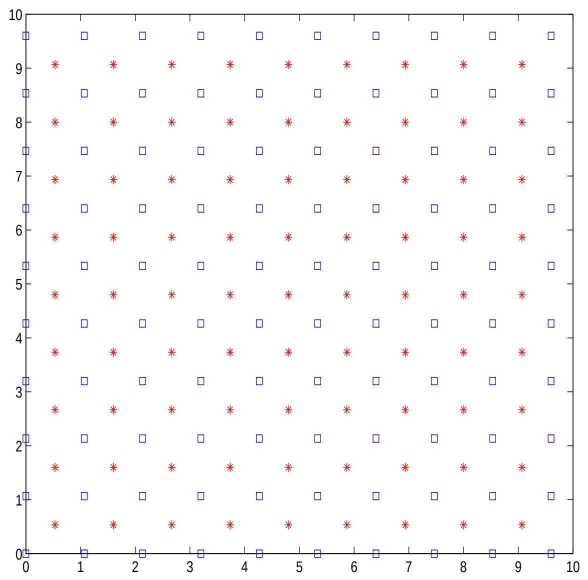

Figure 3.6: Sampling technique used in conjunction with natural logarithm to transform exponential relationship into a more linear one

Here, the points in the D_1-D_2 plane which would be evaluated as part of Grid A are marked by blue squares. Points associated with Grid B are marked by red stars.

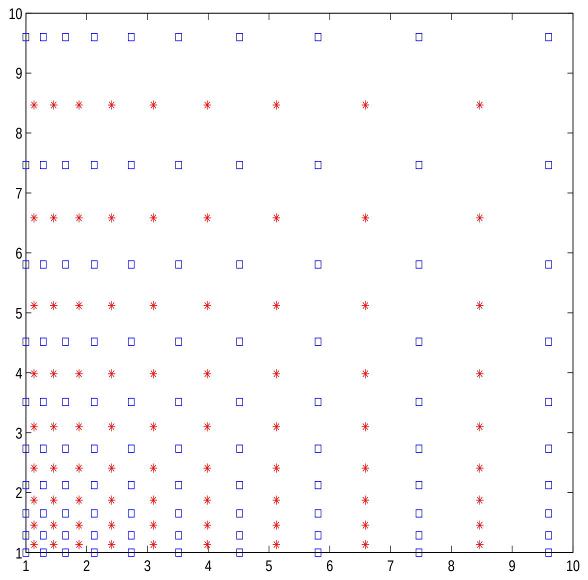

Figure 3.7: Sampling technique used directly with source data to enable performance gains through sparse matrix multiplication.

Again, the points in the D_1-D_2 plane which would be evaluated as part of Grid A are marked by blue squares. Points associated with Grid B are marked by red stars.

clearly required.

By sampling each of D_1 and D_2 linearly in log-space, it is possible to utilize the FDR without an unacceptable increase in error. By sampling this way, and again using a dual grid system, this time illustrated in Fig 3.7, the system is "linearized" in a different manner. By selecting sample points in this manner, the function between each sample point, while still behaving exponentially, will be more easily modelled by a linear relationship. This amounts to fitting the sample points to the data, rather than the data to the sample points, as was previously done.

We are now presented with two choices of dynamic range, as well as two choices of sampling method. This yields a total of four possible combinations. While the both sampling patterns perform better with the LCDR than with the FDR, the error produced using the FDR with the logarithmically spaced sample points is sufficiently small, and computational gains can still be exploited. In many cases, the error produced through the use of logarithmically spaced sample point with the FDR was actually less than that resulting from the use of the LCDR and linearly sampled data points. Because of these results, the details of which are in §(3.3.2), all solutions obtained for the results section of this thesis were obtained using the FDR with a logarithmic sampling method.

3.2.3 A Specific Case

In the previous two sections we have developed a method by which the amount of computation required to generate the first order Born matrix associated with a time domain DOT system can be drastically reduced. This method makes use of underlying symmetries which are present in both the source-detector geometry and the underlying Green's functions used to compute the matrix weights. While these redundancies are readily evident from the equations, in the general case they are not so easy to recognize directly from the source-detector configuration, and the voxelization of the space. Being able to identify these redundancies from the system configuration, as well as the equations, is useful in obtaining a more intuitive understanding of what is occurring.

As just stated, in a generalized source-detector configuration, it is difficult to visualize exactly where all of these numerous redundancies are occurring. In the case where both sources and detectors are arranged on grids, with all of the detectors collecting data from each source, the level of redundancy can be in the thousands. Attempting to visualize this level of complexity is difficult at best. However, in the case of a system such as that in use for this work, the situation is significantly different. For this geometry, the redundancies can easily be found and visualized from the source-detector configuration alone. This allows one to see exactly where the redundancies are occurring, and helps in establishing ways in which the problem can be expanded to include either more data or a smaller voxelization, with as minimal an increase in computation as possible.

Looking at the system configuration defined in §(3.1), it can be seen that the detectors are always in the same location relative to the sources. Because of this, as long as a voxel has the same location relative to the source and detector, it will have the same values for (D_1, D_2, Z) associated with it, regardless of which source it is being computed for. This means that the values can be explicitly calculated for a single source and its detectors, for a volume which encompasses all possible relative voxel locations. This amounts to computing the values associated with each triplet in the set S_{min}.

There are two conditions which are met by this system, and enable these symmetries to be present:

(1) The detectors must always be in the same location with respect to the source.

(2) The sources must be placed on a grid where spacing is an integer multiple of voxel size.

The first condition ensures that the structure we are looking for is present. If the detectors are not in the same location with respect to the source, the values associated with it in the forward matrix will differ and require separate computation.

The second condition is required to ensure that the values being used are actually redundant. Imagine two spaces, one for the computed source-detector combination, and one for the desired source-detector combination, both voxelized using the same size voxels. In order for the values associated with each of these voxels to be equal, the two voxelized spaces must line up appropriately. This will only occur, and still maintain relative voxel locations, if the two sources are separated by an integer number of voxels in both the X and Y directions. This condition is easily satisfied through judicious choice of voxel size, given that the sources are located on some type of uniform grid. This source-location grid condition is not as easily assured as it is fixed by the physical experiment, thus we assume that it is the case here.

This concept is best expressed graphically using a simplistic system to explain. Figure 3.8a shows a view of a space voxelized into $4 \times 4 \times 1$ voxels. Sixteen sources are used, located at the corners of each voxel, in the z=0 plane. For each of these sources, there is a single detector located directly underneath the source. To explicitly generate all required matrix values, 400 values would need to be computed (25 sources * 1 detector per source * 16 voxel values per source/detector pair).

If instead, 64 values are computed, corresponding to an $8 \times 8 \times 1$ system with a single source located in the center, an alternate solution is obtainable. Instead of explicitly computing the values for each source location, all that is required is to use the values computed for 3.8b, and select and appropriate subset of them. For example, to obtain the matrix associated with a source at (4,3), the voxels shaded in 3.8c are used, which results in a system that appears like 3.8d. Likewise, 3.8e shows the voxels which would be needed to obtain 3.8f.

Another point of note with this type of system in particular is that as long as the detectors are symmetrically arranged around the source, there is additional redundancy present when computing the $8 \times 8 \times 1$ space (or similar space which is $4\times$ the size of the original). Because of the symmetry of the sources, only one quarter of the larger space actually need be computed. This reduces the required level of computation to the same as would have been required for a single source and its associated detectors with respect to the original space. Thus, in such situations, it is possible to reduce the amount of overhead computation required by a factor roughly equivalent to the number of sources.

3.3 Error Analysis of Rapidly Generated Matrices

An important question to be asked with regards to this use of symmetry and interpolation is how closely the resulting matrices resemble their explicitly computed counterparts.

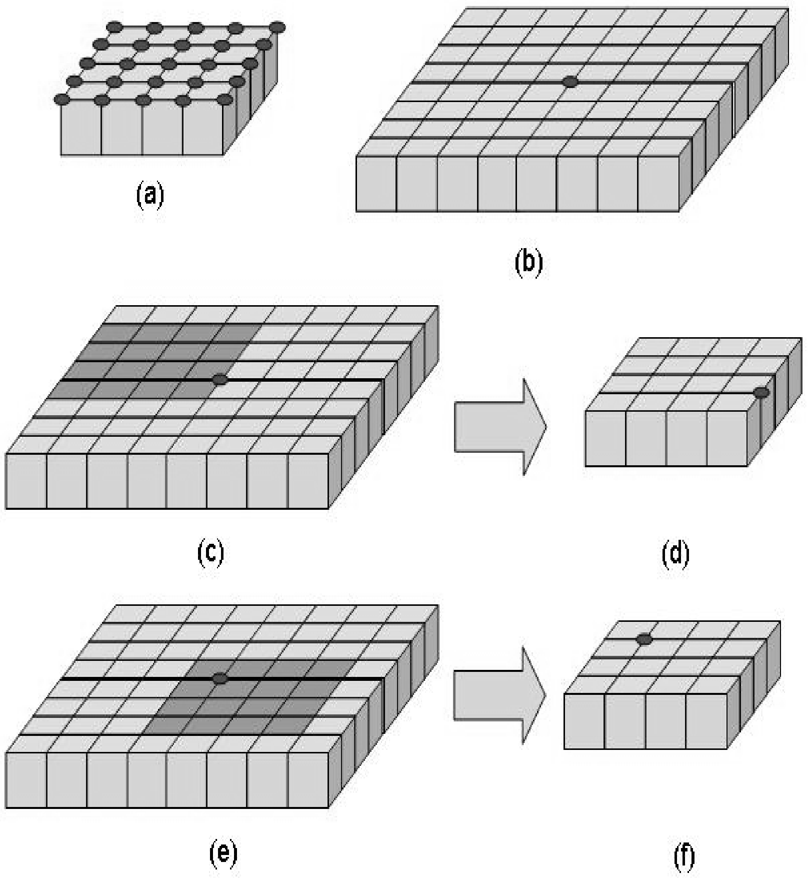

Figure 3.8: Utilizing a single overcomputed source to duplicate the source-detector geometry of an arbitrary source location.

3.3.1 Symmetry Exploitation

In the case of symmetry exploitation, theory states that utilizing the method introduces no error at all into the system. Due to the fact that (3.2) is explicitly calculated for all unique points, the exact value for every point in the set S_{pts} will be present in $\mathbf{A}_{sm}$, and simply needs to be pulled out using the selection matrices. In practice, however, this will likely not be the case. These equations are being evaluated numerically on computers with a finite level of precision, so differences in the final values obtained will occur. These differences will only be on a level associated with machine precision, however.

3.3.2 Interpolation

Unlike the symmetry exploitation, the use of interpolation in constructing the matrix $\mathbf{A}_{sm}$ can introduce a significant amount of error into the matrix. This level of error can range from nearly zero to a significant percentage of the total value, depending on how many points are computed.

To examine the effects of various interpolation levels upon $\mathbf{A}_{sm}$, the matrices $\mathbf{Q}_{interp}$ and $\mathbf{V}_{comp}$ were generated for a volume of $7cm \times 7cm \times 6cm$. This volume was divided into cubic voxels 0.5cm on a side, giving a total of $14 \times 14 \times 12$ voxels, or 2342 total. This volume was used with a variety of interpolation levels to generate $\mathbf{A}_{sm}$ from $\mathbf{Q}_{interp}$ and $\mathbf{V}_{comp}$. For comparison, this was done using all four combinations of dynamic range and sampling method. This was done to give a full comparison of the effects each had upon the interpolation. The interpolated versions of $\mathbf{A}_{sm}$ were then compared with the fully computed $\mathbf{A}_{sm}$. Comparison was made on the basis of three different norm errors: Frobenius Norm, Infinity Norm, and One Norm. The results of this comparison are summarized in Table 3.3.2. The errors are all represented in terms of a percentage of the norm associated with the fully computed matrix.

Looking at Table 3.3.2, it can be seen that the best results are obtained through the use of the LCDR with exponentially spaced sample points. The worst results are obtained with linearly spaced points and the FDR. An interesting observation, however, is that the FDR, used with exponentially spaced sample points, produces better results in most cases than the LCDR with linearly spaced points. This suggest that the linear sampling of data points is not an effective method of sampling this space, regardless of whether or not the LCDR is utilized. Because of this, further comparison will only be made between those methods utilizing exponentially spaced sample points.

