To the Graduate Council:

I am submitting herewith a dissertation written by Fernando R. Rannou entitled "Iterative Reconstruction of Three-Dimensional Magnetic Resonance Images from Boron Data." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Jens Gregor

Jens Gregor, Major Professor

We have read this dissertation
and recommend its acceptance:

Michael G Thomason

Michel W Berry

M. A. Abidi

Accepted for the Council:

Lewis Linkel

Associate Vice Chancellor and
Dean of The Graduate School

Iterative Reconstruction of Three-Dimensional Magnetic Resonance Images from Boron data

A Dissertation

Presented for the

Doctor of Philosophy Degree

The University of Tennessee, Knoxville

Fernando Rodrigo Rannou

May 1998

Copyright © by Fernando R. Rannou, 1998
All rights reserved

To Jenny, Andrés, and ... Alex too.

Acknowledgments

During these long years of graduate school at Knoxville, Tennessee, I have received the constant support and encouragement from my family and friends. To all of them, I am truly thankful.

I would like to express my highest gratitude to my advisor Jens Gregor, who supported me in all aspects of this work. His assistance and patience kept me right on track. Appreciation is also extended to the other members of my committee, Mongi A. Abidi, Michael W. Berry, and Michael G. Thomason.

Gratitude is also offered to the Departamento de Ingeniería en Informática at Universidad de Santiago de Chile and the Presidente de la Republica fellowship for giving me the opportunity to continue my education and for providing financial support.

I would also like to thank the Department of Computer Science at The University of Tennessee, Knoxville for providing me with support during my studies.

Special thanks go to Srta. Dora Holzer whose loving guidance has been so helpful during these years.

Finally, I want to thank my wife Jenny and my son Andrés, sources of happiness in my life.

Abstract

This dissertation develops a computational framework for iterative reconstruction of three-dimensional (3D) Magnetic Resonance Images (MRI) from boron data. The application is Boron Neutron Capture Therapy (BNCT) which is a promising technique for cancer treatment and MRI is used to monitor the various phases of the therapy. Traditionally, boron MRI is carried out by sampling the object's Fourier spectrum along spherical trajectories and reconstructing the image with interpolation onto a 3D Cartesian grid followed by Fourier inversion. However, this approach does not allow the incorporation of prior spatial knowledge into the reconstruction computation. We therefore propose to reconstruct boron images using iterative algorithms to avoid interpolation in Fourier space as well as to allow for the introduction of spatial constraints.

The spherical data sampling algorithm is modified such that true Radon projection data is obtained and geometrical symmetries are induced. The image is modeled as a linear combination of Kaiser-Bessel basis functions whose 3D Radon transforms are derived. Iterative 3D reconstruction algorithms have vast memory and CPU time requirements. We address this problem by distributing the computation across a network of workstations and by making use of the aforementioned symmetries. Based on the theory of support functions, we propose a 3D focus-of-attention preprocessing technique to further reduce the computational demands. An experimental study based on a noise-free phantom is provided to assess the quality and computational performance of two specific iterative solvers that produce least-squares and maximum-likelihood solutions, respectively. These algorithms are compared with and without focus-of-attention data preprocessing and for different Kaiser-Bessel window parameters. The experiments indicate that the least-squares algorithm converges faster to the true image than the maximum-likelihood algorithm. Finally, it is shown that the combined use of symmetries and focus-of-attention allows us to carry out the 3D image reconstruction using a very small network of workstations.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Boron neutron capture therapy 2

1.3 Objectives 4

1.4 Overview 5

2 Magnetic Resonance Imaging **7**

2.1 Basic physics 7

2.1.1 The free induction decay signal 7

2.1.2 Relaxation factors 8

2.1.3 Spin echo: recovering the signal 10

2.2 Image formation 10

2.2.1 k space 12

2.2.2 Two-dimensional Cartesian sampling 12

2.2.3 Three-dimensional Cartesian sampling 15

2.3 Boron imaging 17

2.4 Image reconstruction 19

2.4.1 Fourier reconstruction 20

2.4.2 Projection reconstruction 20

3 Image Basis Representation **24**

3.1 Introduction and overview 24

3.2 Problem formulation 25
3.3 Basis functions 26
3.3.1 General properties 26
3.3.2 Effectively band-limited windows 28
3.3.3 Spherically symmetric windows 29
3.4 The Kaiser-Bessel basis function 31
3.4.1 Blob definition and properties 31
3.4.2 The three dimensional Radon transform 32
3.5 Blob parameter selection 33

4 Iterative Reconstruction from Boron Data **37**
4.1 Introduction 37
4.2 Obtaining projection data 38
4.2.1 Spherical sampling 38
4.2.2 Modifications to the spherical sampling 40
4.2.3 The readout signal 42
4.2.4 Projection data 43
4.3 The discrete formulation 44
4.4 Geometrical symmetries 45
4.5 Iterative algorithms 49
4.5.1 Least squares 49
4.5.2 Maximum likelihood 51
4.6 Distributed computation 54
4.6.1 Forward and backward projections 55
4.6.2 Load balancing 56

5 Three-Dimensional Focus of Attention **59**
5.1 Introduction and overview 59
5.2 Support functions 61

5.2.1 Support functions and support hyperplanes 61
5.2.2 Consistency conditions 62
5.2.3 Support samples: the discrete case 64
5.3 Application in 3D constrained reconstruction 67
5.3.1 Formulation of the problem 67
5.3.2 Finding local families 68
5.3.3 Optimization 73
5.4 Computational impact 80
5.4.1 On memory 81
5.4.2 On computation and communication 82

6 Experimental Results **84**
6.1 Method 84
6.1.1 Computing environment 84
6.1.2 Data material 84
6.2 Image quality 88
6.2.1 Reconstructions 88
6.2.2 Error measures 94
6.2.3 Computational performance 98
6.3 Summary 100

7 Conclusion **102**
7.1 System description 102
7.2 Contributions and discussion 105
7.3 Future work 107

Bibliography **109**

Appendix **114**

Vita **119**

List of Tables

2.1 Approximate relaxation time values of hydrogen protons for various tissues at a magnetic field of 1.0 Telsa (From [41]). 9

2.2 Boron relaxation times (From [9]). 17

3.1 Optimum values for R and α to approximate a constant function. 36

4.1 Number of nonzero entries and memory requirements in gigabytes for various blob radius with and without symmetries. 49

5.1 Statistics for algorithm *Find_Local_family* with 24 processors using symmetries. 72

5.2 Computational requirements for a generic reconstruction problem. 81

5.3 FOA based memory requirements for two different objects. 82

5.4 Computation and communication timings using FOA 83

6.1 Projection geometry parameters. 85

6.2 Overall mean wall-clock time per iteration for RL-ML. 100

List of Figures

1.1 Boron-neutron reaction. 3

2.1 Schematic representation of magnetic fields in MRI. (a) $\mathbf{M_0}$ is the net magnetization due to a static magnetic filed $\mathbf{B_0}$. (b) Rotating frame of reference. $\mathbf{M_0}$ is flipped onto the transverse plane by a second, oscillating magnetic field $\mathbf{B_1}$ perpendicular to $\mathbf{B_0}$. 8

2.2 Spin echo rephasing and formation of the spin echo signal. (a) All spins are in phase and the FID is maximum. (b) After a time t_1 some dephasing has occurred. (c) The application of a 180° $\mathbf{B_1}$ pulse reverses the phase so that $\mathbf{M_0}$ points in the opposite direction. (d) After a time t_1 spins regain phase coherence and a spin-echo signal is produced. 11

2.3 Pulse sequence for spin echo. 11

2.4 Timing diagram of a spin-echo pulse sequence designed for 2D MRI. 13

2.5 Timing diagram of a spin-echo pulse sequence designed for 3D MRI. 16

2.6 Spherical sampling of the Fourier spectrum. The number of rays and their orientations are chosen to cover the spectrum evenly. 18

2.7 Pulse sequence for projection reconstruction in boron imaging. After the RF pulse is applied $G_x(t)$, $G_y(t)$, and $G_z(t)$ are turned on and data sampling begins. 19

2.8 Relation between the k-space and object space. A line through the origin of the Fourier space (a) corresponds to a planar projection in object space (b). 22

3.1 The 3D projection reconstruction geometry. 26
3.2 Illustration of the shifting property of the Radon transform in 2D. 27
3.3 Illustration of the independence on $\mathbf{r}$ of the Radon transform for spherically symmetric functions. Vector $\mathbf{x} \in \mathbb{R}^3$ is orthogonal to $\mathbf{r}$ in (a) and the vector $\mathbf{y} \in \mathbb{R}^2$ in (b). 30
3.4 The Kaiser-Bessel window basis function for $\alpha = 6, 10$, and 20. $K = 2$ and $R = 1.5$ for all curves. 32
3.5 Computing the 3D Radon transform. The filled area is the plane of integration, s is the distance from it to the origin, and R is the blob radius. . . 33
3.6 3D Radon transform of blobs for $\alpha = 6.0, 10.0$ and 20.0 and fixed radius $R = 1.5$. 34
3.7 Contour graph of the logarithm of the error of approximating a constant function with blobs. 35

4.1 Distribution of projections in k-space. (a) side view, (b) top view. 40
4.2 Simplified pulse sequence and projection readout gradient for boron imaging. 42
4.3 Diagram of geometrical symmetries. (a) Rotation symmetry for the half-space $z > 0$ and (b) as seen from the z-axis. (c) Mirroring symmetry. . . . 46
4.4 Schematic representation of the memory savings obtained for matrix $\mathbf{A}$ based on rotation and mirroring operations. 47
4.5 Sparse matrix data representation. 48
4.6 The LSQR algorithm. 52
4.7 The RL-ML algorithm. 53
4.8 Forward and backward projection procedures for a symmetry based system matrix. 55

4.9 Description of the dependency of the area on θ: (a) area of intersection between image and all projection planes at $\mathbf{r}_i$, (b) plot of the area as a function of θ. 57

4.10 Wall-clock time needed by a backward operation for two workload distributions: (a) raster, (b) heuristic. 58

5.1 Example of an object and its support function. (a) object and support geometry, (b) support function. 62

5.2 Geometrical interpretation of the determinantal condition in 3D: (a) condition is satisfied, (b) condition is not satisfied. (Adapted from [19].) 64

5.3 Local family example. In (a) $\mathcal{L}_4 = \{\text{cone}^+\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}, \text{cone}^+\{\mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}, \text{cone}^+\{\mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}\}$. The introduction of a new vector in (b) changes the family to $\mathcal{L}_4 = \{\text{cone}^+\{\mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}, \text{cone}^+\{\mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}\}$. (Adapted from [19].) 66

5.4 The initial estimate of the support vector is computed with a moving-average based thresholding over each projection. 67

5.5 Procedure *Find_Local_Family*. 70

5.6 Illustration of uniqueness search for $\mathbf{u}_k$. The dashed arrows show the alternating search for vectors with same elevation angles. The solid lines show the order in which elevation angles are searched. The black dots correspond to the starting search vectors at each elevation. 71

5.7 Ellipsoid and ideal support function. 78

5.8 Ellipsoid support vectors. (a) Consistent, (b) inconsistent due to added Gaussian noise. 78

5.9 Error measures for linearized Jacobi algorithm. (a) Root mean square error between ideal and estimated support vector. (b) Minimum test value. . . . 79

5.10 Estimate support vectors after 100 iterations of linearized Jacobi with $\gamma =$ 0.0015: (a) ideal estimate, (b) quantized estimate. 80

6.1 Simulated coefficient phantom used in the experiments. (a) Slice $z = 31$ and (b) corresponding central profile $z = 31$, $y = 31$. 86

6.2 Region-of-interest (ROI) specifications. 86

6.3 Simulated phantom images and central profiles. Images correspond to $z = 31$ and profiles correspond to $z = 31$, $y = 31$. For (a) and (b) $R = 1.25$, $\alpha = 3.6$, (c) and (d) $R = 1.5$, $\alpha = 6.4$, and (d) and (e) $R = 2.0$, $\alpha = 10.4$. . 87

6.4 Reconstructed images obtained by the RL-ML and LSQR algorithms with and without FOA with $R = 1.25$, $\alpha = 3.6$. 89

6.5 Reconstructed images obtained by the RL-ML and LSQR algorithms with and without FOA with $R = 1.5$, $\alpha = 6.4$. 90

6.6 Reconstructed images obtained by the RL-ML and LSQR algorithms with and without FOA with $R = 2.0$, $\alpha = 10.4$. 91

6.7 Central profiles obtained by the RL-ML and LSQR algorithms with FOA with $R = 1.25$ and $\alpha = 3.6$. For (a) $z = 31, y = 31$, and (b) $z = 31, x = 31$. . 92

6.8 Central profiles obtained by the RL-ML and LSQR algorithms with FOA with $R = 2.0$ and $\alpha = 10.4$. For (a) $z = 31, y = 31$, and (b) $z = 31, x = 31$. . 93

6.9 Plots of the normalized error between the true and reconstructed images by the RL-ML and LSQR. For (a) $R = 1.25$, $\alpha = 3.6$, (b) $R = 2.0$, $\alpha = 10.4$. 95

6.10 Mean plus-minus one standard deviation results obtained with the RL-ML algorithm for each of the six ROI's. Markers $\bigcirc$ and $\times$ indicates results with and without FOA, respectively. 96

6.11 Mean plus-minus one standard deviation results obtained with the LSQR algorithm for each of the six ROI's. Markers $\bigcirc$ and $\times$ indicates results with and without FOA, respectively. 97

6.12 Mean wall-clock time per iteration for RL-ML and LSQR with and without FOA with $R = 2.0$, $\alpha = 10.4$. (a) and (b) RL-ML timings, (c) and (d) LSQR timings. 99

6.13 Mean wall-clock time per iteration for RL-ML with FOA with $R = 2.0$, $\alpha = 10.4$, for 4, 8, and 12 processors. 100

7.1 Schematic system description for iterative reconstruction of 3D images from boron data. 103

A.1 Computing the three dimensional Radon transform. The filled area is the plane of integration; s is the distance from it to the origin and R is the blob radius. 116

Chapter 1

Introduction

1.1 Motivation

In this research, we address sampling and computational aspects pertaining to a three-dimensional (3D) image reconstruction algorithm for use in Boron Neutron Capture Therapy (BNCT) which is a potential technique for cancer treatment [26, 30, 17]. A patient is first injected with a boron compound designed to specifically be absorbed by tumor cells and then irradiated with thermal neutrons. Ideally, the tumor is destroyed thereby while the surrounding tissue is left intact. Magnetic Resonance Imaging (MRI) [7, 6, 28, 38, 2] may be used as a tool for determining the tissue-selectivity of a particular boron compound and for patient monitoring. MRI is based on the magnetic properties of certain nuclei, e.g. hydrogen which is often referred to as spin or a proton. When placed in a strong magnetic field, these nuclei can be made to resonate thereby generating a signal which can be recorded and used to reconstruct the object. By modifying the magnetic field with gradients, it is possible to encode spatial information with frequency. Thus, different points in the object will resonate at a different frequency depending on their position with respect to the gradients.

Due to very short relaxation times of the boron MR signal, traditional proton sampling techniques are inadequate and a spherical scheme must be employed instead [9]. The

standard approach for reconstructing 3D images sampled this way is to use filtered back-projection or, alternatively, to interpolate onto a Cartesian grid and subsequently apply direct Fourier-inversion [9]. However, neither of these two approaches allows prior spatial information to be incorporated into the reconstruction computation. We therefore take a different approach, namely, formulate the image formation process as a linear system of equations and develop a computational framework based on iterative reconstruction algorithms. The main disadvantage of such an approach is that it imposes extensive memory requirements for storing the system matrix and significant CPU time usage for lengthy iterations. In order to alleviate the space/time drawback, other authors [20, 5] have made use of supercomputers. This solution is not economically viable. We tackle the problem by distributing the reconstruction task across a small network of workstations and by making use of symmetries to reduce memory requirements. We also apply a Focus-Of-Attention preprocessing technique [14, 13] to further improve computational efficiency.

Focus-Of-Attention is based on the theory of support functions [19] and can be seen as a basic form of spatial prior to constrain the reconstruction. But more importantly, it is technique that reduces the dimensionality of the problem substantially making iterative algorithms a feasible approach. It has been successfully applied to 2D image reconstruction where the theory [33] and its implementation [14, 13] are greatly simplified. However, its application to three and more dimensions has been precluded by the large amount of computational resources involved [19]. By specifically targeting the theory to our 3D reconstruction problem we are able to exploit symmetries in a distributed environment and produce a feasible Focus-Of-Attention scheme.

1.2 Boron neutron capture therapy

In BNCT a stable isotope of boron, e.g. B-10, is administered to the patient and given time to accumulate preferentially in the tumor. The tumor is then irradiated with a beam of neutrons which are captured by the boron. A fission process results in which the B-10

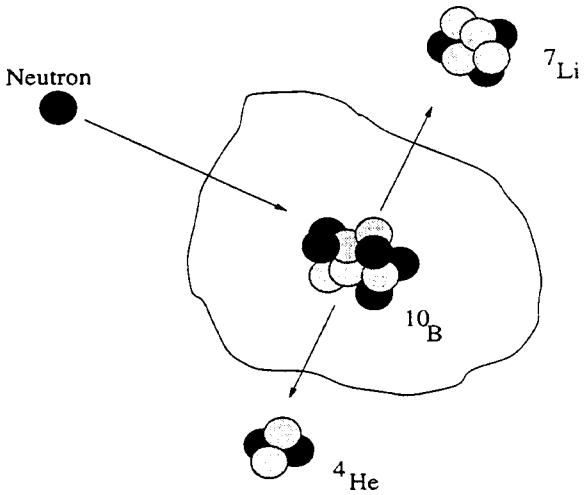


Figure 1.1: Boron-neutron reaction.

splits releasing an alpha particle and a lithium nucleus:

$$B^{10} + \text{Neutron}^{1} \longrightarrow He^{4} + Li^{7} + \text{Heat}$$

Figure 1.1 shows a typical boron-neutron reaction.

These highly-energetic charged particles are short in range which means that they deposit their energy in a volume of radius of one or two cells. Thus, ideally only tumor cells containing boron are destroyed and healthy tissue is left intact.

The main components of the therapy are [39]:

1. Boron compound selection. Since the compound is administered into the blood stream for prolonged periods and in high concentrations, the compound must be nontoxic.

2. Selective assimilation of compound in tumor cells. Ideally, the compound should be absorbed only by tumor cells such that upon neutron radiation they are the only cells destroyed. In practice, however there is always a probability of damaging normal cells. Research is being carried out to develop compounds that minimize this probability.

3. Boron compound activation with neutrons. The energy of the neutrons must be adequate for penetration to the depth of the tumor: low-energy (thermal neutrons) for shallow tumors and high-energy (epithermal neutrons) for deep tumors.

MRI has been used to monitor the BNCT process [16, 9, 17]. In principle, MRI can measure the tumor's location, size, and depth and thus help customize the therapy to a specific patient. After the compound has been injected, an MRI image can reveal compound distribution in the tumor and help decide whether the treatment will be effective. Finally, it can non-invasively show the results of the treatment.

1.3 Objectives

The principal objective of this dissertation is *to provide a computational vehicle for iterative reconstruction of 3D boron MRI images.* In particular, this will be accomplished by:

1. Modifying a spherical sampling algorithm for boron MRI such that (a) true Radon projection data can be sampled and (b) geometrical symmetries can be introduced and subsequently exploited.

2. Modeling the image formation process using spherical symmetric basis functions to facilitate the handling of the above mentioned symmetries. This calls for the three-dimensional Radon transform of the Kaiser-Bessel window function to be derived.

3. Studying the feasibility and performance of distributed, iterative algorithms for boron image reconstruction.

4. Applying the theory of support functions and Focus-Of-Attention to (a) introduce a basic form of prior knowledge into the reconstruction and (b) reduce the computational requirements of the system.

We point out that there is general interest in the development of fully 3D reconstruction algorithms in the field of medical imaging. One aspect considered to be of paramount

importance for most applications is the handling of the vast amounts of data involved [12]. Thus, although the research described in this proposal is focused on MRI and the BNCT application, some of the results may eventually applied in other contexts such general 3D Positron Emission Tomography (PET).

1.4 Overview

The dissertation is organized as follows. The basic principles of MRI are outlined in Chapter 2. Elementary concepts of magnetic fields, magnetic resonance and the MRI signal are reviewed. Then, the fundamental ideas of k-space and image formation prepare the reader for the problem of boron imaging and image reconstruction. The objective of this chapter is to motivate the necessity of spherical sampling and projection reconstruction algorithms for use in BNCT, which in turn allows us to propose iterative algorithms in boron imaging.

Chapter 3 models the problem from a series representation point of view. The image and its Radon transform are assumed to be linear combinations of basis functions, whose desired properties for image reconstruction are stated. In particular, the Kaiser-Bessel window function, a three-parameter spherically symmetric window, is adopted for boron imaging and its 3D Radon transform is derived. Finally, an existing method for selecting optimum Kaiser-Bessel window parameters for smooth images is outlined.

Chapter 4 formulates the image reconstruction problem as solving a system of linear equations. A spherical sampling procedure for boron imaging is modified in order to: (1) obtain Radon projection data and (2) induce symmetries in the sampling geometry that substantially reduce storage requirements of the system matrix. Two well known iterative algorithms are proposed for solving the system of equations. The chapter ends with a study of distributed computation issues associated with the 3D reconstruction algorithms.

Chapter 5 extends the Focus-Of-Attention preprocessing technique to the case of 3D image reconstruction for a spherical projection geometry. The chapter begins with a

description of the basic components of the Focus-Of-Attention system for support constrained reconstruction. The theory of support functions is briefly reviewed and then applied to create the 3D preprocessing scheme. In this chapter, special emphasis is put on computational aspects and distributed computation issues that make the theory applicable to 3D image reconstruction. The chapter closes with an experimental study to evaluate the computational benefits of using Focus-Of-Attention in iterative algorithms.

Chapter 6 presents the experimental results based on simulated phantom data. The experiments consist of a qualitative and a computational study of 3D image reconstruction. The objective of the qualitative study is to asses the quality of reconstructed images modeled with Kaiser-Bessel functions. The two iterative algorithms from Chapter 4 are compared. The computational study evaluates the overall performance of the algorithms in a distributed environment.

Chapter 7 concludes the dissertation with a summary and a discussion of the main results of this work. Finally, future areas of research are outlined.

Chapter 2

Magnetic Resonance Imaging

2.1 Basic physics

This section is based on [7, 6, 28, 38, 2].

2.1.1 The free induction decay signal

MRI is based on the magnetic properties of nuclei. When placed in a static magnetic field $\mathbf{B_0}$, nuclei that have an angular momentum will align either parallel or antiparallel therewith and produce a net magnetization $\mathbf{M_0}$ as shown in Figure 2.1 (a). Since most nuclei align parallel with the field (lower energy level), $\mathbf{M_0}$ points in the same direction as $\mathbf{B_0}$. The nuclei also rotate, or precess, around the axis of the field at a rate called the Larmor frequency given by

$$\omega_0 = \gamma B_0$$

where γ is the nuclei specific gyromagnetic (MHz/T) ratio and B_0 is the strength of the magnetic field (T).

In order to produce an MR signal, $\mathbf{M_0}$ must be moved into the transverse (xy) plane. This is achieved by applying a second magnetic field $\mathbf{B_1}$ precessing around the axis of $\mathbf{B_0}$

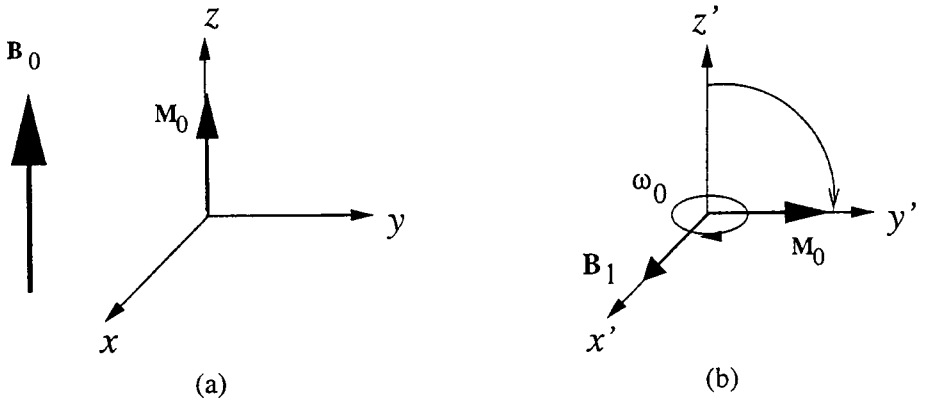


Figure 2.1: Schematic representation of magnetic fields in MRI. (a) $\mathbf{M_0}$ is the net magnetization due to a static magnetic filed $\mathbf{B_0}$. (b) Rotating frame of reference. $\mathbf{M_0}$ is flipped onto the transverse plane by a second, oscillating magnetic field $\mathbf{B_1}$ perpendicular to $\mathbf{B_0}$.

at the Larmor frequency. The effect of $\mathbf{B_1}$ is illustrated in Figure 2.1 (b) in a rotating frame of reference $x'y'z'$ in which the $x'y'$ axis are assumed to be rotating around the z' axis at a frequency ω_0. Since the purpose of $\mathbf{B_1}$ is to flip $\mathbf{M_0}$ 90 degrees from its equilibrium position, it is also called a 90° radio frequency (RF) pulse.

The RF pulse causes the nuclei to go from a low-energy state to a high-energy state. In addition, it rephases the nuclei so that they all precess in coherence. Once the RF pulse is turned off, the nuclei return to their low energy states re-emitting the absorbed energy in the form of a RF signal called the free induction decay (FID) signal. The amplitude of the FID is proportional to the number of "spinning" nuclei in the sample and thus spin density. Right after the RF pulse, the amplitude of the FID is maximum because all protons are aligned in the same direction and precessing in phase. Shortly thereafter, it starts decaying to zero due to three relaxation factors.