In three out of every four cases, the LCDR method performs better than the FDR method. Only when the number of samples along the Z-axis is decreased sufficiently does the FDR method outperform the LCDR one. At such low sampling levels, however, the error induced in $\mathbf{A}_{sm}$ increases significantly

In the cases where the FDR method is outperformed, the difference in performance varies widely. At the high and mid sampling levels ($(20, 20, 20)$ and $(15, 15, 15)$), the increase in Frobenius norm error is as much as 400%. Reducing the sampling level to $(10, 10, 15)$, however, actually decreases the difference in performance. At that sampling level, there is only an increase of 50% in the Frobenius norm error as a result of using the FDR.

An interesting point of note is that the performance of the FDR method with log-sampled points seems to be tied more closely with the precise locations of the sample points than the LCDR method. As the sampling density is decreased, the error induced in $\mathbf{A}_{sm}$ by the LCDR method increases as well, regardless of sampling method. This is not the case with the FDR method. Instead, when the D_1-D_2 sampling level is reduced from 15 samples to 10, the error induced in $\mathbf{A}_{sm}$ decreases. This could possibly be a result of the precise location of each of those samples. If the points to be interpolated happen to fall closer on average to computed points, it is likely that the interpolation would improve solely based upon that proximity, rather than the actual number of points being computed. Because of this, one important extension to this work would determine an optimal method of selecting sample points, such that these benefits could be maximally exploited.

Looking at the matrices interpolated from the FDR data, it can be seen that the 1-norm and infinity norm errors seem to be most closely tied to the D_1 and D_2 sampling levels. For many $D_1 - D_2$ sampling levels, these two error metrics stay the same, even when the Z-sampling level is decreased. While this assumption breaks down for the 1-norm error when the Z-sampling level is decreased too far, it holds for the infinity norm. Even decreasing the Z-sampling to its lowest level does not affect the infinity norm error.

Both of these observations pertain solely to the use of the FDR with exponentially spaced sample points. When the LCDR method is used, all of the error values become primarily functions of the number of points computed, rather than specific sampling densities.

After examining the results of the various interpolation methods and levels upon the error in $\mathbf{A}_{sm}$, it was decided that using the FDR method with exponentially spaced points was the best method to be used for this work. While the error levels were not as low as those obtained when using the LCDR, the advantage of being able to reduce the computation time of matrix-vector multiplications is significant. Because of this, the results presented for analysis in Chapter 4 use only the FDR method with exponentially spaced samples. Of the sampling levels which were used for this comparison, two were chosen to be used for the results in Chapter 4. The two levels chosen were (10,10,15) and (15,15,15). The first level was chosen because it gives the best combination of low error and small number of samples. When using the (10,10,15) sampling level, a Frobenius norm error of only 6.02% is induced when computing only 4.92% of the total points. The second level was chosen because it computes roughly twice as many points, and exhibits roughly one and a half times the Frobenius norm error. This was done to see how much affect the additional error would have upon the final solution. Full results are detailed in Chapter 4.

3.4 Computational/Storage Gains

In practice, the goal is to reduce some combination of the amount of time required to compute the forward matrix, the amount of space required to store that matrix, and the amount of time required to use that matrix. The methods expressed above make significant gains in the first two of those three areas, while additional gains in the third in the case where the sample points are exponentially spaced to improve the interpolation.

Table 3.2: Interpolation Error for a Variety of Interpolation Levels

	Full Dynamic Range (FDR)							
	Log-Sampled Points				Linear Sampled Points			
Sampling Level (n_{d_1}, n_{d_2}, n_z)	Frobenius Norm	One Norm	Infinity Norm	Percent Computed	Frobenius Norm	One Norm	Infinity Norm	Percent Computed
(20,20,20)	3.49%	1.82%	7.93%	18.02%	53.29%	13.40%	120.05%	18.26%
(20,20,15)	3.74%	2.03%	7.93%	14.12%	51.45%	13.15%	112.97%	13.89%
(20,20,10)	5.23%	2.96%	8.48%	10.02%	54.27%	15.88%	112.97%	10.40%
(20,20,5)	11.14%	7.87%	11.58%	5.28%	57.88%	20.68%	120.05%	5.29%
(15,15,20)	8.71%	3.42%	20.04%	11.73%	77.17%	21.39%	175.42%	11.48%
(15,15,15)	9.64%	4.03%	19.20%	9.23%	82.39%	22.59%	175.42%	8.78%
(15,15,10)	10.64%	4.89%	19.20%	6.53%	79.93%	26.14%	175.42%	6.42%
(15,15,5)	12.62%	8.24%	19.20%	3.50%	104.66%	40.68%	175.42%	3.39%
(10,10,20)	5.40%	4.43%	10.93%	6.42%	150.40%	56.44%	237.26%	5.88%
(10,10,15)	6.02%	4.84%	11.29%	4.92%	156.00%	60.19%	223.20%	4.53%
(10,10,10)	6.44%	5.92%	10.93%	3.38%	166.72%	70.70%	237.26%	3.10%
(10,10,5)	11.12%	9.79%	11.29%	1.88%	204.31%	107.44%	237.26%	1.80%
(5,5,20)	74.14%	33.64%	165.72%	2.19%	358.46%	269.07%	305.73%	2.04%
(5,5,15)	75.39%	34.62%	165.72%	1.65%	361.98%	264.92%	312.31%	1.53%
(5,5,10)	79.48%	38.12%	183.79%	1.14%	388.69%	316.81%	312.31%	1.06%
(5,5,5)	113.99%	56.50%	183.79%	0.62%	482.29%	508.41%	305.73%	0.55%
	Logarithmically Compressed Dynamic Range (LCDR)							
	Log-Sampled Points				Linear Sampled Points			
	Frobenius Norm	One Norm	Infinity Norm	Percent Computed	Frobenius Norm	One Norm	Infinity Norm	Percent Computed
(20,20,20)	1.29%	1.10%	1.82%	18.02%	3.81%	2.66%	7.12%	18.26%
(20,20,15)	1.43%	1.25%	1.47%	14.12%	3.02%	2.64%	2.85%	13.89%
(20,20,10)	4.21%	2.71%	9.24%	10.02%	6.85%	4.53%	19.83%	10.40%
(20,20,5)	11.59%	9.17%	11.93%	5.28%	13.10%	10.66%	15.86%	5.29%
(15,15,20)	1.77%	1.44%	2.42%	11.73%	5.93%	4.02%	11.34%	11.48%
(15,15,15)	1.86%	1.66%	2.41%	9.23%	6.27%	4.59%	7.34%	8.78%
(15,15,10)	5.03%	3.33%	8.02%	6.53%	11.23%	7.15%	24.76%	6.42%
(15,15,5)	12.54%	10.02%	13.20%	3.50%	14.57%	12.36%	14.81%	3.39%
(10,10,20)	2.67%	2.19%	5.50%	6.42%	13.92%	9.36%	20.97%	5.88%
(10,10,15)	3.81%	3.04%	5.26%	4.92%	12.39%	9.20%	16.50%	4.53%
(10,10,10)	5.24%	4.20%	10.96%	3.38%	16.62%	12.37%	24.34%	3.10%
(10,10,5)	11.66%	9.90%	11.56%	1.88%	28.05%	22.02%	38.38%	1.80%
(5,5,20)	7.39%	7.44%	12.37%	2.19%	33.52%	30.05%	38.24%	2.04%
(5,5,15)	7.55%	7.87%	9.19%	1.65%	31.51%	29.34%	28.28%	1.53%
(5,5,10)	11.14%	9.89%	29.04%	1.14%	27.63%	26.67%	25.03%	1.06%
(5,5,5)	17.35%	17.02%	16.28%	0.62%	38.50%	30.31%	74.43%	0.55%

SD Configuration	Computational/Storage Improvement
7x7 Raster Scanned	49×
5x5 Sources over 5x5 Detectors 10x10x10 Voxels	690×
10x10 Sources over 10x10 Detectors 10x10x10 Voxels	3844×
25 Random Sources over 25 Random Detectors 10x10x10 Voxels	10×

Table 3.3: Computational/Storage Improvements for a Variety of Source-Detector Configurations

3.4.1 Symmetry Exploitation

For many common source-detector configurations, the level of redundancy eliminated using symmetry exploitation can be quite large. For example, in a system such as that under consideration in this research, where both the source and detectors are raster scanned across the plane, the improvement is roughly equal to the number of source locations that are utilized. For systems with a grid of fixed detectors, an array of source locations, and data collected from every sensor for every source, the effects are even more dramatic. While the redundancies involved are not readily apparent, improvements of several hundred to several thousand times are possible. In the accompanying Table 3.3, a set of 25 sources are spread evenly on a grid above a $10 \times 10 \times 10$ voxel space, yielding an improvement of 690×. When the density of the source and detector grids are doubled, the improvement goes up by a factor of ~ 5.5 to 3844×. Even when the source and detector locations are scattered randomly across the top and bottom surfaces of the slab, an improvement of 10× is still seen.

These results indicate that while this method is highly useful for the particular configuration under examination in this thesis, it is not the configuration which would see the most improvement through the use of this method. The additional redundancy in the spatial relationships between grids of sources and detectors brings along with it a much higher level of redundancy which makes it ideally suited to this technique.

The gains quoted in Table 3.3 apply to both the amount of computation required and the storage space needed. Because the symmetry exploitation only removes redundant calculations, both type of improvement seen are directly proportional to the level of redundancy. Evaluating the Green's function half as frequently not only requires approximately half the time, depending on the exact method by which it is evaluated, but requires exactly half the amount of storage space, because rather than being stored twice, the numbers are only being stored once.

The next question to be answered is whether or not exploitation of symmetry improves the amount of computation required to utilize the matrix $\mathbf{A}$ in matrix-vector and matrix-matrix multiplication (primarily the $\mathbf{A}^T\mathbf{A}$ multiplication that is used in the traditional least squares solution $((\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{g})$.

For matrix-vector multiplications, the equation at each source-detector pair is either:

$$\mathbf{S}_{ij}\mathbf{A}_{sm}\mathbf{g} \tag{3.11}$$

or

$$\mathbf{A}_{sm}^T\mathbf{S}_{ij}^T\mathbf{g} \tag{3.12}$$

depending on whether the total problem is $\mathbf{Ag}$ or $\mathbf{A}^T\mathbf{g}$, respectively.

Looking first at the case of $\mathbf{Ag}$, one can see that $N_{min} * N_T$ multiplications are required to do the first $\mathbf{A}_{sm}\mathbf{g}$ multiplication. The $\mathbf{S}_{ij}\mathbf{A}_{sm}\mathbf{g}$ product is obtained without any further multiplications, as $\mathbf{S}_{ij}$ consists entirely of ones, and thus only selection is occurring. When repeated across N_S sources, with N_D detectors for each source, this gives a total number of multiplications N_{mult} of:

$$N_{mult} = N_S * N_D * N_{min} * N_T \tag{3.13}$$

For the alternate $\mathbf{A}^T\mathbf{g}$ problem, the initial $\mathbf{S}_{ij}^T\mathbf{g}_{data}$ product is obtained through simple selection, and a further $N_{min} * N_T$ multiplications are needed to get to $\mathbf{A}_{sm}^T\mathbf{S}_{ij}^T\mathbf{g}_{data}$. Multiplied by the number of source-detector pairs, this once again gives a total number of multiplications of $N_S * N_D * N_{min} * N_T$.

These two values compare with the number of multiplications required by the full matrix-vector multiplication for either $\mathbf{Ag}$ or $\mathbf{A}^T\mathbf{g}$:

$$N_{mult} = N_S * N_D * N_{xyz} * N_T \tag{3.14}$$

The computational savings that is possible by going from the total forward system to a method utilizing symmetry exploitation is therefore dependant upon the ratio between N_{xyz} and N_{min}. For the system under consideration in this thesis, N_{min} is roughly equal to N_D times N_{xyz}. This is due to a value being required at each voxel for each of the N_D detectors. This means that when looking solely at the computational performance of the matrix-vector product, we pay a price when utilizing this method. This loss is offset, however, by the large gains made in the matrix computation phase.