2.1.2 Relaxation factors

The net magnetization $\mathbf{M_0}$ can be decomposed in two components: the longitudinal magnetization which is the projection of $\mathbf{M_0}$ onto the z'-axis, and the transverse magnetization which is the projection of $\mathbf{M_0}$ into the $x'y'$ plane. Only the transverse component of $\mathbf{M_0}$ can be measured by the receiving coils. After the 90° RF pulse, the transverse magneti-

Table 2.1: Approximate relaxation time values of hydrogen protons for various tissues at a magnetic field of 1.0 Telsa (From [41]).

Tissue	T_1 *ms*	T_2 *ms*
Fat	180	90
Muscle	600	40
Gray matter	520	100
White matter	390	90

zation is maximum but decays quickly because of spin dephasing.

$T1$ and $T2$ relaxation

Longitudinal or T_1 relaxation is a measure of the time in which the longitudinal component of $\mathbf{M_0}$ recovers back to equilibrium. T_1 is also known as spin-lattice relaxation because it is the result of the loss of energy emitted by the nuclei to their environment.

On the other hand, *transverse* or T_2 relaxation is determined by the rate of the loss of the transverse magnetization. It is also referred to as spin-spin relaxation because it is the result of the loss in phase coherence of nuclei as they exchange energy with each other.

Both T_1 and T_2 are characteristic of the nuclei and the surrounding tissue. For example, hydrogen protons resonating in fat will have different T_1 and T_2 from hydrogen protons resonating in muscle tissue. Approximate values for T_1 and T_2 for various tissues are tabulated in Table 2.1.

$T2^*$ relaxation

The *effective* relaxation factor, also called $T2^*$, is due to constant spatial inhomogeneity of the magnetic field. Nuclei located at different positions experience slightly different magnetic fields which results in a loss of phase coherence or transverse dephasing. $T2^*$ causes the transverse magnetization to decay faster than $T1$ and $T2$. In fact, most of the FID signal loss is due to $T2^*$. In general, the relaxation factors have the following

relationship:

$$T2^* \leq T2 \leq T1.$$

2.1.3 Spin echo: recovering the signal

The two major sources for transverse relaxation are $T2$ and $T2^*$. However, since the loss of coherence due to $T2^*$ occurs at a constant rate it is possible to reverse its effects by the application of a 180° RF pulse, as we now explain.

Consider the situation displayed in Figure 2.2. In (a), the transverse magnetization is maximum and all spins are in phase after the 90° pulse. As shown in (b), and due to magnetic field inhomogeneity, phase coherence is eventually lost and the spins precess at different frequencies. At this moment, a 180° pulse applied along the x' axis reverses the direction of spin precession, as shown in (c). In consequence, spins that were precessing at faster rates are now behind of spins precessing at slower rates. Eventually all spins come back into phase, as shown in (d), at which point an echo of the original FID signal is produced. The time elapsed since the 90° pulse and the center of the echo is called echo time or TE. The time elapsed since the 90° pulse and the 180° pulse is $TE/2$.

As shown in Figure 2.3, it is possible to produce several echoes by applying several 180° pulses at appropriate times. Due to irreversible $T2$ phase loss, however, the amplitude of each new echo is smaller than the original FID by the amount of $T2$ dephasing that has occurred since the 90° pulse. In the context of an MRI experiment, several spin-echo pulse sequences are necessary to sample the entire object. The time between two consecutive pulse sequences (two consecutive 90° pulses) is called repetition time or TR.

2.2 Image formation

This section is based on [7, 6, 28, 38, 2].

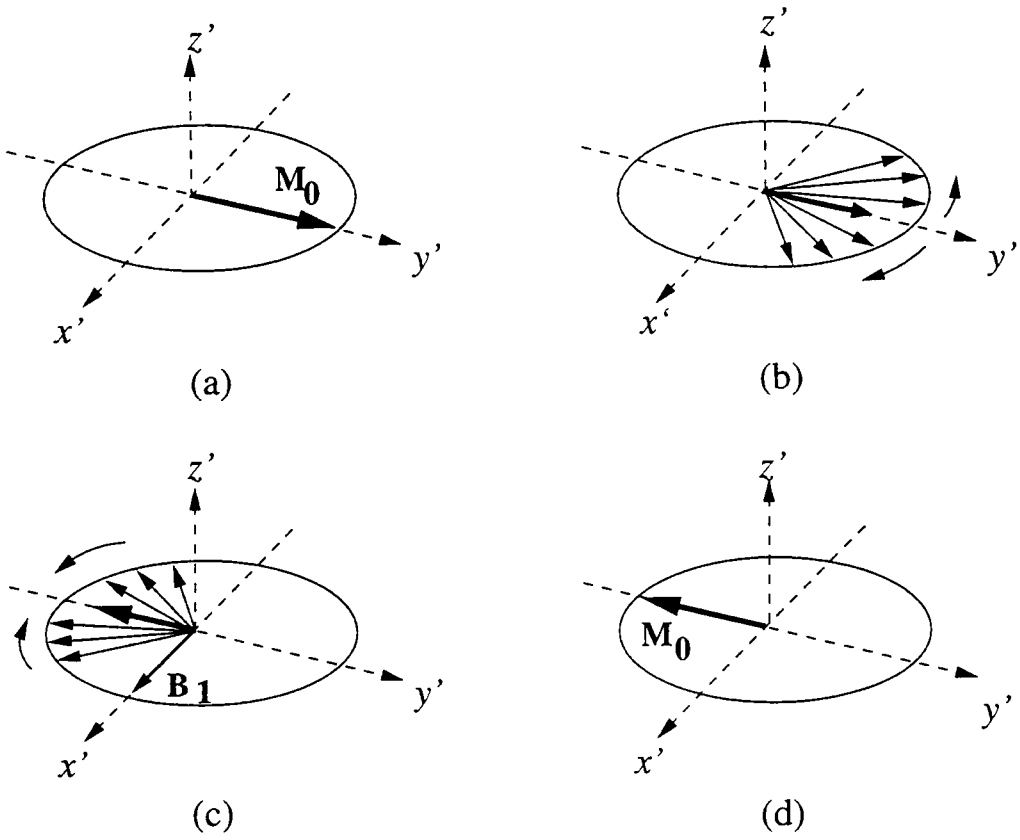


Figure 2.2: Spin echo rephasing and formation of the spin echo signal. (a) All spins are in phase and the FID is maximum. (b) After a time t_1 some dephasing has occurred. (c) The application of a 180° $\mathbf{B}_1$ pulse reverses the phase so that $\mathbf{M}_0$ points in the opposite direction. (d) After a time t_1 spins regain phase coherence and a spin-echo signal is produced.

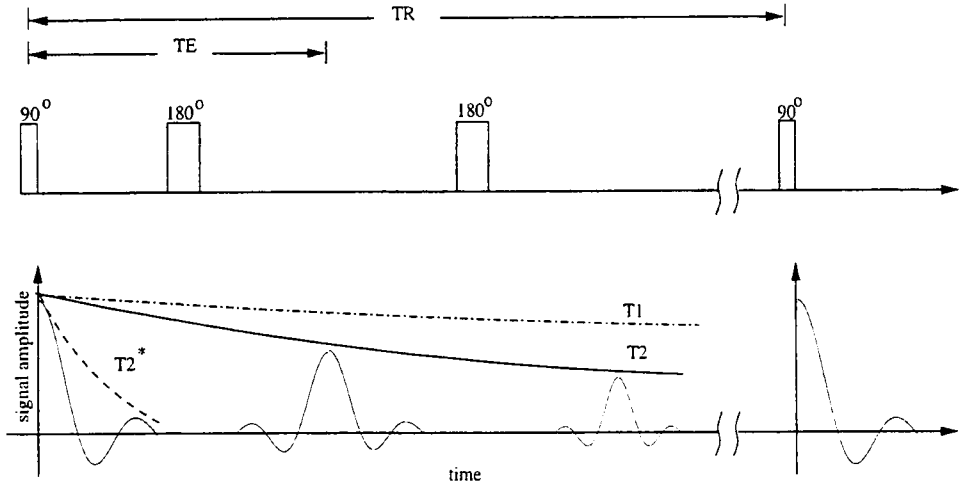


Figure 2.3: Pulse sequence for spin echo.

2.2.1 k space

Consider now introducing a time-varying magnetic field gradient $\mathbf{G}(t)$ whose axial components $G_x(t)$, $G_y(t)$, and $G_z(t)$ are carefully controlled. When superimposing $\mathbf{G}(t)$ on $\mathbf{B}_0$, local magnetic field variations result and the resonance frequency of the corresponding nuclei change accordingly. This defines a time-dependent sampling trajectory in the so-called k-space which is the name used for the Fourier spectrum of the object being imaged. Thus $\mathbf{k}(t) = [k_x(t), k_y(t), k_z(t)]^T$ is the spatial frequency position to be sampled at time t. The dependence of $\mathbf{k}(t)$ on the gradient is given by:

$$\mathbf{k}(t) = \gamma \int_0^t \mathbf{G}(\tau) d\tau. \tag{2.1}$$

Ignoring $T2^*$ relaxation, the FID can be expressed as

$$S(\mathbf{k}(t)) = M_0 \int \rho(\mathbf{x}) \exp\{-i\, \mathbf{x}^T \mathbf{k}(t)\} d\mathbf{x} \tag{2.2}$$

where

$$\rho(\mathbf{x}) = \rho_0(\mathbf{x}) \exp\left\{\frac{-TE}{T2}\right\}\left(1 - \exp\left\{\frac{-TR}{T1}\right\}\right) \tag{2.3}$$

and ρ_0 is the initial amplitude of the spin density as a function of $\mathbf{x}$ and M_0 is the strength of $\mathbf{M}_0$. The two exponential factors are the $T2$ and $T1$ relaxation decays, respectively. Note that without $\mathbf{G}(t)$, Equation 2.2 degenerates to a simple volume integral from which the unknown spin density $\rho(\mathbf{x})$ cannot be recovered. Conversely, when $\mathbf{G}(t)$ is used to encode the FID, $\rho(\mathbf{x})$ can be obtained by Fourier inversion. The spatial resolution is a function of how $\mathbf{G}(t)$ is applied.

2.2.2 Two-dimensional Cartesian sampling

With reference to Figure 2.4, a version of a spin-echo pulse sequence for obtaining a 2D image that maps the spin density within a slice of the object is now explained. The first

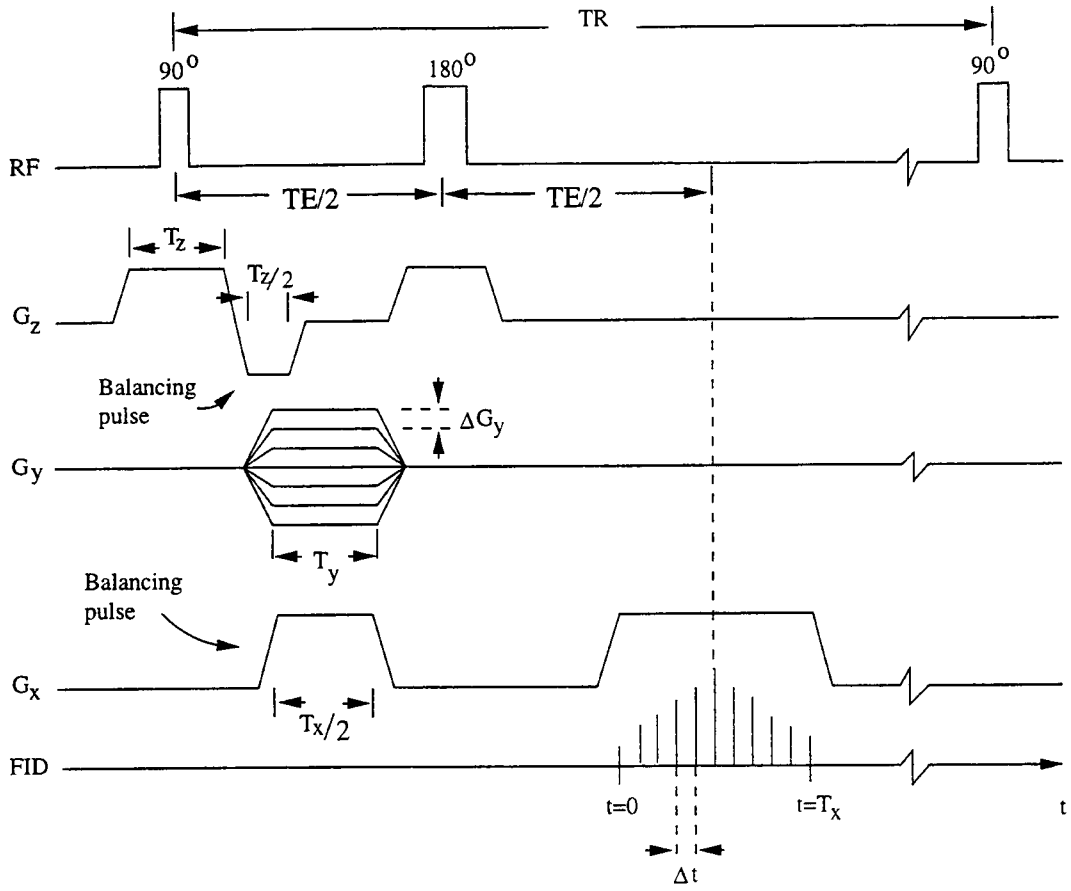


Figure 2.4: Timing diagram of a spin-echo pulse sequence designed for 2D MRI.

line shows the RF pulse activity, the following three lines correspond to the independently controlled axial gradients, and the last line shows the data acquisition timing.