3.4.2 Interpolation

In examining the computational gains from utilizing interpolation in the generation of $\mathbf{A}_{sm}$, two cases need to be taken into account. The first of these is the case in which the LCDR is utilized. The second case is using the FDR. In both cases, it is presumed that sample points are exponentially spaced, although this has no impact on the computational complexity beyond changing the number of sample points. Looking at the two cases, the first requires the use of a logarithm and then a subsequent exponential. Thus two methods had drastically differing orders of magnitude of computational burden.

In the first case, with the LCDR, the computations required for each source-detector pair are (using Matlab notation):

$$\exp(\mathbf{S}_{ij}\mathbf{Q}_{interp}\ln(\mathbf{V}_{comp}))\mathbf{g}_{ij} \tag{3.15}$$

Because each of the $\mathbf{S}_{ij}$ are simply selection matrices, and a large amount of computation is required for the repeated exponentials, it is more efficient to move the exponential inside the selection. By changing the equation to:

$$\mathbf{S}_{ij}\exp(\mathbf{Q}_{interp}\ln(\mathbf{V}_{comp}))\mathbf{g}_{ij} \tag{3.16}$$

the $\exp(\mathbf{Q}_{interp}\ln(\mathbf{V}_{comp}))$ can be done once, with the results used as $\mathbf{A}_{sm}$ for each source-detector pair. This reduces the equation down to:

$$\mathbf{S}_{ij}\mathbf{A}_{sm}\mathbf{g}_{ij} \tag{3.17}$$

which is identical to the situation without interpolation. What this shows is that for the case when the LCDR is used, there is no computational gain involved aside from what is obtained through the reduced number of matrix values which must be explicitly computed.

In the case where the FDR is used, the situation improves significantly. Without the previously required logarithm and exponential terms, the equation for each source detector pair is:

$$\mathbf{S}_{ij}\mathbf{Q}_{interp}\mathbf{V}_{comp}\mathbf{g}_{ij} \tag{3.18}$$

Both $\mathbf{S}_{ij}$ and $\mathbf{Q}_{interp}$ are highly sparse matrices, and can be multiplied together to form a single sparse matrix with only four nonzero values per row. By doing this multiplication before multiplying by the data vector, the number of multiplications required is reduced. Because $\mathbf{S}_{ij}$ consists entirely of ones, the product of the two sparse matrices requires no multiplications. Selection of the appropriate rows of $\mathbf{Q}_{interp}$ is the only operation that needs to be performed. The resulting equation requires only that the data be projected onto $\mathbf{V}_{comp}$, and that four multiplications be done for each row in the $\mathbf{S}_{ij}\mathbf{Q}_{interp}$ product. Given this situation, $(N_{pt}N_T + 4 * N_{xyz})$ multiplications are now needed, where previously $N_{xyz}N_T$ multiplications were required per source-detector pair. The reduction in required computation will then be one minus the ratio between these two values. Presuming that the $4 * N_{xyz}$ term is small with respect to the total number of computations, the ratio is approximately equal to N_{pt}/N_{xyz}. Thus the improvement in number of multiplications is directly proportional to the amount of interpolation done. If values are explicitly computed for only 10$ of the voxel locations ($N_{pt} = 0.1 * N_{xyz}$), then the ratio will be $0.1/1 = 0.1$, giving a 90% reduction in the number of computations required.

Chapter 4

Results

Over the course of this research, three different data sets were utilized in testing our algorithm. The first two sets were simulated data sets generated using a matched model, both with and without additive noise. A simulated phantom object, roughly ellipsoidal in shape, was created. This object was then used with the first order Born model to generate a set of perturbation data. This noise free data was the first set of data used. Noise was then added to the first set in the form of additive Gaussian white noise to generate the second data set.

The third data set that was used was another simulated data set. This data set was provided to us by Advanced Research Technologies (ART). It consists of a homogeneous background, with a single absorbing phantom of unknown size and location buried within. The data set was generated using their proprietary forward and noise models.

All results were computed using code written in Matlab. This code was run on a dual 1GHz Linux server with 1GB of RAM per processor. When time requirements are explicitly stated, they are with regards to this machine. As a result of using Matlab to execute this method, the code is neither as optimized nor as fast as custom-written code would be. Computation times are given primarily as a method of comparing one method to another. Each set of reconstruction results is presented as a set of Z-slices through the solution volume. Starting with the slice closest to the source in the upper left, and working left to right, top to bottom, each slice is progressively displayed.

The source-detector configuration for all of the data sets consists of a 7×7 grid of source locations, with a grid spacing of 0.5cm. For each source, five detectors were used. One was placed directly beneath the source, and the others were arranged symmetrically a short distance away. For each source-detector pair, a series of 177 data points were collected, each spaced 62.5ps apart, with the first collected at $t = 0$. This gave a total data collection time of approximated 12ns. The 7×7 grid of sources was centered over a $7cm \times 7cm \times 6cm$ volume, which was then voxelized into 2352 voxels, each a cube 0.5cm on a side. This voxelization results in the solution spaced being divided into $14 \times 14 \times 12$ voxels. The background parameters, estimated from the ART simulated set, are $\mu_a = 0.0688$ and $\mu_s = 9.05$. These values are used for the generation of all matrices.

4.1 Matched Models

The first set of data that was used to test the data reuse and interpolation methods was a set of data generated using the full Born forward model. A ellipsoidal phantom was generated (Fig 4.1), and data were generated using the Born forward model for a system with the previously stated background values. The absolute magnitude of the perturbation used in this case does not matter, as it just acts as a global multiplier. Thus for this case it was simply set equal to one.

4.1.1 Noise Free

Initially, the system was run with no noise added, so that a direct comparison of the inversion results with and without interpolation could be made. The resulting data was then inverted using both a Tikhonov regularized least squares with a first order gradient

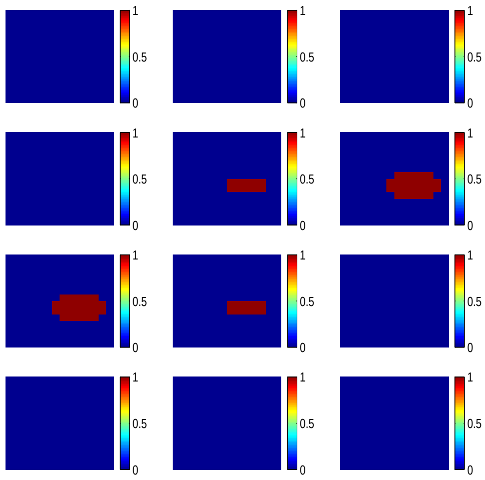

Figure 4.1: Phantom used with matched model for generation of data sets

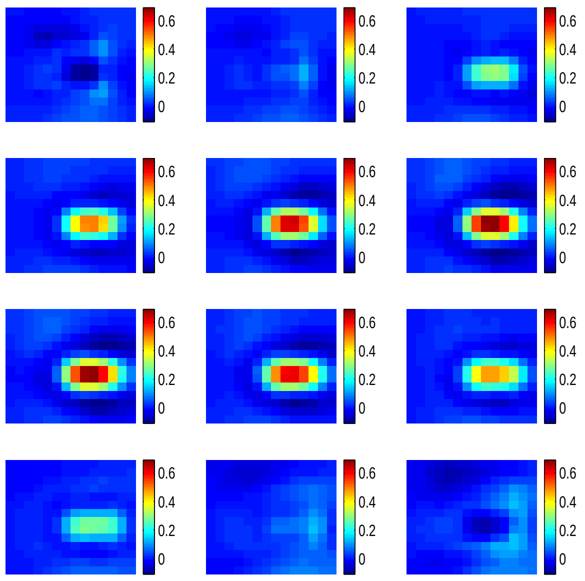

Figure 4.2: Matched Models, No Noise: Using Fully Computed A_{sm} - Solution Utilizing Tikhonov Inversion

A 70.17% 2-norm error is present between this and the actual solution.

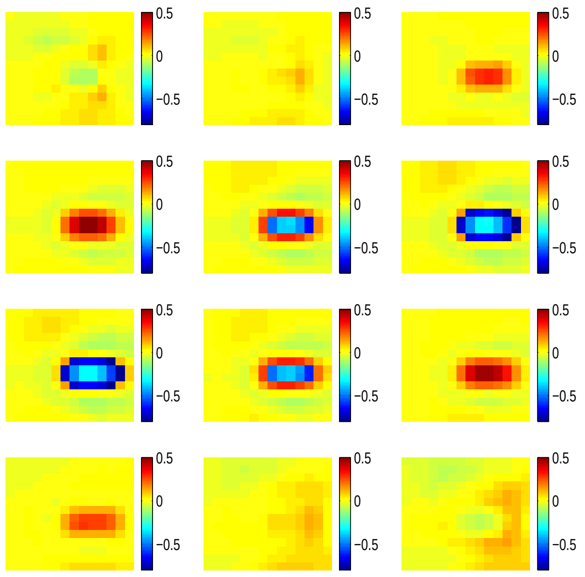

Figure 4.3: Matched Models, No Noise: Using Fully Computed A_{sm} - Difference between actual and computed solutions with Tikhonov Inversion

regularizer, as well as the LSQR algorithm. Due to the idealities involved in such a system (matched forward/inverse model, no noise), regularization is not theoretically needed in order to obtain a stable solution. In any realistic system, however, the regularization would be necessary. Because of this, regularization was used even in the no-noise case, at a level identical to that required by the case where the SNR was set at 10dB. This regularization level was determined through the use of an L-Curve analysis. For details on this method of regularization parameter selection, see [14]. This method was used for all Tikhonov inversion results.

The system was first run using the fully computed $\mathbf{A}_{sm}$. Inversion using Tikhonov regularized least squares resulted in the solution shown in Figure 4.2. To obtain this solution, approximately 80 minutes was required on the aforementioned system. Of this 80 minutes, 60 was spent on the explicit generation of $\mathbf{A}_{sm}$, while the other 20 minutes was spent obtaining a solution. These time values are consistent between the noise free matched, noised matched, and mismatched systems. This is due to the fact that the number of source-detector locations are the same for each system, as well as the voxelizations.

While the percentage 2-norm error in the solution is large, at 70.17%, the object can easily be seen buried in the medium. Looking at the error in Figure 4.3, it can be seen that the large 2-norm error is primarily the result of mismatches in the value of the perturbation. The solution consistently underestimates the magnitude of the perturbation, which leads to a large error in the solution. Even in this ideal situation, the system is poorly conditioned, and as such it is difficult to obtain a highly accurate solution.

The first interpolation level used in generating the matrix was (15,15,15). The first two values correspond to the number of samples along the D_1 and D_2 axes, respectively, while the last value is the number of samples along the Z axis. The D_1 and D_2 axes are sampled using the method detailed in §(3.2.2), while the Z axis is sampled linearly. This sampling method, combined with these sampling levels, resulted in 1086 points being computed, rather than the 11760 that would be required if all were explicitly calculated. This results in a 90.7% reduction in the amount of time required for initial computation.

Rather than the hour previously required, in this case, only 320 seconds are required to compute the necessary forward matrix values. The computation of a solution still required approximately 20 minutes, resulting in a total computation time of $\tilde{2}5$ minutes. The failure of the interpolation to provide an increase in solution computation time is due to the fact that $\mathbf{A}$ shows up in the solution as both $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{g}$. The matrix-matrix multiplication requires the majority of the computation, and cannot be accelerated due to the fact that the sparse matrices flank the dense matrices on either side. Once again, due to the shared geometry between this and the other data sets, the times obtained with the other data sets are nearly identical.

The results of the inversion can be seen in Figure 4.4. This solution appears to be very similar to that obtained with the fully computed matrix. In fact, it is difficult to find any differences in the solutions from only a simple visual observation.

Looking at the results analytically reveals an interesting situation. Comparing the results from the fully computed matrix with that of the interpolated matrix gives a 2-norm error of 11.96%. Seemingly, the two solutions have a significant amount of differences. In fact, this large error stems from a large number of small errors spread across the volume. Looking at Figure 4.5, it seems that the interpolated solution tends to give a higher value to the perturbation, while giving a lower value to the background than the fully computed solution did. The error is clearly not random, and does possesses a significant amount of spatial structure. Whether this structure is inherent in the interpolated matrix or a result of the structure of the perturbation is unknown. If it is inherent to the matrix, the difference between the fully computed and the interpolated solutions should always have error with roughly the same structure. If it is a result of the structure in the perturbation, then the error will likely be structured around the perturbations and background.

Comparing the interpolated solution with the actual perturbation used to generate the data, the 2-norm error is 71.10%, only 0.93% more than the fully computed solution. This result leads one to believe that the error between the fully computed and interpolated solutions may not be the best tool for determining the accuracy of an interpolated solution. Because of the significant amount of error already present in the fully computed solution, it may not be the best reference point for comparison. It would be possible to

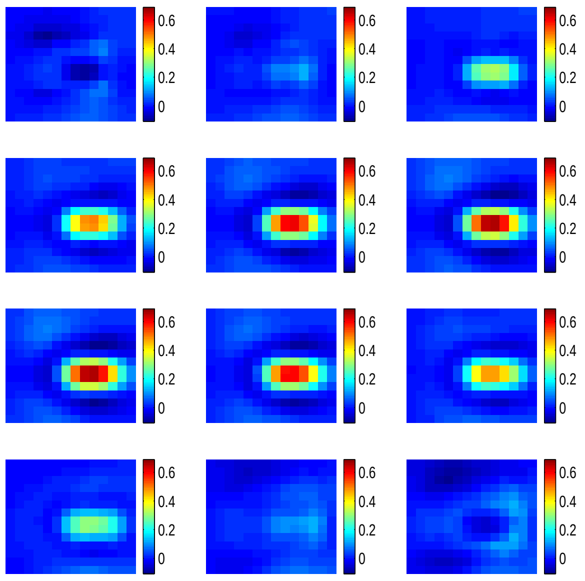

Figure 4.4: Matched Models, No Noise: (15,15,15) Sampling - Tikhonov Inversion

A 11.96% 2-norm error is present between this and the solution using the fully computed A_{sm}. However, error between this and the actual solution is 71.10%, a difference of only 0.93%, making it a 1.3% increase in the error

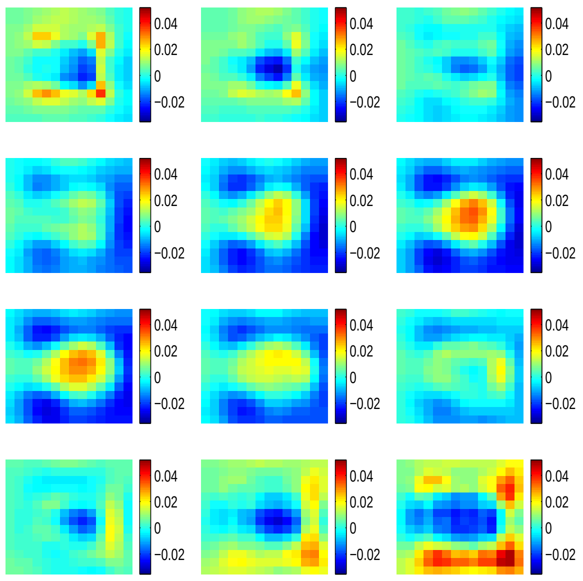

Figure 4.5: Matched Models, No Noise: (15,15,15) Sampling - Tikhonov Inversion Error

find a solution which were significantly better in terms of actual accuracy, but which was significantly different from the fully computed solution. Obtaining a better solution was not a goal of this work, however. The primary objective with regards to absolute error is to ensure that the error is not increasing drastically. This is the case in this situation, with the absolute error increasing less than 1%, even though the error relative to the fully computed solution is 10%.

The next interpolation level shown here is (10,10,15). This level of interpolation resulted in 579 points being explicitly computed instead of 11760. This brought the computation time required to generate $\mathbf{V}_{comp}$ down to only 177 seconds, or about 3 minutes. This is a twenty-fold improvement over the originally required one hour. As before, no improvement is seen when using Tikhonov inversion due to the demands of computing $\mathbf{A}^T\mathbf{A}$ and inverting it.

This time, there is a 12.47% 2-norm error between the fully computed and interpolated results. However, the error between the interpolated results and absolute truth is 70.35%, just 0.18% greater than the error in the fully computed solution. This result is actually superior in absolute accuracy than the (15,15,15) solution which computed nearly twice as many sample points. This result was somewhat expected, however, as when $\mathbf{A}_{sm}$ was generated using the (10,10,15) level, all of the error metrics were approximately equal to, or significantly less than, those when using the (15,15,15) sampling level. The unexpected part of this result is that while the absolute error is less than with the (15,15,15) sampling level, the difference between this solution and the fully computed one is actually greater than before. One would expect that the difference between the two would be less with this sampling level, for the same reasons that were stated as reasons to expect a superior absolute error. This differentiation between the error in $\mathbf{A}_{sm}$, and the error induced in the solution is an area of potential future work.

The noise free data were next inverted using LSQR. Initially, the system was run using the fully computed $\mathbf{A}_{sm}$ to give a baseline result for comparison. The results of this inversion can be seen in Figure 4.8. Once the hour had been spent to obtain $\mathbf{A}_{sm}$, 1505 seconds were required to execute 100 iterations of the algorithm, giving an average iteration time of 15 seconds. Because no suitable conversion criterion was found, the

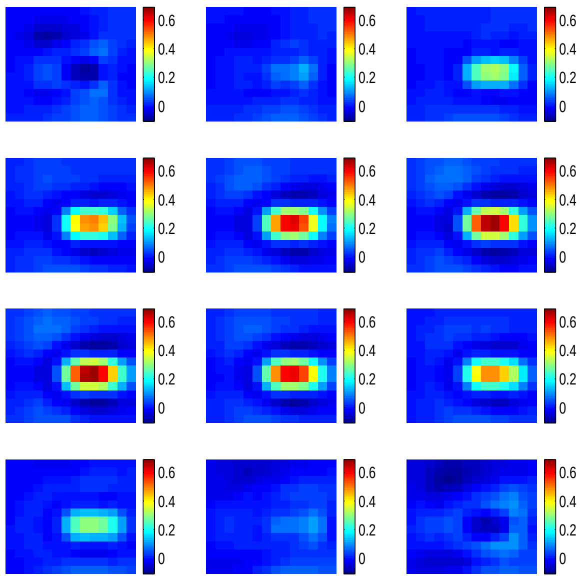

Figure 4.6: Matched Models, No Noise: (10,10,15) Sampling - Tikhonov Inversion

A 12.47% 2-norm error is present between this and the solution using the fully computed A_{sm}. However, error between this and the actual solution is 70.35%, a difference of only 0.28%, making it a 0.4% increase in the error

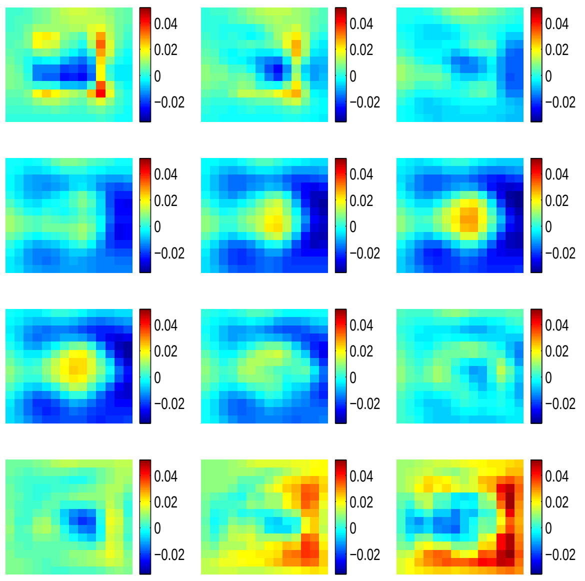

Figure 4.7: Matched Models, No Noise: (10,10,15) Sampling - Tikhonov Inversion Error

number of iterations at which to truncate the solution was determined through comparison with the fully computed solution. This error was minimized after 33 iterations had been completed. At 10.2 seconds per iteration, this yields a total inversion time of 5.6 minutes. This compares favorably with the 20 minutes required to evaluate a Tikhonov solution.

The results in Figure 4.8 are similar to those from Figure 4.2. The perturbation is once again clearly visible. However, there is now additional noise present in the solution that was not there before. The background is not as smooth as with the Tikhonov solution, and that is reflected in the 2-norm error of 77.67% when compared to the actual solution. This is more than 7.5% greater than was obtained with the Tikhonov inversion. This error could be reduced through the use of additional iterations of the algorithm, given that there is no noise present in this data, and the data is matched to the model.

The error between this solution and the actual perturbation is shown in Figure 4.9. As with the other fully computed solution, the error present is primarily a result of the algorithm poorly determining the value of the perturbation. While values are underestimated at the actual location of the perturbation, voxels which are part of the background rather than the perturbation are given nonzero values. The sum of these numerous small errors results in a large overall 2-norm error.

Computing an LSQR solution using the (15,15,15) interpolation level provides similar results. This time, 1072 seconds were required to evaluated 100 iterations of the algorithm. This yields only 10.7 seconds per iteration, roughly a 30% decrease in computation time. This is clearly not the optimal improvement, which should be nearly a 90% decrease. It is clear from this that the code is not fully optimized to make use of these gains.

Comparing the (15,15,15) solution to the fully computed one, there is a 11.64% difference in 2-norm. The absolute error present is 78.80%, an increase of 1.04% over the fully computed solution.

Moving to a (10,10,15) sampling level reduces the time required to evaluate 100 iterations from 1069 second to 1018 seconds. This reduces the average iteration time to

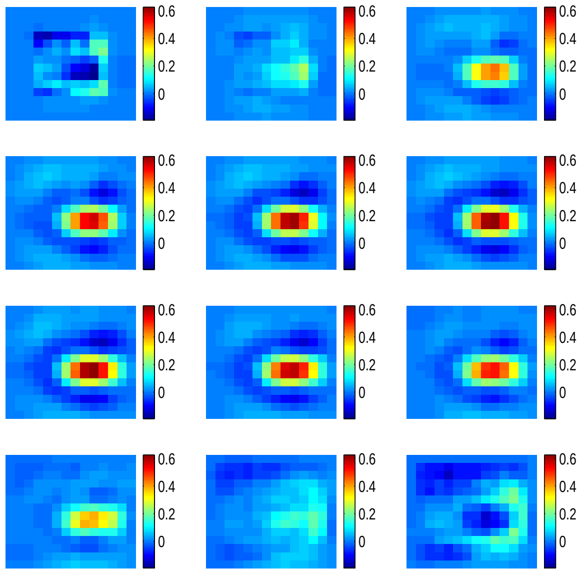

Figure 4.8: Matched Models, No Noise: Full Forward Matrix - LSQR Inversion

A 77.76% error is present between this solution and the actual perturbation.

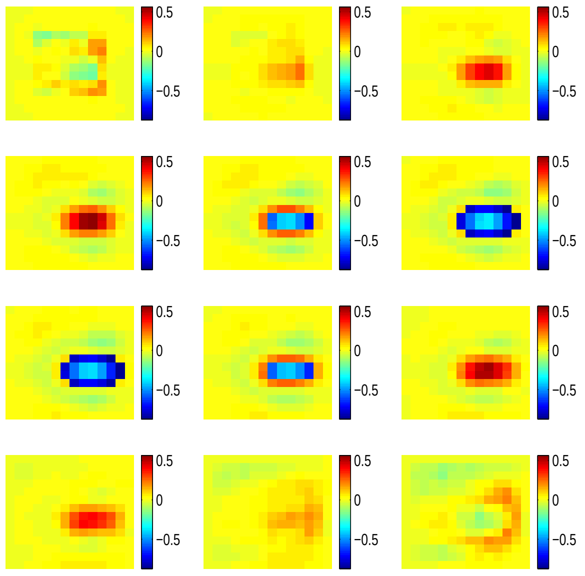

Figure 4.9: Matched Models, No Noise: Full Forward Matrix - LSQR Inversion

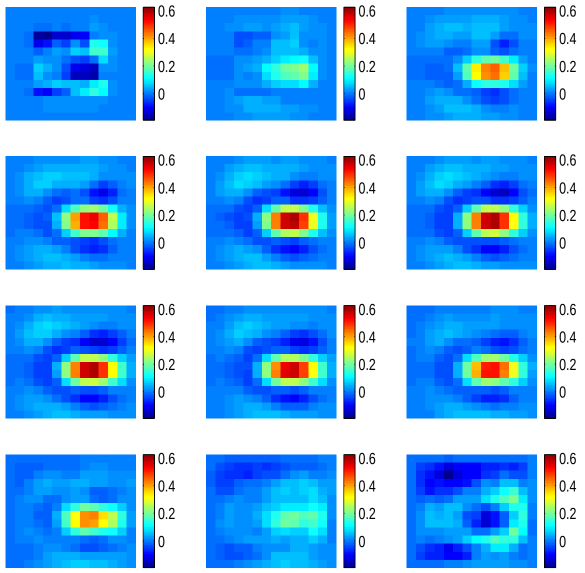

Figure 4.10: Matched Models, No Noise: (15,15,15) Sampling - LSQR Inversion

A 78.80% error is present between this solution and the fully computed one.