First, a tuned 90° RF pulse is broadcast to flip the net magnetization into the transverse plane and G_z is pulsed to introduce a linear gradient in the z direction. This step ensures that only the nuclei located in the slice defined by the frequency content of the RF signal become resonant. Assume the slice selected is at position $z = z_0$. Next, G_y is pulsed to introduce a linear gradient along the y direction. This causes spins to have different precessional frequencies and phase according to their position along the gradient. When the gradient is turned off, the spins return to have the same frequency as determined by the main magnetic field, but now with different phases. The net effect is that position along the y direction has been encoded with phase.

Recall that after the 90° RF pulse the signal immediately begins to decay due to $T2^*$ relaxation. In order to rephase the spins, a 180° RF pulse is applied to the same slice of tissue at time $TE/2$.

The readout gradient, G_x, is used to encode the x direction with frequency, that is spins will precess faster or slower depending on their position with respect to the gradient. It has the similar effects as G_y but this time the frequency change is recorded as the signal is immediately sampled.

G_y is pulsed with a different value, proportional to ΔG_y, after a time TR. The number of phase encoding steps (number of G_y's) defines the resolution in the y direction which are the number of rows in the final image. On the other hand, the frequency encoding direction (G_x) is sampled always with the same amplitude G_x at a rate Δt. The total number of points sampled defines the image resolution in the x direction. At the (m, n)th step, the sampling trajectory is given by

$$\begin{aligned} k_x(t) &= \gamma m \Delta t G_x \qquad\qquad 0 \le t \le T_x \\ k_y(t) &= \gamma n \Delta G_y T_y \\ k_z(t) &= \gamma G_z T_z. \end{aligned}$$

Assuming that M_y lines in the y direction are collected each consisting of M_x samples along the x direction, the two-dimensional data set is given by:

$$S(m,n) = M_0 \iint \rho(x, y, z = z_0) \exp\{-i\gamma(m\Delta t G_x x + n\Delta G_y T_y y)\}\, dx\, dy$$

for $0 \le m \le N_x - 1$ and $-M_y/2 + 1 \le n \le M_y/2$.

The purpose of the *balancing* pulses in a spin-echo sequence is now explained. The application of magnetic field gradients has the effect of changing the precessional rates and phase of spins. This is the mechanism by which the signal is encoded with phase and frequency and permits us to resolve the spin density up to a point. However, when G_z, the slice select gradient, is applied with the 90° RF pulse, a momentary spin dephasing

also occurs which is not necessary for signal encoding. Phase dispersion begins in the second half, of the 90° pulse, increases linearly while the gradient is on (assuming constant gradient), and ends when G_z is turned off. In order to compensate for this phase dispersion, G_z must be turned on in the opposite direction, for as long as half the slice selection time. After the application of this balancing pulse, the spin phase dispersion is zero.

The second application of G_z does not need a balancing pulse because the 180° RF pulse inverts the phase of the spins, so that the phase coherence lost during the first half of G_z is recovered during the second half of G_z. In this case, phase dispersion begins immediately with G_z because the spins are now in the transverse plane.

The readout gradient also needs a balancing pulse to negate the spin dephasing caused by G_x during data acquisition. The G_x balancing pulse introduces a positive phase dispersion during a time $T_x/2$ which is inverted by the following 180° pulse and stays constant until the readout period begins. The second G_x pulse also introduces a phase dispersion such that at the center of the spin-echo it is zero and the signal has maximum strength.

2.2.3 Three-dimensional Cartesian sampling

Although a 3D image could be obtained by stacking a sequence of reconstructed 2D images, it is advantageous to use a true 3D sampling procedure. The main reason is that each data point thereby is made to reflect properties of the entire volume and not just a slice. This typically results in images with better signal-to-noise ratios and allows oblique images to be reconstructed without having to rescan the object. The 3D sampling is a straightforward extension of the 2D sampling described above. As shown in Figure 2.5, the idea is to excite the entire volume for sampling using a non-selective 90° RF pulse prior to any of the gradient pulses. Then, a *first* phase-encoding gradient pulse, say G_z, is applied to partition the excited volume into thin slices. Simultaneously, a *second* phase-encoding gradient pulse, say G_y, is applied to resolve structures within each slice. Finally, G_x is pulsed to frequency-encode individual points within the strips of the slices. One of the phase-encoding gradients is nested within the other and at the (m, n, p)th step, the

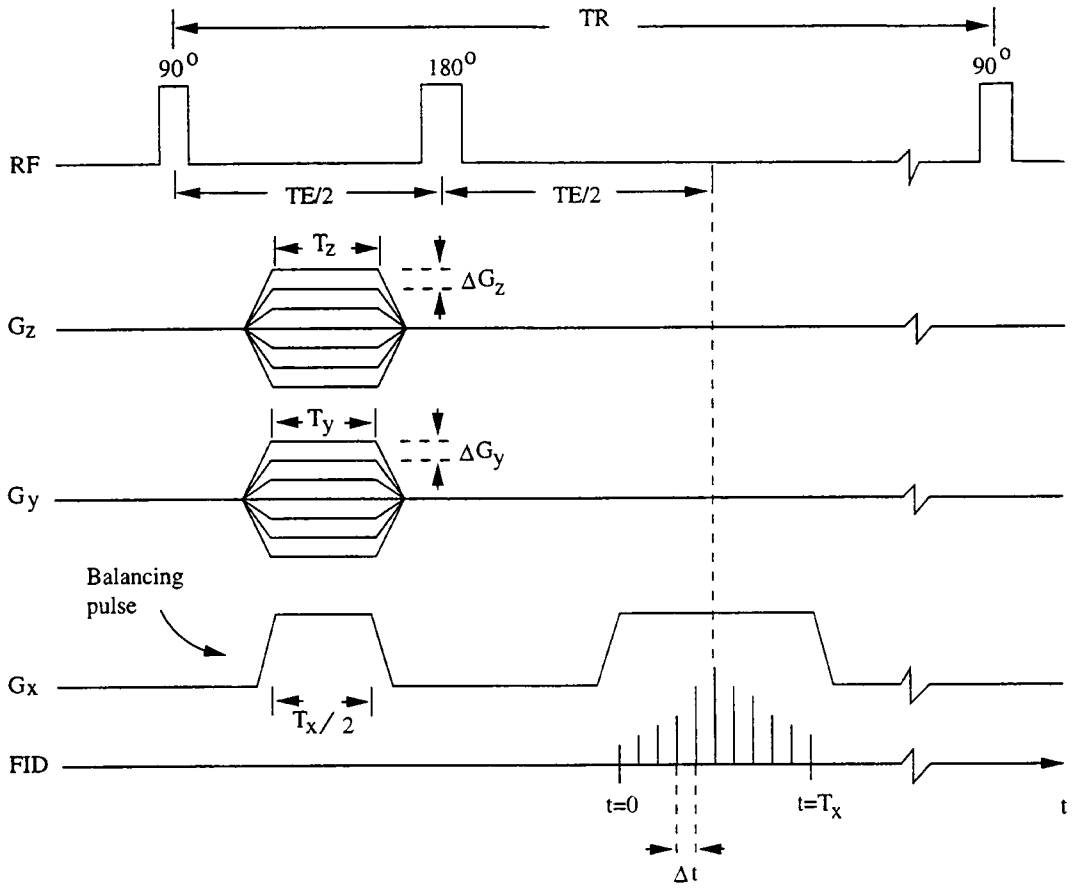


Figure 2.5: Timing diagram of a spin-echo pulse sequence designed for 3D MRI.

sampling trajectory is given by

$$k_x(t) = \gamma m \Delta t G_x \qquad 0 \leq t \leq T_x$$

$$k_y(t) = \gamma n \Delta G_y T_y$$

$$k_z(t) = \gamma p \Delta G_z T_z.$$

Typically, the various parameters are chosen to sample the k-space as if it were a Cartesian grid with equidistant points.

Assuming we partition the volume in $M_z \times M_y \times M_x$ voxels, the three-dimensional

data is:

$$S(m,n,p) = M_0 \iiint \rho(x,y,z) \exp\{-i\gamma(m\Delta t G_x x + n\Delta G_y T_y y + p\Delta G_z T_z z)\}\, dx\, dy\, dz$$

for $0 \leq m \leq N_x - 1$, $-M_y/2 + 1 \leq n \leq M_y/2$, and $-M_z/2 + 1 \leq p \leq M_z/2$.

2.3 Boron imaging

It has been observed [9, 17] that Cartesian sampling is not adequate for boron imaging, due to the short relaxation times. As shown in Table 2.2, $T2$ is of the order of 0.6 milliseconds for boron. However, in standard MRI units using the traditional encoding schemes outlined above the FID cannot be detected in less than one millisecond after a preparatory RF pulse. It is undesirable therefore to spend time in lengthy pulses for slice selection and phase encodings, while the signal decays fast to zero because of $T2$ relaxation.

Table 2.2: Boron relaxation times (From [9]).

Isotope	γ (Mhz/T)	$T1$ *(ms)*	$T2$ *(ms)*	M_0 (T)
B-11	13.66	0.93	0.65	2.0
		1.68	0.78	4.7
B-10	4.59	1.2	0.84	2.0
		1.1	0.94	4.7

In order to alleviate this problem, a 3D projection reconstruction method for nuclei with short $T2$ was described in [9]. With reference to Figure 2.6, the method is based on sampling data along spherical trajectories (θ, ϕ) of the Fourier spectrum where angles θ and ϕ define latitude and longitude positions, respectively.

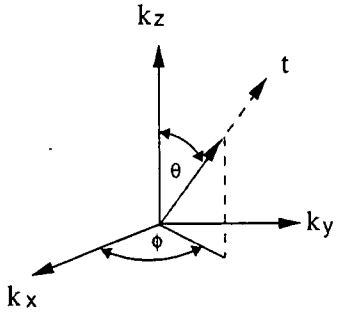


Figure 2.6: Spherical sampling of the Fourier spectrum. The number of rays and their orientations are chosen to cover the spectrum evenly.

The pulse sequence shown in Figure 2.7 starts with a nonselective 90° RF pulse that excites the entire volume of interest. Then, all three magnetic field gradients are turned on simultaneously to create a projection readout gradient. Specifically, at direction (θ, ϕ) we have:

$$\begin{aligned} k_x(t) &= \gamma t \sin\theta \cos\phi G_x \\ k_y(t) &= \gamma t \sin\theta \sin\phi G_y \qquad 0 \le t \le T_x = T_y = T_z \\ k_z(t) &= \gamma t \cos\theta G_z \end{aligned}$$

From Equation (2.2) and assuming $G_x = G_y = G_z = G$ we have that:

$$S(\mathbf{r}, t) \triangleq S(\mathbf{k}(t)) = M_0 \int \rho(\mathbf{x}) \exp\{-i\gamma t\, G\mathbf{x}^T\mathbf{r}\}\, d\mathbf{x} \tag{2.4}$$

where $\mathbf{r} = [\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta]^T$. Note that the very short relaxation times of boron calls for sampling to take place even during the ramp time, i.e., before the magnetic field gradients have stabilized. The resulting nonuniform sampling of the k-space along each radial direction can be corrected by means of 1D interpolation.

This procedure is repeated for various angles θ and ϕ such that the 3D Fourier spectrum of the object is covered as uniformly as possible. We will come back to this procedure in Chapter 4.

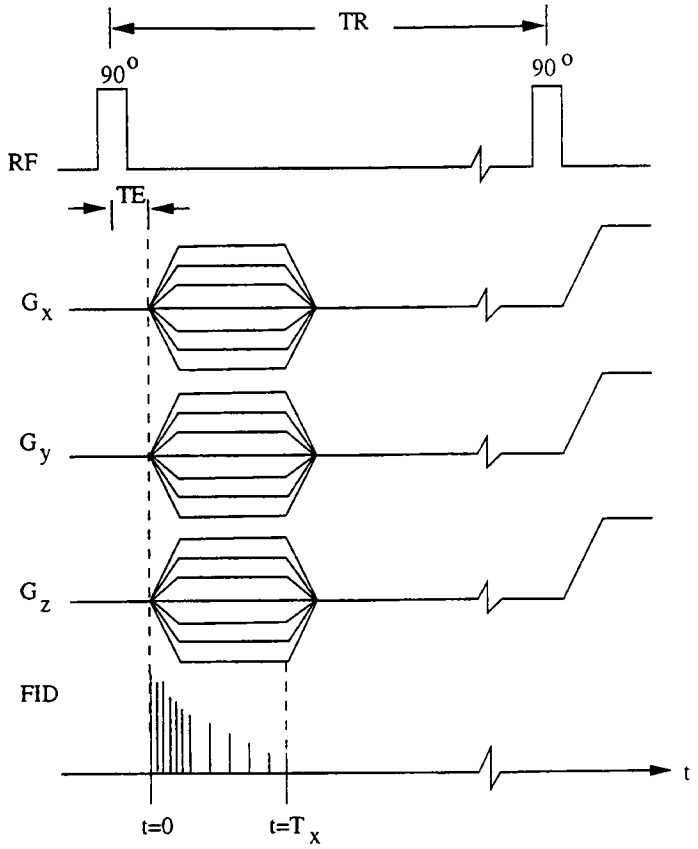


Figure 2.7: Pulse sequence for projection reconstruction in boron imaging. After the RF pulse is applied $G_x(t)$, $G_y(t)$, and $G_z(t)$ are turned on and data sampling begins.