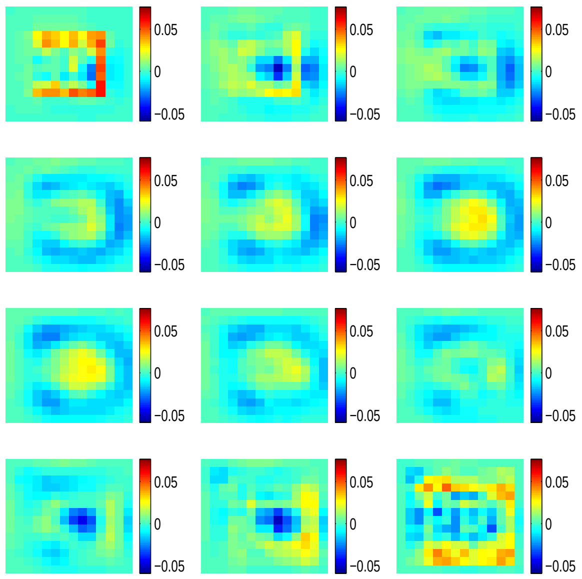

Figure 4.11: Matched Models, No Noise: (15,15,15) Sampling - LSQR Inversion Error

10.2 seconds. Once again, the gain seen here is not as much as would be expected. A gain of approximately 5% is made over the (15,15,15) sampling level, where theory would expect a gain of nearly 100%. It may simply be that the cost of the overhead involved in each iteration is taking up a large part of the computation on each iteration, and that the gains involved in the actual matrix vector computation are becoming less relevant.

The solution obtained with the (10,10,15) sampling level is shown in Figure 4.12, while the difference between it and the fully computed solution is shown in Figure 4.13. The difference present between the fully computed solution and this solution is 14.29%, while the error with regards to the actual perturbation is 78.05%. Once again, the lower sampling level actually provides results which are superior to those obtained with the higher sampling level. Additionally, this is despite having a larger difference when compared to the fully computed solution. In both this case, and the case with a (15,15,15) sampling level, the difference between these solutions and the fully computed one seems to share similar structure with the error present in the Tikhonov solutions.

4.1.2 Additive Noise, SNR = 10dB

Once an ideal system had been run (matched model, no noise), additive white Gaussian noise was added to bring the signal to noise ratio up to 10dB. This was done to see what effect, if any, the noise would have on the accuracy of the interpolated solution. Noise was added to the signal using the Matlab function *awgn()* to measure the signal strength and add the appropriate amount of noise.

Results inverting the noisy data using the fully computed $\mathbf{A}_{sm}$ and Tikhonov inversion can be seen in Figure 4.14. Even with a reasonably significant amount of noise present in the signal, Tikhonov inversion is able to easily localize the perturbation. The perturbation is once again identified very well in the X and Y dimensions, with the object appearing quite compact. However, resolution is again lost along the Z-axis, with the object being spread heavily across six Z-slices, with a noticeable presence in an additional two. This contrasts with the four layers in which the perturbation actually exists. Comparing this result to the actual answer, a 75.62% 2-norm error is present, compared to the 70.17% error which was present in the noise free result.

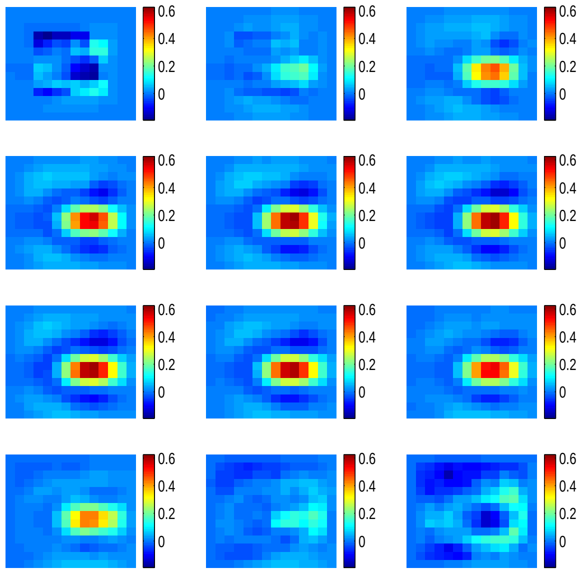

Figure 4.12: Matched Models, No Noise: (10,10,15) Sampling - LSQR Inversion

A 78.05% error is present between this solution and the fully computed one.

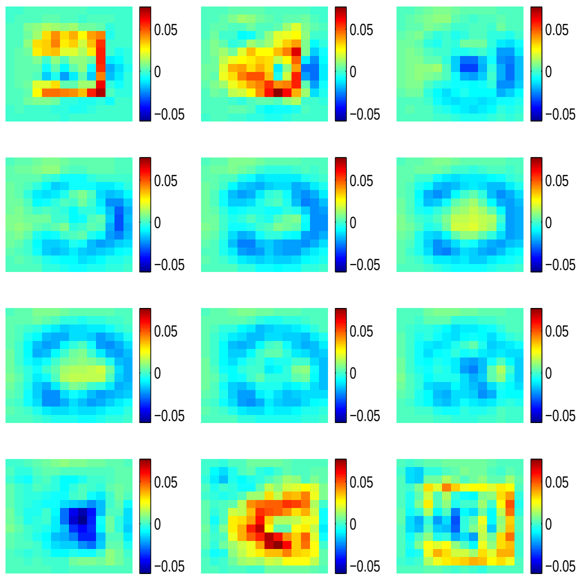

Figure 4.13: Matched Models, No Noise: (10,10,15) Sampling - LSQR Inversion Error

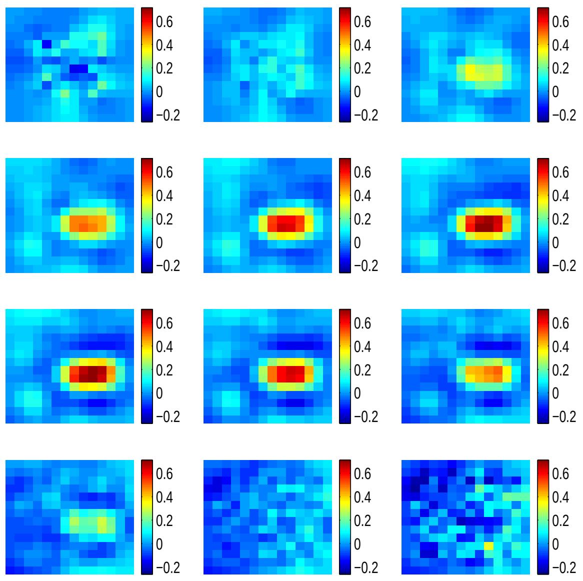

Figure 4.14: Matched Models, SNR 10dB: Using Fully Computed A_{sm} - Tikhonov Inversion

A 75.62% 2-norm error is present between this and the actual solution.

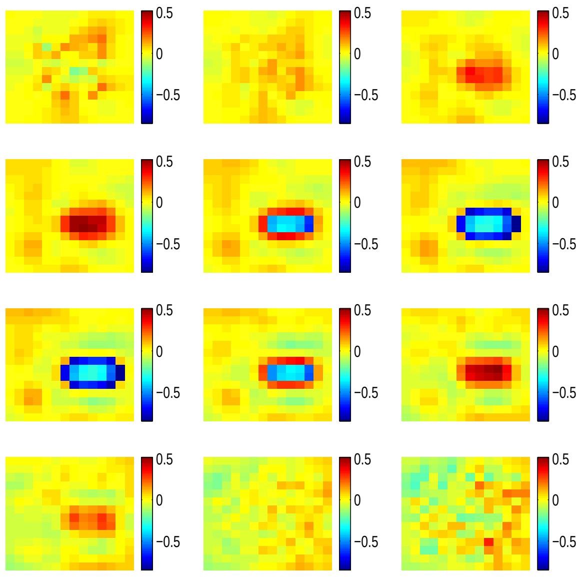

Figure 4.15: Matched Models, SNR 10dB: Using Fully Computed A_{sm} - Difference between computed solution and actual perturbation

Looking at the results using the (15,15,15) interpolation level, a similar situation is present as with the noise free data. While there is more noise in the result, this is expected because the data is no longer ideal. Comparing the result with that obtained from the fully computed $\mathbf{A}_{sm}$ gives a 14.45% 2-norm error. This compares with a 81.62% 2-norm error between the interpolated solution using noisy data and the actual perturbation phantom. Once again, error between the fully computed and interpolated solutions does not seem to be an indicator of comparatively large error between the interpolated solution and the actual perturbation values. Also, it does not appear as though the error present in the interpolated version of $\mathbf{A}_{sm}$ is acting to amplify the noise present in the signal.

Examining the difference between the interpolated solution and the fully computed solution, we again see error which is not completely random. In this case, the difference seems to be concentrated in the areas where a perturbation was determined to lie. Outside of these areas, there seems to be a rather consistent difference between the two solutions. Again, however, the magnitude of these differences is quite small in comparison to the total value of the solution. With the additional noise in the system, however, more noise has begun to appear in the layers nearest the sources and detectors.

Solving the system using (10,10,15) interpolation provides results similar to those previously obtained with error free data. This time, there is an 13.13% 2-norm error between the interpolated solution and the reference solution using the fully computed $\mathbf{A}_{sm}$. This contrasts with a 76.25% 2-norm error between the interpolated solution and the actual answer. In this case, the results obtained from the (10,10,15) sampling level are actually closer to the fully computed solution than those obtained with the (15,15,15) sampling level. However, as before, the actual error is less when using the (10,10,15) level.

The noisy data was then inverted using LSQR as the inversion routine. The times required to compute the solution were comparable with those required for the noise free data. Results using the fully computed solution are shown in Figure 4.20, with the error in the solution being shown in Figure 4.21. The absolute error present is 81.43%, which is an increase of about 6% over the solution using Tikhonov inversion.

Using a (15,15,15) sampling level results in an absolute error of 81.62%, with a difference of 14.45% between the interpolated and fully computed solutions. These results

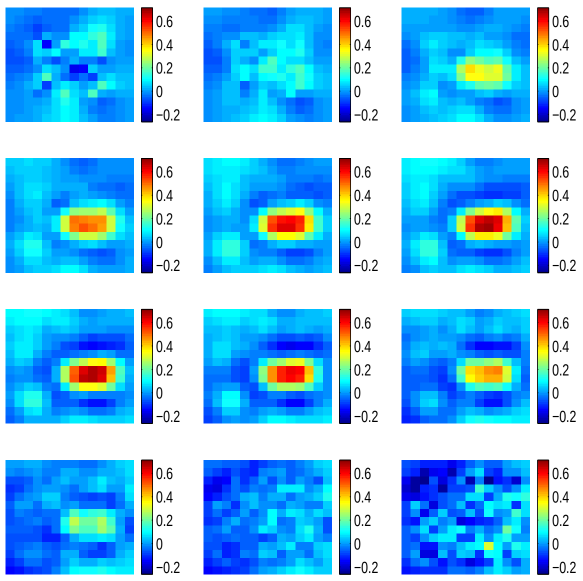

Figure 4.16: Matched Models, SNR 10dB: (15,15,15) Sampling - Tikhonov Inversion

A 13.32% 2-norm error is present between this and the solution using the fully computed A_{sm}. However, error between this and the actual solution is 76.57%, a difference of 0.95%

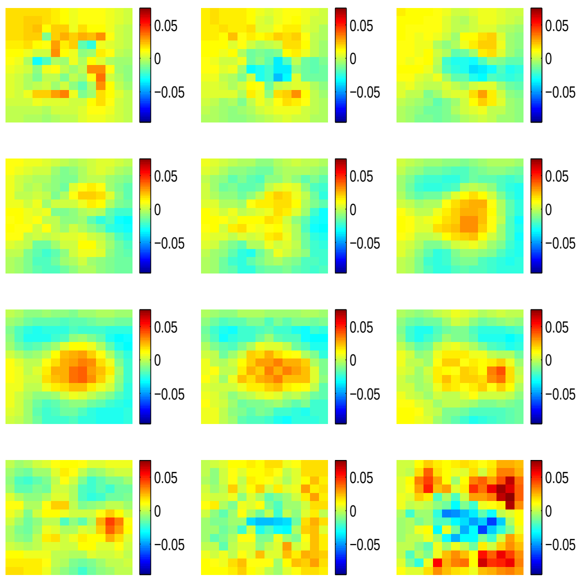

Figure 4.17: Matched Models, SNR 10dB: (15,15,15) Sampling - Tikhonov Inversion Error

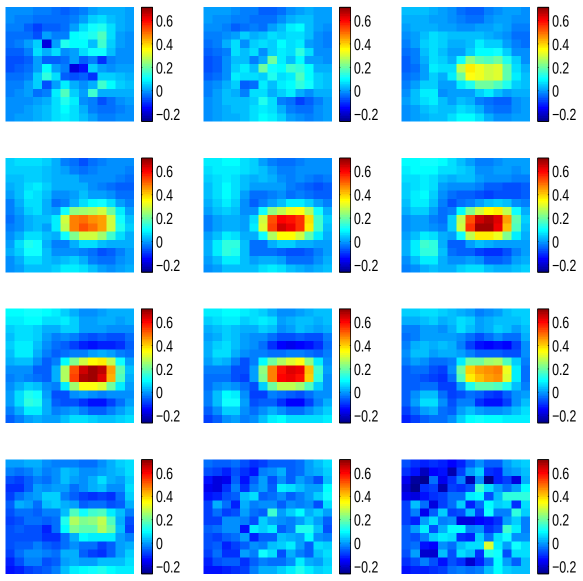

Figure 4.18: Matched Models, SNR 10dB: (10,10,15) Sampling - Tikhonov Inversion

The solution here has a 13.13% 2-norm error when compared to the fully computed solution. Comparing against the actual value results in only a 76.25% error, a 0.62% increase over the fully computed solution.

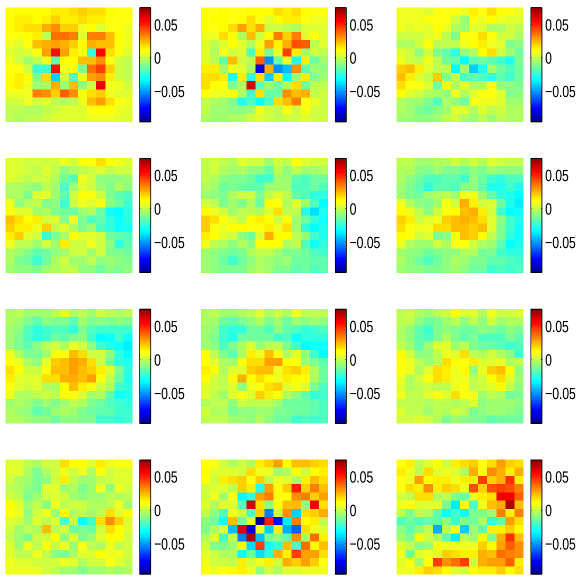

Figure 4.19: Matched Models, SNR 10dB: (10,10,15) Sampling - LS Inversion Error

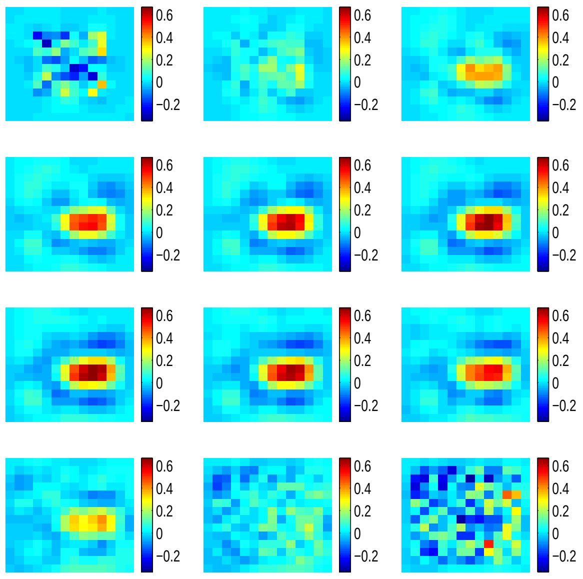
Figure 4.20: Matched Models Full Forward Matrix - LSQR Inversion

A 2-norm error of 81.43% is present in this solution.

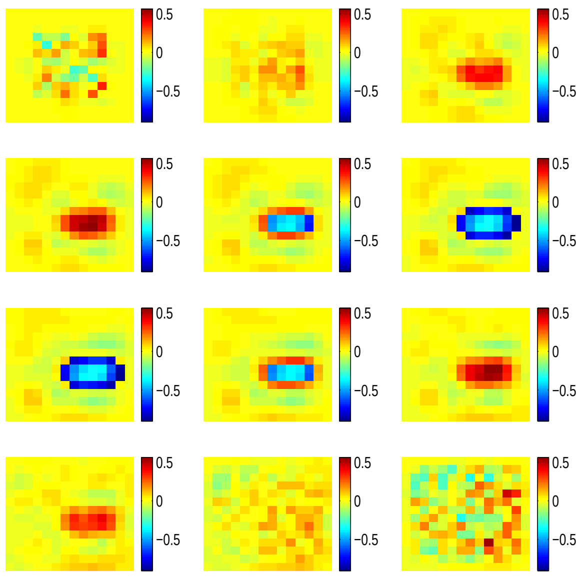
Figure 4.21: Matched Models Full Forward Matrix - LSQR Inversion Error

can be see in Figures 4.22 and 4.23. Once again, the error has a similar structure as that obtained with the fully computed $\mathbf{A}_{sm}$. However, there is an additional amount of noise which enters into the slices closest the sources and detectors.

Using a (10,10,15) sampling level results in a similar solution, with the solution shown in Fig 4.24 and the difference between this solution and the fully computed one shown in Fig 4.25. This time, the absolute error is 81.65%, while the difference between fully computed and interpolated solutions is 14.45%. The error had the same structure which has been seen numerous times before, although the magnitude of the additional noise here is significantly higher than has been seen before.

4.1.3 Mismatched Models

Further simulated results were obtained using a simulated data set generated using a different (unknown) forward model, and an unknown noise model. The system setup for this data set utilizes the same source-detector configuration as the other data sets. Voxelization is also identical to that used for the matched data sets.

Figure 4.26 shows the results of applying Tikhonov regularized least squares to the fully computed forward matrix. The solution shows a single absorbing object buried in the medium. This object appears to be roughly spherical in shape. In addition to the object, a large number of inhomogeneities are clustered near each boundary of the slab. Anecdotal evidence indicates that this is a common occurrence with such systems, and could possibly be the result of mismatched background parameters. Another possibility that has been suggested is that setting the voxel size such that the sources are exactly one voxel apart may lead to these errors.

The first of the two interpolation levels presented here for the mismatched model is with a sampling rate of (15,15,15) as described earlier. The results of the Tikhonov inversion can be seen in Figure 4.27. Visual analysis of the results seems to indicate that the results are very close to those obtained utilizing the full matrix. When an analytical approach is taken, the resulting errors are significantly higher than would be expected from simple visual inspection. For this case, the 2-norm error between the interpolated and fully computed results is 11.06%. While the absolute truth is not known in this

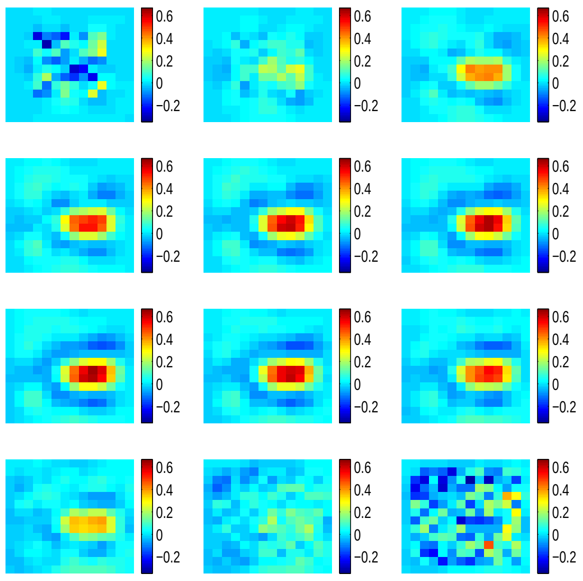

Figure 4.22: Matched Models (15,15,15) Sampling - LSQR Inversion

An absolute 2-norm error of 81.62% is present in this solution. The difference between this solution and the fully computed solution is 14.45%.

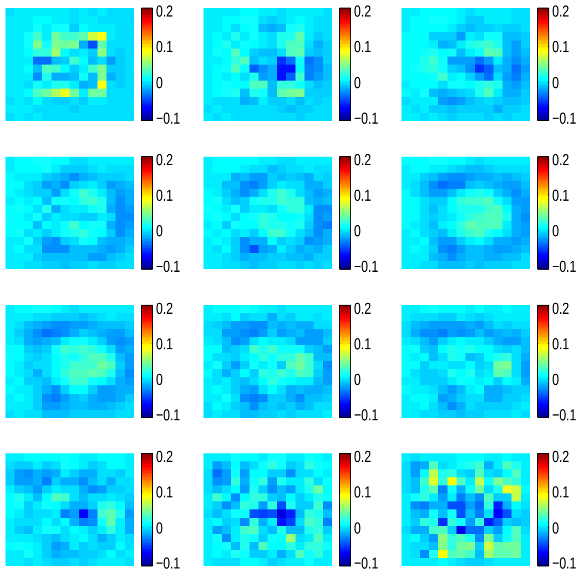

Figure 4.23: Matched Models (15,15,15) Sampling - LSQR Inversion Error

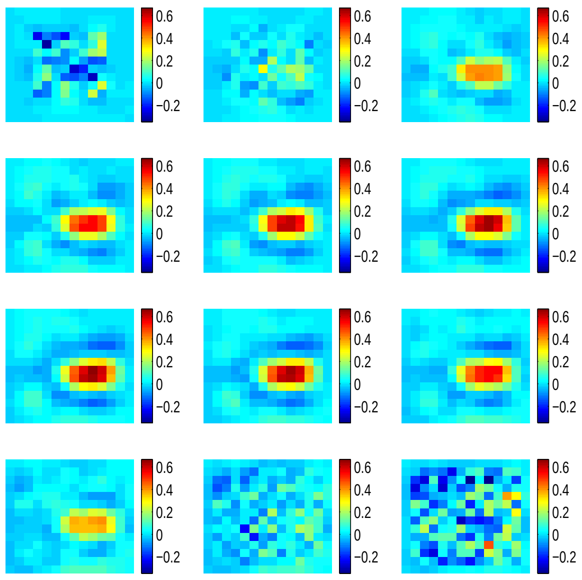

Figure 4.24: Matched Models (10,10,15) Sampling - LSQR Inversion

An absolute 2-norm error of 81.65% is present in this solution. The difference between this solution and the fully computed solution is 19.55%.