2.4 Image reconstruction

In the MRI literature, we often find the term "Fourier" reconstruction or Fourier imaging used to denote image reconstruction from Cartesian sampling. Likewise, "projection" reconstruction is used for image reconstruction from polar or spherical sampling. In fact, any MRI sampling strategy will always require the use of a Fourier transform in one way or another because the sampled data in **k**-space corresponds to samples from the object's Fourier spectrum.

2.4.1 Fourier reconstruction

The three-dimensional Fourier transform of a function $f(\mathbf{x})$ is defined as

$$\hat{f}(\mathbf{u}) \triangleq [\mathcal{F}_3 f](\mathbf{u}) = \int_{\mathbb{R}^3} f(\mathbf{x}) \exp\{-i\, 2\pi \mathbf{x}^T \mathbf{u}\}\, d\mathbf{x} \tag{2.5}$$

where $\mathbf{u}$ denotes spatial frequency. Correspondingly, the inverse Fourier transform of $\hat{f}(\mathbf{u})$ is

$$f(\mathbf{x}) \triangleq [\mathcal{F}_3 \hat{f}]^{-1}(\mathbf{x}) = \int_{\mathbb{R}^3} \hat{f}(\mathbf{u}) \exp\{2\pi i \mathbf{u}^T \mathbf{x}\}\, d\mathbf{u}. \tag{2.6}$$

We will also write $[\mathcal{F}_1]$ to denote the one-dimensional Fourier transform. From Equations (2.1), (2.2) and using a suitable change of variables, we find that

$$S(\mathbf{k}) = M_0[\mathcal{F}_3 \rho](\mathbf{k}). \tag{2.7}$$

That is, the MRI signal corresponds to the Fourier transform of the spin density. Therefore, the spin density can be obtained with direct Fourier inversion:

$$\rho(\mathbf{x}) = (1/M_0)[\mathcal{F}_3 S]^{-1}(\mathbf{x}). \tag{2.8}$$

2.4.2 Projection reconstruction

The mathematical basis that unifies different reconstruction problems is the Radon transform. In two dimensions the Radon transform is the same as the X-ray transform which is used in computed tomography (CT) and emission tomography (ET). The difference between the Radon and X-ray transform is that the former integrates along hyperplanes and the latter only along rays. Here, we are only interested in the Radon transform. An excellent account on the subject can be found in [31].

The 3D Radon transform of a function $f(\mathbf{x})$, is defined as its integrals over planes

of $\mathbb{R}^3$. Specifically, if $\mathbf{r} = [\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta]^T$ is a unit vector in $\mathbb{R}^3$, $\mathbf{r}^\perp$ is the subspace of vectors perpendicular to $\mathbf{r}$, and $\tau \in \mathbb{R}$, then

$$p(\mathbf{r}, \tau) = [\mathcal{R}_3 f](\mathbf{r}, \tau) = \int_{\mathbf{r}^\perp} f(\tau\mathbf{r} + \mathbf{x})\, d\mathbf{x} \tag{2.9}$$

is the Radon transform of f over the plane orthogonal to $\mathbf{r}$ at a distance τ from the origin. The quantity $p(\mathbf{r}, \tau)$ is also called a projection. Alternatively, the Radon transform can be expressed in terms of the one-dimensional δ-function

$$[\mathcal{R}_3 f](\mathbf{r}, \tau) = \int_{\mathbb{R}^3} f(\mathbf{x})\delta(\tau - \mathbf{r}^T\mathbf{x})\, d\mathbf{x}. \tag{2.10}$$

The inversion of the integral equation (2.9) was solved by Radon [34] and requires the use of the projection theorem, also known as Fourier slice theorem. Let $P(\mathbf{r}, t)$ be the one-dimensional Fourier transform of projection data $p(\mathbf{r}, \tau)$ with respect to τ and $w \in \mathbb{R}$. Then, the projection theorem reads

$$P(\mathbf{r}, t) = \hat{f}(w\mathbf{r}). \tag{2.11}$$

That is, the 1D Fourier transform of a planar projection orthogonal to $\mathbf{r}$ of a function f corresponds to a central ray (one passing through the origin) of the 3D Fourier transform of f at direction $\mathbf{r}$ (see Figure 2.8).

The inversion of Equation (2.9) is given by the backprojection operator $\mathcal{B}$:

$$f(\mathbf{x}) = [\mathcal{B}\hat{p}](\mathbf{x}) \triangleq \int_0^\pi \int_0^{2\pi} \hat{p}(\mathbf{r}, \tau) \sin\theta\, d\theta\, d\phi \tag{2.12}$$

where

$$\hat{p}(\mathbf{r}, \tau) = -\frac{1}{8\pi^2} \frac{\partial^2 p(\mathbf{r}, \tau)}{\partial \tau^2}$$

are filtered projections.

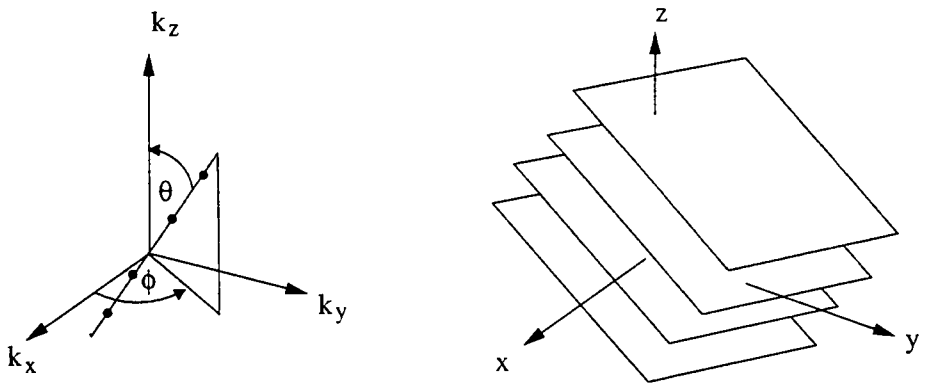


Figure 2.8: Relation between the k-space and object space. A line through the origin of the Fourier space (a) corresponds to a planar projection in object space (b).

We notice that although boron data $S(\mathbf{r}, t)$ (see Equation (2.4)) is sampled along spherical trajectories, it does not necessarily correspond to central rays of the Fourier spectrum. They can be considered as half projections extending from the origin to high frequencies in k-space. In order to use the projection theorem we would need two half projections line up so that one is the continuation of the other at the opposite side of the k-space. We will come back to this issue in Chapter 4 where we describe a solution for this problem.

Image reconstruction from boron data $S(\mathbf{r}, t)$ in [9] is carried out in four steps: (1) each projection is resampled by means of 1D interpolation, (2) the new projection data is used to fill the 3D Fourier space in a polar fashion, (3) the polar data is interpolated to yield 3D Cartesian data, and (4) the spin density is computed with a 3D fast inverse Fourier transform of the Cartesian data. Alternatively, if $S(\mathbf{r}, t)$ were sampled as needed by the projection theorem, the spin density could also be recovered with filtered-back projection as follows: (1) planar projections are obtained by a 1D inverse Fourier transform of each projection readout, (2) filtered projections are obtained by taking second order differences to the planar projections, and (3) the spin density is computed by backprojecting the filtered projections.

Fourier inversion and filtered back projection are transform methods in the sense that they reconstruct the image by inverting the Radon integral in a few steps by the appli-

cation of Fourier transforms. These algorithms are the commercial standards for image reconstruction not only in MRI, but also in other areas of medical imaging like CT and SPECT/PET.

Chapter 3

Image Basis Representation

3.1 Introduction and overview

There are two fundamentally different approaches to solve the projection image reconstruction problem at hand: transform (or direct) and series expansion (or iterative) methods. In the transform approach, the solution is obtained by inverting the Radon integral equation. The problem is formulated and handled in the continuous domain until the final formulas are discretized for computer implementation. In the series expansion approach, the problem is discretized at the beginning by expanding the image as a finite linear combination of basis functions. The problem of finding an image is translated into one of finding coefficients such that the image they produce is as close as possible to the true image. This is usually done by minimizing an optimization criterion with iterative algorithms.

The boron image reconstruction approaches described in the previous Chapter both constitute transform methods. This dissertation focuses on the feasibility and performance of iterative methods for solving the same problem. In order to do so, we define in this chapter all the necessary elements involved in series representation for image reconstruction.

3.2 Problem formulation

In the finite series-expansion approach to image reconstruction [3], an image f is assumed to be a linear combination of translated versions of a basis function, say ψ:

$$\begin{aligned} f(\mathbf{x}) \approx \hat{f}(\mathbf{x}) &= \sum_{j=0}^{N-1} c_j \psi(\mathbf{x} - \mathbf{x}_j) \\ &= \sum_{j=0}^{N-1} c_j \psi_j(\mathbf{x}) \end{aligned} \tag{3.1}$$

where $\{c_j \in \mathbb{R}\}$ is a set of coefficients, $\{\mathbf{x}_j \in \mathbb{R}^3\}$ is a set of grid points, and $\psi_j(\cdot)$ is the basis function positioned at $\mathbf{x}_j$. We stress the fact that a single function is translated and scaled to create the image. A more general series representation would use different functions at different grid positions.

With reference to the projection geometry shown in Figure 3.1 we take the 3D Radon transform along the plane given by unit vector $\mathbf{r} = [\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\phi]^T$ and distance τ from the origin to Equation (3.1) and obtain

$$[\mathcal{R}_3\hat{f}](\mathbf{r}, \tau) = \sum_{j=0}^{N-1} c_j [\mathcal{R}_3\psi_j](\mathbf{r}, \tau) \tag{3.2}$$

$$p(\mathbf{r}, \tau) = \sum_{j=0}^{N-1} c_j \Psi_j(\mathbf{r}, \tau) \tag{3.3}$$

where $p(\mathbf{r}, \tau)$ is the projection data (input data) and $\Psi_j(\mathbf{r}, \tau)$ is the 3D Radon transform of basis function ψ_j.

The reconstruction problem in the series expansion approach can be formulated as the inversion of Equation (3.3), that is, given projection data $p(\mathbf{r}, \tau)$ we find a finite set of coefficients $\{c_j\}$ such that the estimate $\hat{f}$ defined by Equation (3.1) is as close as possible to the true image f. For a given reconstruction problem $p(\mathbf{r}, \tau)$ is fixed and independent of the basis functions. Therefore, different choices of basis functions will produce different image estimates for the same input data.

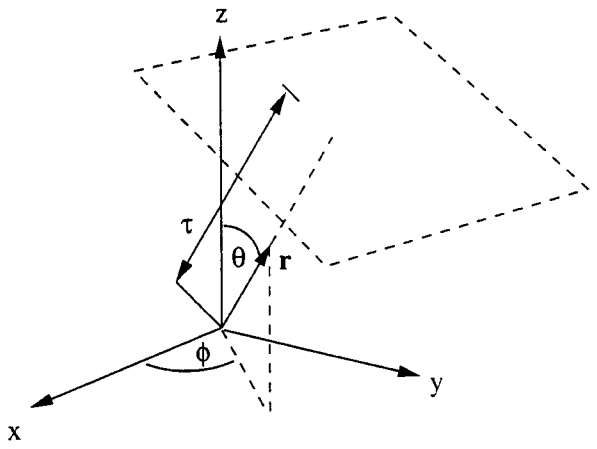


Figure 3.1: The 3D projection reconstruction geometry.

We can further simplify Equation (3.3) by using the shifting property of the Radon transform which is illustrated for the two-dimensional case in Figure 3.2. Using Equation (2.10), the alternative definition of the Radon transform, we find that:

$$\Psi_j(\mathbf{r},\tau) = \int_{\mathbb{R}^3} \psi_j(\mathbf{x})\delta(\tau - \mathbf{r}^T\mathbf{x})\, d\mathbf{x} \tag{3.4}$$

$$= \int_{\mathbb{R}^3} \psi(\mathbf{x} - \mathbf{x}_j)\delta(\tau - \mathbf{r}^T\mathbf{x})\, d\mathbf{x} \tag{3.5}$$

$$= \int_{\mathbb{R}^3} \psi(\mathbf{y})\delta(t' - \mathbf{r}^T\mathbf{y})\, d\mathbf{y} \tag{3.6}$$

$$= \Psi(\mathbf{r}, t') \tag{3.7}$$

where $\mathbf{y} = \mathbf{x} - \mathbf{x}_j$ and $t' = \tau - \mathbf{r}^T\mathbf{x}_j$. In other words, the Radon transform of any shifted basis function can be computed using the Radon transform of the basis function centered at the origin.

3.3 Basis functions

3.3.1 General properties

Some desirable properties of a basis function for 3D image reconstruction are:

- Ability to approximate images with minimum error;

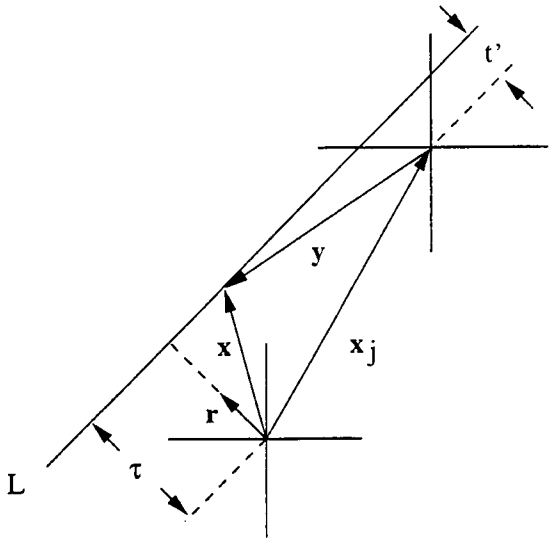


Figure 3.2: Illustration of the shifting property of the Radon transform in 2D.

- Linear independence;
- Good localization in space and Fourier domains;
- Efficient computation of the Radon transform; and
- Spherical symmetry.

For all practical purposes, a complete set of orthogonal basis functions can exactly represent any image with a unique set of coefficients. However, since exact reconstruction is incompatible with symmetry and the reconstruction problem is ill-posed in the sense that the projection data does not uniquely define the image, we reconstruct image estimates that satisfy less demanding but more realistic properties. Good localization in space means that the function is locally defined, i.e., that the overlap of adjacent versions of the basis function is small. Localization of a basis function in the Fourier domain implies that most of its spectral energy falls within a specified bandwidth. In general, good localization in space implies poor localization in frequency and good localization in frequency means poor localization in space. The objective is then to find functions that deliver a good compromise between these two properties.

A typical set of basis functions arises when the image is discretized in a set of non-overlapping, and therefore orthogonal, square voxels (volume elements) and assumed to be constant within each voxel region. Then we have:

$$\psi_j(\mathbf{x}) = \begin{cases} 1 & \text{inside the } j\text{th voxel} \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

Since these functions are completely localized in space and they do not overlap each other, the computation of f is greatly simplified. In fact, the set $\{c_j\}$ is the same as the image. On the other hand, they are not spherically symmetric making the computation of Ψ_j difficult. Voxels will tend to produce images with sharp transitions which is desirable for boundaries between regions, but not for smooth varying regions. Finally, the Fourier transform of a voxel is a sinc function which is not band-limited.

3.3.2 Effectively band-limited windows

Let $\hat{\psi}$ be the Fourier transform of the window ψ. Then

$$\Omega(W) = \frac{\int_0^W |\hat{\psi}(R)|^2 R^2 \, dR}{\int_0^\infty |\hat{\psi}(R)|^2 R^2 \, dR} \tag{3.9}$$

is a measure of the ratio between the spectral energy within a band of radius W and the total spectral energy. If $\Omega(W) = 1$ then $\hat{\psi}$ is called band-limited to a frequency band of radius W. If $\Omega(W) = 1 - \epsilon$ for some small positive ϵ, then $\hat{\psi}$ is said to be effectively band-limited to a frequency band of radius W at level ϵ.

As discussed in [23] the reasons for using effectively band-limited windows in image reconstruction are:

1. Avoid ghosts. Ghosts are functions whose Radon transforms are zero for a discrete set of directions. They have a Fourier spectra complementary to those of effectively band-limited functions.

2. Correspondence between data and model. Because of sampling restrictions and limited resolution of the projection data, the resulting image is effectively band-limited and recovery of Fourier components above the band limit of the data should not be attempted.

3. Reduce noise: By effectively band-limiting the reconstructed image, we reduce the effects of noise that usually appears in the high frequency components of the spectra.

4. Sampling with negligible aliasing. If the basis function is effectively bandlimited, then the image and its Radon transform are also effectively bandlimited and they can be sampled with negligible aliasing provided the sampling interval is at least twice the effective band limit.

3.3.3 Spherically symmetric windows

From a computational perspective, we are especially interested in basis functions that are compactly supported and are spherically symmetric. Compact support means that the function is zero outside some finite connected region therefore its contribution to the Radon transform is limited to a local region. Symmetry simplifies the computation and storage of the Radon transform of the basis function and also facilitates the handling of symmetries in the projection reconstruction geometry.

The value of a spherically symmetric function at any point $\mathbf{x}$ depends only on the distance from $\mathbf{x}$ to the center of the function, that is

$$\psi(s) \triangleq \psi(\mathbf{x} - \mathbf{x}_j) \tag{3.10}$$

where $s = \|\mathbf{x} - \mathbf{x}_j\|_2$ and $\mathbf{x}_j$ is the center of the function. This property implies that the Radon transform is independent of $\mathbf{r}$ and only depends on the distance from the center to

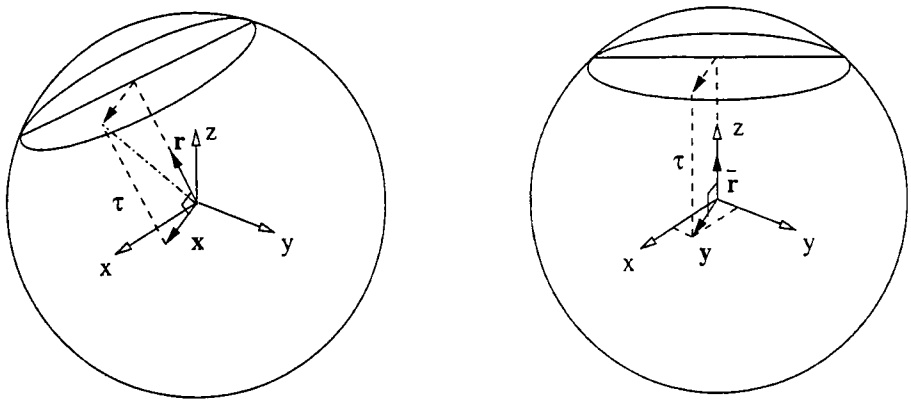


Figure 3.3: Illustration of the independence on $\mathbf{r}$ of the Radon transform for spherically symmetric functions. Vector $\mathbf{x} \in \mathbb{R}^3$ is orthogonal to $\mathbf{r}$ in (a) and the vector $\mathbf{y} \in \mathbb{R}^2$ in (b).

the plane of integration. With reference to Figure 3.3 (a), we have that

$$\begin{aligned} \Psi(\mathbf{r}, \tau) &= \int_{\mathbf{r}^\perp} \psi(\tau\mathbf{r} + \mathbf{x})\, d\mathbf{x} \\ &= \int_{\mathbf{r}^\perp} \psi(\|\tau\mathbf{r} + \mathbf{x}\|)\, d\mathbf{x} \\ &= \int_{\mathbf{r}^\perp} \psi(\sqrt{\tau^2 + \|\mathbf{x}\|})\, d\mathbf{x} \end{aligned} \tag{3.11}$$

Since we are only interested in the norm of $\mathbf{x}$ and not its orientation, we can arbitrarily rotate $\mathbf{r}$ to $\bar{\mathbf{r}}$, as shown in Figure 3.3 (b), and have $\mathbf{x}$ lie in the xy plane. Therefore,

$$\begin{aligned} \Psi(\mathbf{r}, \tau) &= \int_{\bar{\mathbf{r}}^\perp} \psi(\sqrt{\tau^2 + \|\mathbf{x}\|})\, d\mathbf{x} \\ &= \int_{\mathbb{R}^2} \psi(\sqrt{\tau^2 + \|\mathbf{y}\|^2})\, d\mathbf{y} \end{aligned} \tag{3.12}$$

which is the Abel transform of the radial profile ψ, clearly independent of $\mathbf{r}$. For notational convenience, we write this last integral as $\Psi(\tau)$, so that

$$\Psi(\tau) \triangleq \Psi(\mathbf{r}, \tau).$$

3.4 The Kaiser-Bessel basis function

3.4.1 Blob definition and properties

Lewitt [23, 24] introduced a three parameter family of spherically symmetric Kaiser-Bessel window basis functions (called blobs) defined by

(3.13)

$$\psi(s;R,K,\alpha) = \begin{cases} \frac{1}{I_K(\alpha)}\left[\sqrt{1-(s/R)^2}\right]^K I_K\left(\alpha\sqrt{1-(s/R)^2}\right) & 0 \le s \le R \\ 0 & \text{otherwise} \end{cases}$$

where I_K is the modified Bessel function of the first kind order K, α is a scalar that controls the blob shape, and R is the blob radius. Figure 3.4 display the blob's profile for different values of α and fixed $K = 2$. These functions are localized in space and effectively localized in frequency. The functions and their Fourier transforms have rotational symmetry in any dimensions.

The derivative of this function with respect to s is [23]:

$$\frac{\partial\psi(s;R,K,\alpha)}{\partial s} = \left(\frac{-1/R}{\alpha^{K-2}I_K(\alpha)}\right)\left(\frac{s}{R}\right)\left(\alpha\sqrt{1-(s/R)^2}\right)^{K-1} I_{K-1}\left(\alpha\sqrt{1-(s/R)^2}\right)$$

When $K = 0$ and $s = R$, $\partial\psi/\partial s = \infty$, i.e. the first derivative is discontinuous at the radius. This also can be seen from Equation (3.13) as the jump that the function experiences from $\psi(s;R,0,\alpha) = 1/I_0(\alpha) \neq 0$ for $s = R$ to $\psi(s;R,0,\alpha) = 0$ for $s > R$. In general, the Kaiser-Bessel window can be constructed to have k continuous derivatives by choosing $K > k$. In all our experiments we have chosen $K = 2$, so that the first derivative is continuous at the radius. From now on, and for reasons of simplicity we will omit the explicit dependence of ψ on R, K and α.

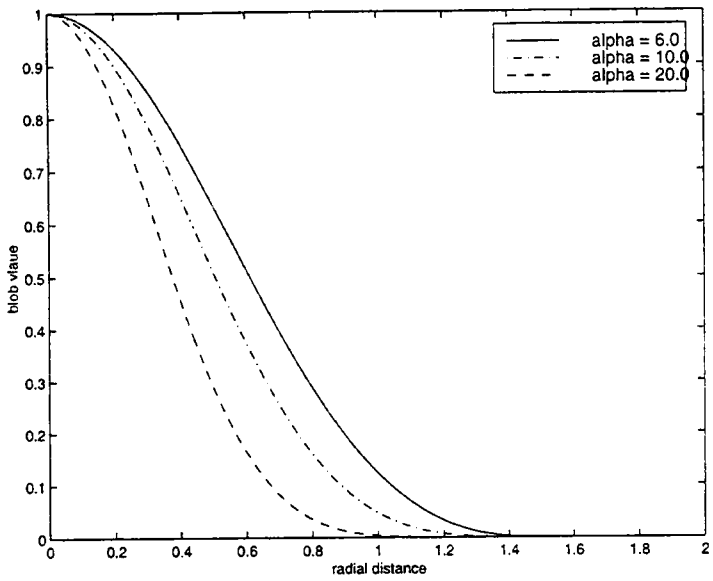


Figure 3.4: The Kaiser-Bessel window basis function for $\alpha = 6, 10$, and 20. $K = 2$ and $R = 1.5$ for all curves.

3.4.2 The three dimensional Radon transform

Lewitt [23] also derived the 3D X-ray transform of a blob. Here we need the 3D Radon transform, because we deal with planar rather than line integral projections. The details of the derivation are in Appendix 1. Since ψ is spherically symmetric, the Radon transform is independent of $\mathbf{r}$ and it suffices to compute the transform along a plane at distance s from the center of the blob. Moreover and without loss of generality, we assume the blob is centered at the origin and that the plane of integration is parallel to the xy plane, as shown in Figure 3.5. Then, from Equation (3.12) we obtain

$$\Psi(s) = \int_{\mathbb{R}^2} \psi(\sqrt{s^2 + \|\mathbf{y}\|^2})\, d\mathbf{y} \quad (\mathbf{y} \in \mathbb{R}^2) \tag{3.14}$$

$$= 4 \int_0^l \int_0^{\sqrt{l^2 - x^2}} \psi(\sqrt{s^2 + x^2 + y^2})\, dy dx \tag{3.15}$$

$$= \frac{4}{I_K(\alpha)} \int_0^l q(x)\, dx \tag{3.16}$$

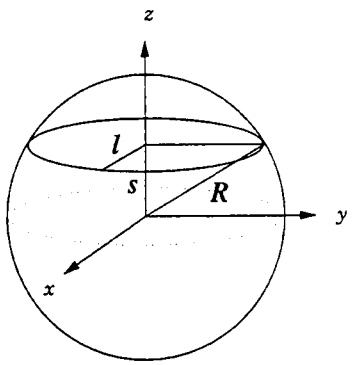


Figure 3.5: Computing the 3D Radon transform. The filled area is the plane of integration, s is the distance from it to the origin, and R is the blob radius.

where

$$q(x) = \int_0^{\sqrt{R^2-(s^2+x^2)}} \left[\sqrt{1 - \frac{s^2+x^2+y^2}{R^2}}\right]^K I_K\left[\alpha\sqrt{1 - \frac{s^2+x^2+y^2}{R^2}}\right] dy. \tag{3.17}$$

After some manipulations and the application Sonine's integral [40], we obtain:

$$\Psi(s) = \begin{cases} \frac{2\pi R^2}{\alpha I_K(\alpha)} \left[\sqrt{1-(s/R)^2}\right]^{K+1} I_{K+1}\left(\alpha\sqrt{1-(s/R)^2}\right) & 0 \le s \le R \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

Figure 3.6 plots the Radon transform for various values of α and fixed blob radius $R = 1.5$.