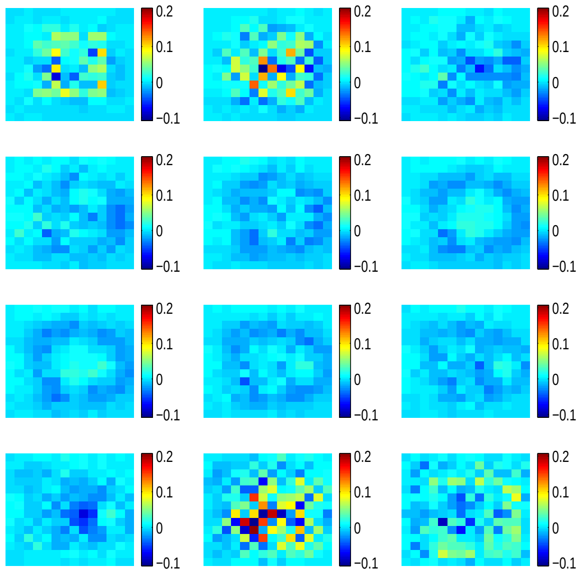

Figure 4.25: Matched Models (10,10,15) Sampling - LSQR Inversion Error

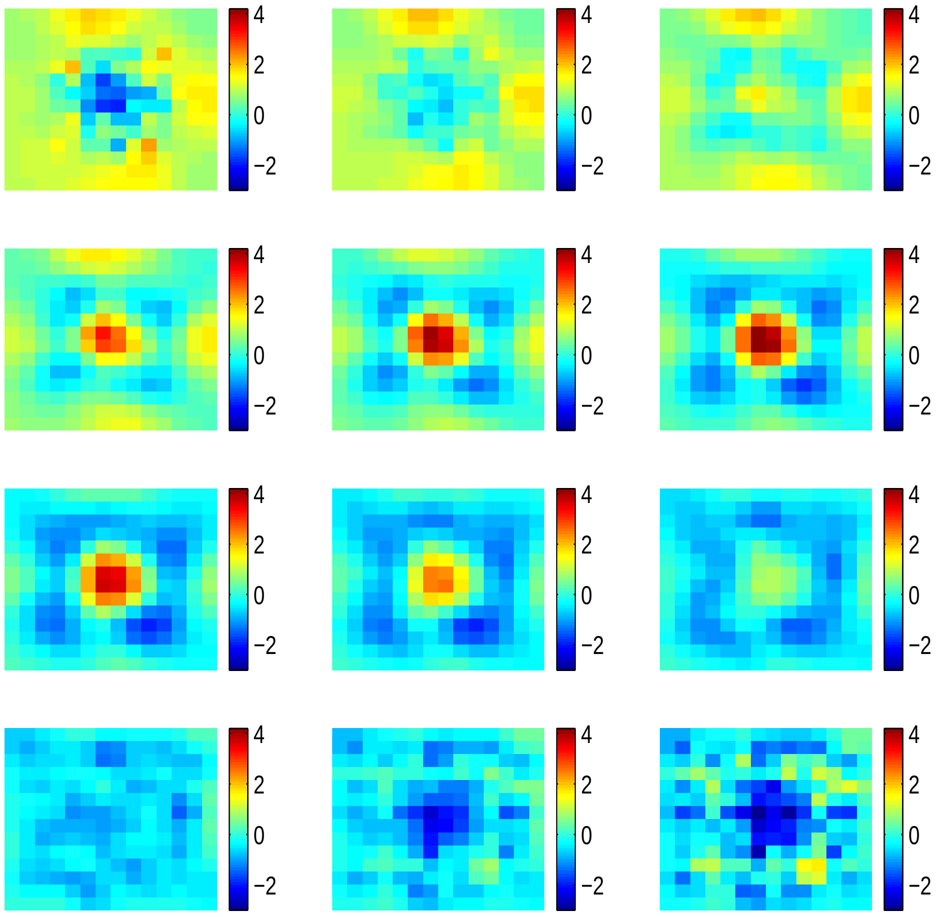

Figure 4.26: Mismatched Models Fully Computed A_{sm}- Tikhonov Inversion

case, it was previous shown that the differences between solutions obtained with the fully computed $\mathbf{A}_{sm}$ and solutions from the interpolated version is not necessarily a good indicator of absolute error in the interpolated version.

Looking at the difference between the two results (Fully computed and interpolated), plotted in Figure 4.28, it can be seen that the error tends to fall in one of several distinct areas. This time, the perturbation holds a negative error, while the area immediately surrounding it has a positive error. Continuing outwards, the error drops to nearly zero. It should be noted that the scale on the plot of the error is different than that of the solution to increase the contrast. If both were to be plotted on the same scale, the error would appear solely as small perturbations to the background, without any readily evident structure. Because the error is spread across the entire volume, a small error in each voxel value becomes a significant total error when multiplied by the total number of voxels. One possibility for this occurrence is that the interpolation has somehow stripped a type of DC offset out of the solution, resulting in this widespread background error.

The second interpolation level presented here uses a sampling level of (10,10,15). Inversion results for this interpolation level, utilizing Tikhonov inversion, can be seen in Figure 4.29.

As with the matched model, the (10,10,15) interpolation level provides results which are as good or better than those provided by the (15,15,15) level. Once again, there is a significant amount of difference between this result and the one obtained using the fully computed A_{sm}. This time the 2-norm error is 14.86%, but as previous stated, this likely has little relation to the actual error present in this solution.

Looking at the difference between the (10,10,15) solution and the fully computed solution (Figure 4.30), we see a picture similar to those previously generated. While the error has structure, is it not exclusively confined to a specific area. Instead it is spread across the majority of the background, with different magnitudes depending on location. This results in a large number of small errors. This again leads to the large 2-norm difference between the interpolated and fully computed solutions.

Next, the data set was run using the fully computed A_{sm} and LSQR as the inversion routine. The goal here was again to improve upon the time required to obtain

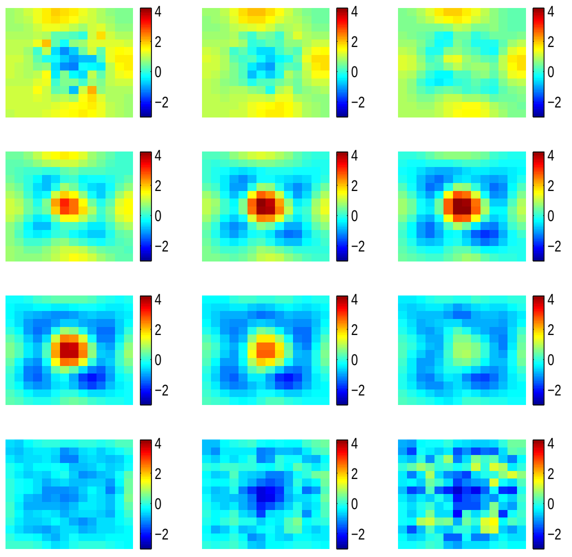

Figure 4.27: Mismatched Models (15,15,15) Sampling - Tikhonov Inversion

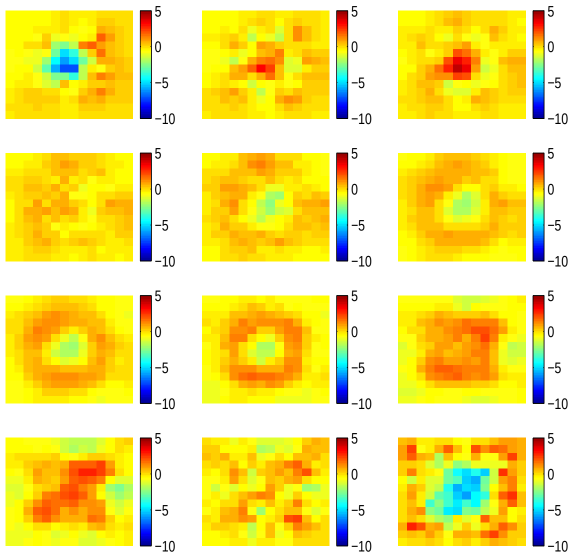

Figure 4.28: Mismatched Models (15,15,15) Sampling - Tikhonov Inversion Error

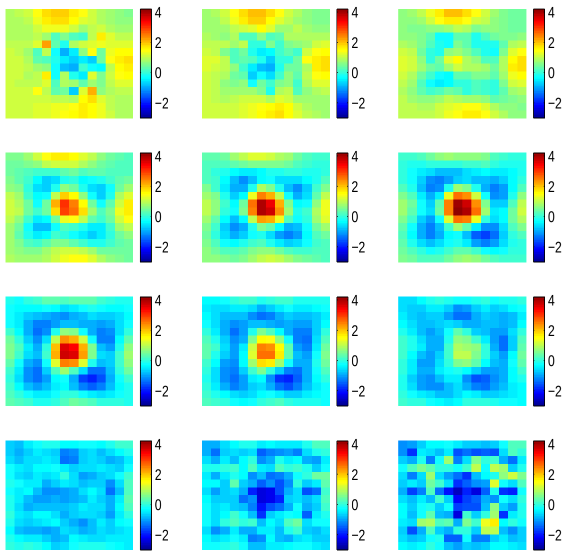

Figure 4.29: Mismatched Models (10,10,15) Sampling - Tikhonov Inversion

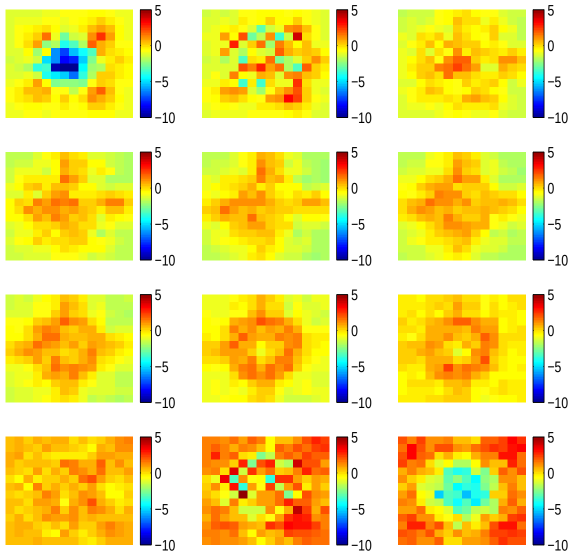

Figure 4.30: Mismatched Models (10,10,15) Sampling - Tikhonov Inversion Error

a solution, without significantly sacrificing accuracy. Using the fully computed matrix, and truncating the iteration at 28 steps resulted in the solution seen in Figure 4.31. This number of iterations was chosen because it gave the closest match to the solution using the fully computed Tikhonov method. This was determine through examination of the error for a large number of iteration values, and choosing the one with the best performance. As with the Tikhonov inverted results, a buried absorbing object, roughly spherical in shape, can clearly be seen. Ideally, the solution would be almost identical to the solution obtained with Tikhonov inversion, as both are ultimately solving the same minimization problem, just through different methods. The primary difference between the two methods is that while Tikhonov inversion required 20 minutes to compute $\mathbf{A}^T\mathbf{A}$ and then do Gaussian elimination on it to invert, LSQR required only 420 seconds (15 seconds per iteration,

Using (15,15,15) interpolation and LSQR resulted in the image in Figure 4.32. The result shown is after 28 iterations of the algorithm. An object is easily identified from visual inspection of the solution, and a 2-norm difference between it and the fully computed solution of 16.28% is present. This result is similar to the 11.06% error that was present in the (15,15,15) solution using Tikhonov inversion.