3.5 Blob parameter selection

The question that naturally arises when using blobs for image representation is which combination of parameter values R and α provides good approximations. Although, the answer to this question is application dependent, theoretical guidelines exist for choosing the parameters. An excellent discussion of the subject together with an experimental study of parameter selection is given in [29].

The experimental study consists in computing an error function between a constant (ideal) function f and its blob representation $\tilde{f}$ for different combinations of R and α.

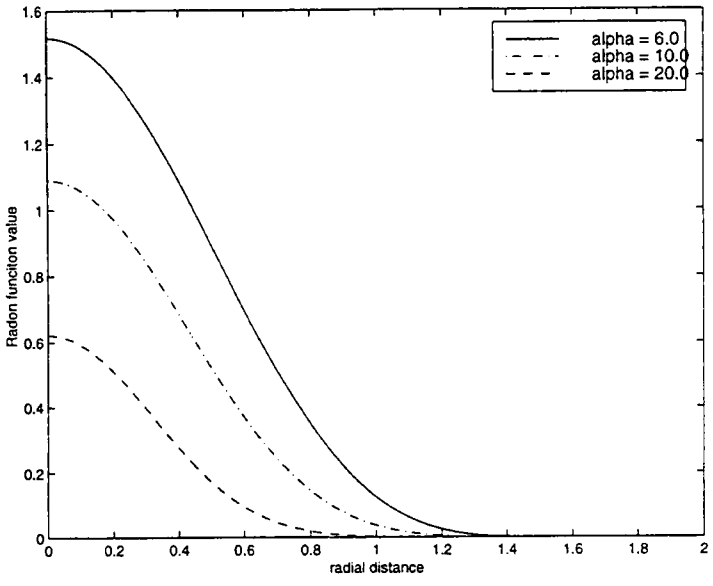


Figure 3.6: 3D Radon transform of blobs for $\alpha = 6.0, 10.0$ and 20.0 and fixed radius $R = 1.5$.

Here, we use the normalized error function:

$$E(f, \tilde{f}) = \frac{\|f - \tilde{f}\|}{\|f\|}. \tag{3.19}$$

For a given pair (R, α), $\tilde{f}$ is evaluated inside a region of unit volume (with $80 \times 80 \times 80$ subsamples) with Equation (3.1). The region can be anywhere in the image, but for simplicity we choose it to be centered at one of the grid points (blob center). The blob radius is relative to the grid increment which is set to 1.0. Then, $\tilde{f}$ is normalized with respect to the value at the region's center such that it approximates the constant function $f = 1.0$. Inside this region, the error is computed for 72×72 (R, α) pairs for $1.0 \leq R \leq 2.6$ and $0.0 \leq \alpha \leq 12.0$.

A contour plot of the logarithm of the error function is displayed in Figure 3.7. There are three main valleys, indicated here with darker gray values, running from small radius and small α to large radius and large α. As we move along a valley from small values of R and α toward large values of R and α, the error decreases rapidly. If R is held constant while α is increased, the error will fluctuate between small and large values as we cross

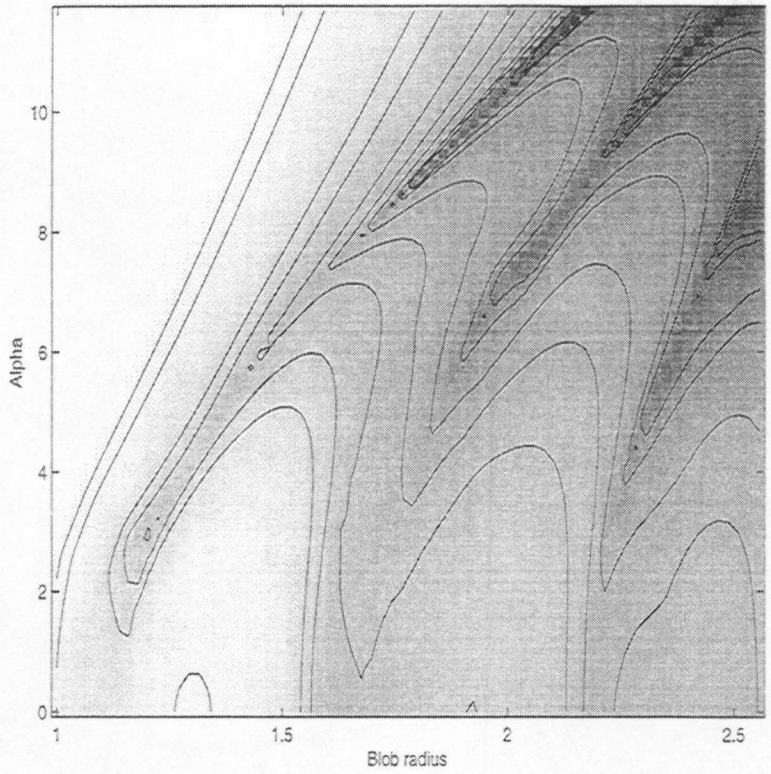


Figure 3.7: Contour graph of the logarithm of the error of approximating a constant function with blobs.

Table 3.1: Optimum values for R and α to approximate a constant function.

R	α	$E(f,\hat{f})(\%)$
1.20	3.0	1.62
1.25	3.6	1.28
1.40	5.3	1.52
1.50	6.4	1.09
2.00	10.3	0.07

the valleys, but it will increase indefinitively for very large values of α. This indicates that both parameters must be increased simultaneously and in accord with the pattern shown here to obtain better approximations of uniform images.

Table 3.1 lists relative error values for (R, α) pairs in the first (left-most) valley. Values of α are chosen to minimize the error function for the given radii. Although the minimums agree with the ones reported in [29], the absolute error function values do not. This may be due to the different resolution used inside the region and/or to a slightly different procedure used for computing the error function.

When choosing blob parameters one is usually constrained to a limited amount of computational resources which is a function of the blob radius. Larger radius (together with larger α's) means better approximation of smooth regions but also greater memory requirements and longer computation times. Thus the choice of a pair (R, α) must take into account not only reconstruction quality but also computational demands.

Chapter 4

Iterative Reconstruction from Boron Data

4.1 Introduction

The main objective of this chapter is to present and resolve issues related to iterative reconstruction of boron images in a distributed environment.

The relaxation properties of the boron signal make Cartesian sampling of the **k**-space inadequate. The signal decays quickly to zero leaving no time for frequency encoding. A projection-based sampling along spherical trajectories must be used instead. By making simple modifications to this strategy, true Radon projection data can be obtained and symmetries can be introduced.

The reconstruction problem is then formulated as solving a very large and sparse system of linear equations. The memory requirements for such a system can be reduced with the symmetries introduced in the sampling geometry.

Least-squares and maximum-likelihood iterative algorithms are reviewed and their distributed implementations are addressed.

4.2 Obtaining projection data

4.2.1 Spherical sampling

A 3D projection reconstruction method for species with short $T2$ is described in [9]. The method is based on sampling the 3D k-space uniformly along spherical trajectories (or directions) $\mathbf{u} \equiv (\theta, \phi)$, $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$, where angles θ and ϕ are elevation and polar directions, respectively. At any given direction $\mathbf{u}$, sampling starts at the origin of the k-space and extends to higher frequencies of the sampling sphere (see Figure 2.6). In this section we describe how the $\mathbf{u}$ directions are chosen such that the sphere is covered as uniformly as possible. The actual data sampling of the MRI signal along one direction is described in the next section.

It is convenient to define a *rotation* and a *reflection* operation for the θ and ϕ directions, respectively. Let $\mathbf{u}_i = (\theta, \phi)$ be a unit vector representing a spherical trajectory. A rotation operation by an angle α is defined as:

$$\mathbf{u}_k = R_\alpha \mathbf{u}_i \equiv (\theta, \phi + \alpha) \tag{4.1}$$

that is, $\mathbf{u}_k$ is obtained by rotating $\mathbf{u}_i$ around the z-axis. A reflection operation by an angle β is defined as:

$$\mathbf{u}_k = T_\beta \mathbf{u}_i \equiv (\theta + \beta, \phi) \tag{4.2}$$

that is, $\mathbf{u}_k$ is the result of reflecting $\mathbf{u}_i$ around the xy plane. Clearly, these two operators commute.

Spherical sampling can be describe as the mechanism by which the surface of the sampling sphere is covered by the direction ray endpoints. In this context, the terms sampling and sample refer to the ray endpoints on the sphere and not to traditional signal data sampling. With reference to Figure 2.6, at any elevation angle θ, the strategy is to obtain samples around the circle of radius $\sin\theta$ perpendicular to the k_z axis. This process

is repeated for several elevation angles such that the sphere is covered uniformly.

Let N_θ be a prespecified number of elevation angles. Then for uniform sampling of θ, we set $\Delta\theta = \pi/(N_\theta - 1)$ and $\theta_m = m\Delta\theta$ for $m = 0, \ldots, N_\theta - 1$.

To sample ϕ, we first define the sampling density (number of ray endpoints per arc length) at elevation angle θ_m as:

$$D(\theta_m) = \frac{N_\phi^m}{2\pi \sin\theta_m}$$

where N_ϕ^m is the number of polar samples at elevation θ_m and $2\pi \sin\theta_m$ is the circumference of the circle of radius $\sin\theta_m$. If we choose N_ϕ^m to be constant for all θ_m, that is uniform sampling of ϕ, then the density of samples will be higher near the poles ($\theta_m = 0, \pi$) than near the equator ($\theta_m = \pi/2$). To achieve a constant density $C \in \mathbb{R}$ at all elevation angles, we need $D(\theta_m) = C$, which implies that

$$N_\phi^m = 2\pi C \sin\theta_m \tag{4.3}$$

$$\Delta\phi_m = \frac{2\pi}{N_\phi^m} = \frac{1}{C \sin\theta_m} \tag{4.4}$$

where $\Delta\phi_m$ is the elevation dependent sampling interval of the polar direction. Constant C can be found by adding the additional requirement that $\Delta\theta = \Delta\phi_m$ at elevation $\theta_m = \pi/2$. That is, the sampling interval of the great circle passing through the poles is the same as the sampling interval of the great circle around the equator. Then, we have that

$$\Delta\theta = \Delta\phi_m$$

$$\frac{\pi}{N_\theta - 1} = \frac{1}{C \sin\theta_m} = \frac{1}{C} \quad \Longrightarrow \quad C = \frac{N_\theta - 1}{\pi}$$

From (4.3) we obtain that $N_\phi^m = 2(N_\theta - 1)\sin\theta_m$. To guarantee one sample at both poles, we set

$$N_\phi^m = 2(N_\theta - 1)\sin\theta_m + 1.$$

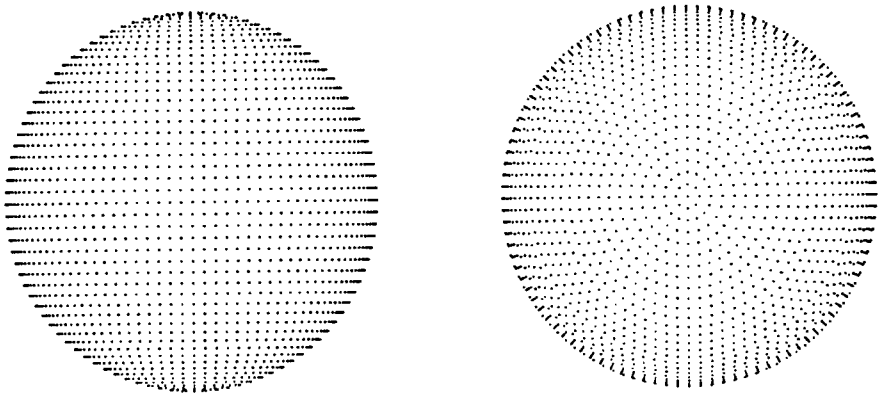


Figure 4.1: Distribution of projections in k-space. (a) side view, (b) top view.