Moving to a (10,10,15) inversion level results in the solution seen in Figure 4.34. Here, a 2-norm error of 22.17% is present between the interpolated and fully computed results. While this is a significant increase from the 14.86% which was seen in the Tikhonov solution, the object is still clearly visible

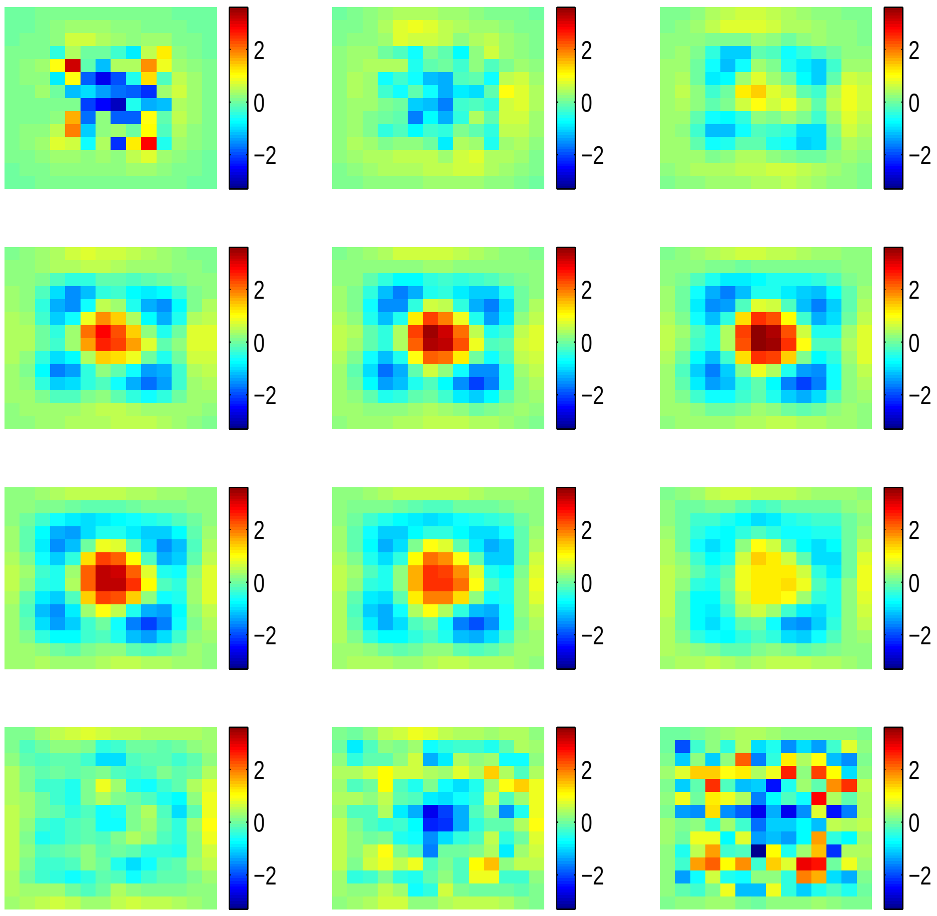

Figure 4.31: Mismatched Models Full Forward Matrix - LSQR Inversion

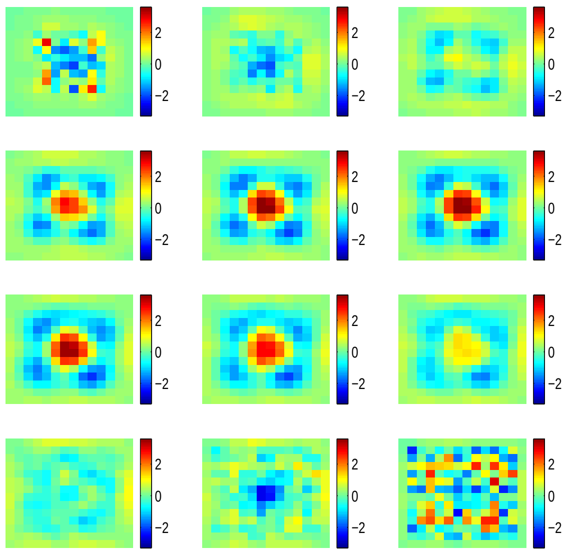

Figure 4.32: Mismatched Models (15,15,15) Sampling - LSQR Inversion

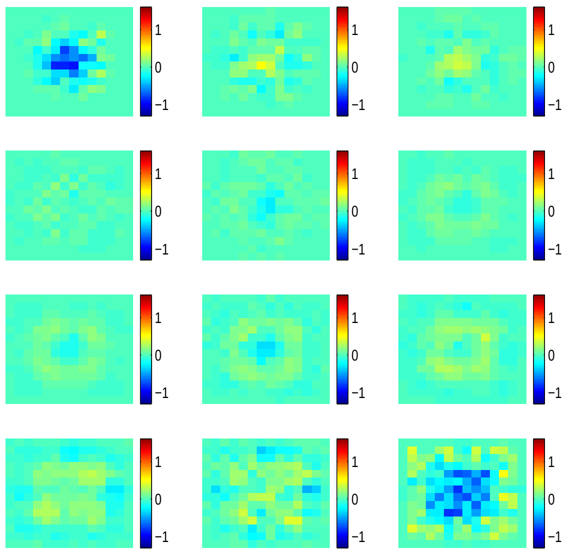

Figure 4.33: Mismatched Models (15,15,15) Sampling - LSQR Inversion Error

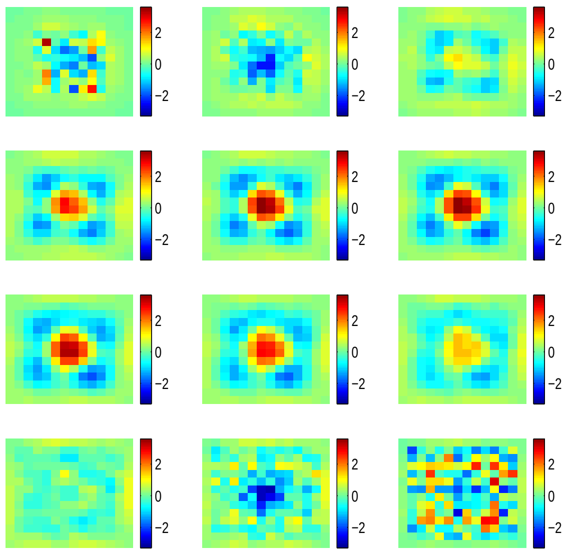

Figure 4.34: Mismatched Models (10,10,15) Sampling - LSQR Inversion

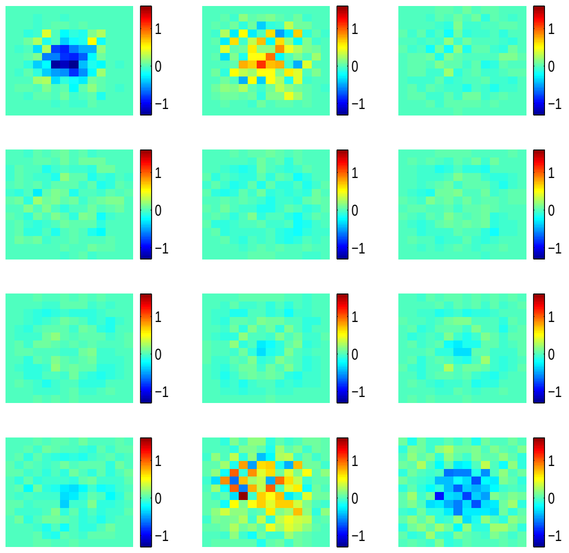

Figure 4.35: Mismatched Models (10,10,15) Sampling - LSQR Inversion Error

Chapter 5

Conclusions and Future Work

We have presented here a method by which first order Born matrices can be rapidly generated for time domain diffuse optical tomography. Such problems take a large number of time dependent amplitude measurements, and attempt to reconstruct the values of the spatially varying absorption and scattering parameters. These parameters can be used to measure underlying physiological parameters such as blood density and oxygenation. In turn, these can be traced to clinically relevant phenomena such as tumors and brain activity.

The first component of this method involves the elimination of repeated calculations done during generation of the full forward matrix. This method was shown to reduce both the time and storage requirements involved with such large matrices, and allows for the use of such matrices in a much more practical manner. The methodology outlined here could also potentially be extended to other problems beyond the scope of just time domain DOT.

The use of interpolation to further reduce both the amount of computation required both in generating and using the forward matrix was also discussed. Results were given using a basic linear interpolation method which showed that the solution seems to be reasonably tolerant of small levels of error in the forward matrix. Qualitatively the solutions seem very much alike, both with and without the interpolation. Quantitatively, the error is only increased by a few percent, in exchange for a large reduction in computational complexity.

There are several areas in which future work could directly extend what has been reported in this thesis. As was shown in the results section, the number of samples taken is not directly related to the accuracy of the resulting interpolation. This may result from sample points falling closer or farther away from those points which are ultimately required. A further analysis of what is causing this phenomenon, and how to best exploit it, would be highly useful. At this point, the choice of sampling densities and locations is rather arbitrary, and based more on something that is aesthetically pleasing rather than mathematically optimal. It would be useful to develop a method which could take as its input either the desired number of points, or the desired computational reduction, and return an optimally selected set of sample points.

Another area in which further research could be performed is an error analysis. The use of interpolation introduces noise into the forward matrix, and through that, introduces error into the solution. By studying and characterizing the noise that is introduced, it may be possible to compensate for it in some other manner. Simple examination of the error introduced to the solution shows that there is some structure present in the noise. If this structure can be determined and exploited, it may be possible to reduce or eliminate its effects.

Bibliography

[1] Simon R. Arridge, Jeremy C. Hebden, Martin Schweiger, Florian E.W. Schmidt, Martin E Fry, Elizabeth M.C Hillman, Hamid Dehghani, and David T. Delpy, "A method for three-dimensional time-resolved optical tomography," International Journal for Imaging Systems and Technology, vol. 11, pp. 2–11, 2000.

[2] Simon R Arridge, "The theoretical basis for the determination of optical pathlengths in tissue," Phys med Bio, , no. 37, pp. 1531–1560, 1992.

[3] David A. Boas, Dana H. Brooks, Eric L. Miller, Charles A. DiMarzio, Misha Kilmer, Richard J. Gaudette, and Quan Zhang, "Imaging the body with diffuse optical tomography," IEEE Signal Processing Magazine, pp. 57–75, November 2001.

[4] John C Schotland, "Continuous-wave diffusion imaging," J. Opt. Soc. Am. A, vol. 14, no. 1, pp. 275–279, 1997.

[5] Greg Boverman, "Modeling and nonlinear inversion for frequency domain diffuse optical tomography," M.S. thesis, Northeastern University, Boston MA, June 2003.

[6] D. A. Boas, M. A. O'Leary, B. Chance, and A. A. Yodh, "Scattering of diffuse photon density waves by spherical inhomogeneities within turbid meda: Analytic solution and applications.," Proc. Natl. Acad. Sci. USA, vol. 91, pp. 4887–4891, May 1994.

[7] Vadim A Markel and John C Schotland, "Inverse problem in optical diffusion tomography. i. fourier-laplace inversion formulas," J. Opt. Soc. Am. A., vol. 18, no. 6, pp. 1336–1347, 2001.

[8] Michael S Patterson, B Chance, and B.C. Wilson, "Time resolved reflectance and transmittance for the non-invasive measurement of tissue optical properties," Applied Optics, vol. 28, no. 12, June 1989.

[9] Simon R Arridge, "Photon-measurement density function. part 1: Analytical forms," Applied Optics, , no. 34, pp. 7395–7409, 1995.

[10] S Chandrasekhar, Radiative Transfer, Dover, New York, NY, 1960.

[11] Carslaw H S and Jaeger J C, Conduction of Head in Solids, Oxford, Clarendon, 2nd edition, 1986.

[12] Richard C. Haskell, Lars O. Svaasand, Tsong-Tseh Tsay, Ti-Chen Feng, Matthew S McAdams, and Bruce J Tromberg, "Boundary conditions for the diffusion equation in radiative transfer," J. Opt. Soc. Am. A, vol. 11, no. 10, pp. 2727–2741, October 1994.

[13] Simon R Arridge, "Optical tomography in medical imaging," Inverse Problems, , no. 15, pp. R41–R93, 1999.

[14] Per Christian Hansen, Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion, chapter 5, pp. 99–106, Siam Monographs on Mathematical Modeling and Computation. Society for Industrial & Applied Mathematics, 1998.

[15] Christopher Paige and Michael Saunders, "Lsqr: An algorithm for sparse linear equations and sparse least squares," ACM Transactions on Mathematical Software, vol. 8, no. 1, pp. 43–71, 1982.

[16] Gregory Boverman, Eric L. Miller, and David. A. Boas, "Linear and nonlinear reconstruction for diffuse optical tomography in an inhomogeneous background," Unknown, 2004.

[17] Jeremy C. Hebden and Simon R Arridge, "Imaging through the scattering media by use of an analytical model of perturbation amplitudes in the time domain," Applied Optics, , no. 35, pp. 6788–6796, 1996.