Finally, let i be the index denoting the pair (m, n). Then the set of spherical trajectories along which the k-space sampling takes place is:

$$S = \{\mathbf{u}_i = (\theta_m, \phi_n^m), i = 0, \ldots, N_s - 1\} \tag{4.5}$$

where $N_s = \sum_{m=0}^{N_\theta - 1} N_\phi^m$ is the total number of spherical trajectories and

$$\theta_m = m\Delta\theta, \quad \Delta\theta = \pi/(N_\theta - 1) \quad m = 0, \ldots, N_\theta - 1 \tag{4.6}$$

$$\phi_n^m = n\Delta\phi_m, \Delta\phi_m = 2\pi/N_\phi^m \quad n = 0, \ldots, N_\phi^m - 1 \tag{4.7}$$

Figure 4.1 displays the projection ray endpoints of a side and a top view of the k-space sampling for $N_\theta = 51$.

4.2.2 Modifications to the spherical sampling

For reasons that we now proceed to explain, we make the following two modifications to the spherical sampling:

- Sampling along central rays: In order to have Radon projections of the object, the k-space must be sampled along straight lines passing through the origin. That is, each readout projection in the northern hemisphere must line up with a readout projection

in the southern hemisphere. The sampling method in [9] does not guarantee this property.

- N_ϕ^m multiple of four and N_θ odd: By restricting N_ϕ^m to be a multiple of four for all m and N_θ to be an odd number, we introduce geometrical systems that later on will be exploited to reduce the computational requirements of the image reconstruction. Exceptions to the multiple of four restriction are the north and south poles, where there is only one projection, respectively.

It can be seen that the latter constraint naturally covers the former. First, observe that if $N_\theta = 2L + 1$ for some integer L and $\mathbf{u}_i = (\theta_m, \phi_n^m)$, then from (4.6)

$$\theta_m = m\frac{\pi}{2L}$$

and

$$\begin{aligned}\theta_{N_\theta - m - 1} &= (N_\theta - 1)\frac{\pi}{2L} - m\frac{\pi}{2L} \\ &= \pi - \theta_m.\end{aligned}$$

Therefore, there exists an $m' = N_\theta - m - 1$ such that $\mathbf{u}_j = (\theta_{m'}, \phi_n^{m'})$ and $\mathbf{u}_j = T(\mathbf{u}_i)$.

Similarly, if $N_\phi^m = 4K$ for some integer K, then from (4.7)

$$\phi_n^m = n\frac{2\pi}{4K}$$

and

$$\phi_{n+N_\phi^m/4}^m = \phi_n^m + \pi$$

In this case we have $n' = n + N_\phi^m/4$, $\mathbf{u}_k = (\theta_m, \phi_{n'}^m)$, and $\mathbf{u}_k = R_\pi \mathbf{u}_i$. Combining these two transformations shows that for any $\mathbf{u}_i$ there exists a $\mathbf{u}_{i'}$ such that $\mathbf{u}_{i'} = TR_\pi \mathbf{u}_i$. Thus

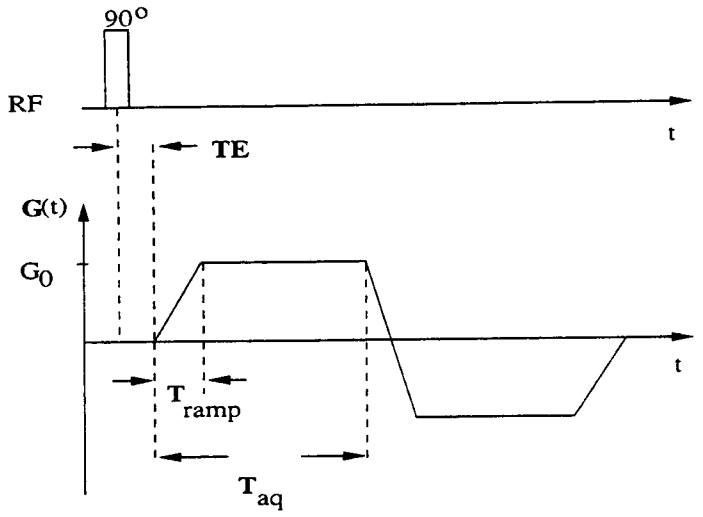


Figure 4.2: Simplified pulse sequence and projection readout gradient for boron imaging.

the signal sampled along $\mathbf{u}_{i'}$ is the "continuation" of the signal sampled along $\mathbf{u}_i$ in the opposite side of the k-space which is the desired property.

4.2.3 The readout signal

Consider the simplified pulse sequence for the three-dimensional projection sampling shown in Figure 4.2. G_0 is the magnitude of the projection readout gradient, T_{ramp} is the time needed for the gradients to stabilize, T_{aq} is the acquisition time, and TE is the echo time. According to Equation (2.1) the maximum sampled frequency corresponds to:

$$k_{\text{max}} = \gamma G_0 (T_{\text{aq}} - T_{\text{ramp}}/2)$$

where γ is the boron gyromagnetic ratio. Thus, if N_k is the number of points sampled, the sampling interval in k-space is $\Delta k = k_{\text{max}}/(N_k - 1)$. The projection data collected is therefore the set $\{S(\mathbf{u}_i, l\Delta k)\}$ for $i = 0, \ldots, N_s - 1$ and $l = 0, \ldots, N_k - 1$. The total number of data points sampled is $M = N_k N_s$.

4.2.4 Projection data

Having defined the sampling geometry, its necessary modifications, and the MRI sampling data, we are now in a position to describe how the object's Radon projection is obtained.

Let $\mathbf{u}_i$ and $\mathbf{u}_{i'} = TR_\pi \mathbf{u}_i$ be two spherical trajectories, for $i = 0, \ldots, N_s/2 - 1$, and let $P(\mathbf{r}_i, l\Delta_k)$, with $\mathbf{r}_i = \mathbf{u}_i$, denote the sequence of length $2N_k - 1$ given by

$$P(\mathbf{r}_i, l\Delta_k) = \begin{cases} S(\mathbf{u}_i, -l\Delta_k) & l = -(N_k/2-1), \ldots, 0 \\ S(\mathbf{u}_{i'}, l\Delta_k) & l = 1, \ldots, (N_k/2-1) \end{cases} \tag{4.8}$$

Then by the projection theorem (2.11) we have that the Radon transform of the object at direction $\mathbf{r}_i$ is the 1D inverse Fourier transform of P with respect to $l\Delta_k$, that is

$$p(\mathbf{r}_i, d\Delta_\tau) = [\mathcal{F}_1 P]^{-1}(d\Delta_\tau) \tag{4.9}$$

where $d = -(N_k/2-1), \ldots, -1, 0, 1, \ldots, (N_k/2-1)$ and $\Delta_\tau = 1/(2k_{\max})$. The set of direction vectors corresponding to Radon projections is the first half set of vectors from the set of spherical trajectories S defined in Equation (4.5), that is:

$$R = \{\mathbf{r}_i = (\theta_m, \phi_n^m), i = 0, \ldots, N_v - 1\} \tag{4.10}$$

where now

$$N'_\theta = (N_\theta - 1)/2 + 1 \tag{4.11}$$

$$N_v = N_s/2 \tag{4.12}$$

$$\theta_m = m\Delta\theta, \quad m = 0, \ldots, N'_\theta - 1 \tag{4.13}$$

$$\phi_n^m = n\Delta\phi_m \begin{cases} n = 0, \ldots, N_\phi^m - 1, & m = 0, \ldots, N'_\theta - 2 \\ n = 0, \ldots, N_\phi^m/2 - 1, & m = N'_\theta - 1 \end{cases} \tag{4.14}$$

and $\Delta\theta$ and $\Delta\phi_n^m$ are the same as for the spherical directions.

4.3 The discrete formulation

Let indices i and j refer to a specific projection plane $(\theta_m, \phi_n^m, d\Delta_\tau)$ and blob ψ_j, respectively. Then a discrete model of the image formation process given by Equation (3.3) can be written as:

$$\mathbf{p} = \mathbf{A}\,\mathbf{c} \tag{4.15}$$

where

$$\begin{aligned}
&\mathbf{A} \equiv (a_{ij}) \in \mathbb{R}^{M\times N}, \quad a_{ij} = \Psi_j(\theta_m, \phi_n^m, d\Delta_\tau) \\
&\mathbf{p} = [p_0, \ldots, p_{M-1}]^T, \quad p_i = p(\theta_m, \phi_n^m, d\Delta_\tau) \\
&\mathbf{c} = [c_0, \ldots, c_{N-1}]^T.
\end{aligned}$$

$\mathbf{A}$ is the system matrix whose elements a_{ij} are the Radon transform of the blobs at different positions, $\mathbf{p}$ is the projection data vector and $\mathbf{c}$ is the coefficient vector whose (unknown) elements c_j produce the image through Equation (3.1).

So far, it has been assumed that the projection data correspond to ideal Radon projections, that is integrals along planes of zero width. More realistically, we can assume that these projections have a width of $2\Delta\tau$ which leads to

$$a_{ij} = \int_{(d-1)\Delta_\tau}^{(d+1)\Delta_\tau} \Psi_j(\theta_m, \phi_n^m, \tau)\, d\tau. \tag{4.16}$$

The Bessel nature of the function being integrated means that the slab integral does not have a closed form solution and, consequently, it must be computed numerically.

By modeling the problem as a set of linear equations we make the problem independent of the scanning process. Most importantly, the problem is now in the framework of pseudoinverse, maximum entropy, and other techniques used in restoration problems. Specially attractive are iterative algorithms, because: (1) they allow for the incorpora-

tion of prior knowledge about the class of feasible solutions; (2) they can be applied even when complete projection data is unavailable; and (3) there is considerable control over the reconstruction process. Some disadvantages are: (1) they require large amount of computation time and storage; and (2) the stopping criterion is not well defined.

4.4 Geometrical symmetries

Matrix **A** constitutes a significant computational problem. For instance, the boron application we might use 1,645 views each consisting of 64 planar projections, and an image volume of $64 \times 64 \times 64$ voxels. This results in **A** having on the order of 28 billion elements. Even when considering only the approximately 800 million nonzero elements, this exceeds the memory capacity of most computers. In the same spirit as in [20, 5], we therefore make use of geometrical symmetries to reduce the memory requirements. Ultimately, this allows us to embed the problem on a small network of regular workstations.

Recall that N_ϕ^m is divisible by four for each value of θ_m. This leads to *rotation symmetries* as follows. Let $\mathbf{r} = (\theta_m, \phi_n^m)$ be a particular projection view and let $\mathbf{A_r}$ denote the block of rows in **A** that contain all the elements associated with view **r**. Then,

$$\mathbf{A_r} \equiv \mathbf{A}_{R_{\pi/2}\mathbf{r}} \equiv \mathbf{A}_{R_\pi \mathbf{r}} \equiv \mathbf{A}_{R_{3\pi/2}\mathbf{r}} \tag{4.17}$$

in the sense that for all $a_{i'j'} \in \mathbf{A}_{R_{k\pi/2}\mathbf{r}}$, for $k = 1, 2, 3$, there exists an $a_{ij} \in \mathbf{A_r}$, such that $a_{i'j'} = a_{ij}$. Figures 4.3 (a) and (b) illustrate the relation between an **r** view (black circle) and $R_{\pi/2}\mathbf{r}$, $R_\pi \mathbf{r}$, and $R_{3\pi/2}\mathbf{r}$ views (open circles). By storing only $\mathbf{A_r}$ we obtain a memory reduction factor of four.

We also exploit the *mirror symmetry*, as shown in Figure 4.3 (c). This symmetry consists of partitioning $\mathbf{A_r}$ in two equivalent submatrices

$$\mathbf{A_r} = \begin{bmatrix} \mathbf{A_r^-} \\ \mathbf{A_r^+} \end{bmatrix}, \quad \mathbf{A_r^-} \equiv \mathbf{A_r^+} \tag{4.18}$$

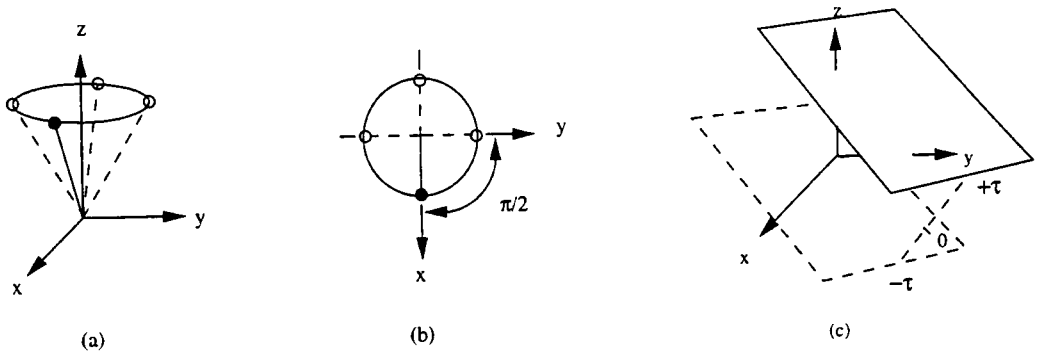


Figure 4.3: Diagram of geometrical symmetries. (a) Rotation symmetry for the half-space $z > 0$ and (b) as seen from the z-axis. (c) Mirroring symmetry.

where $\mathbf{A}_\mathbf{r}^-$ and $\mathbf{A}_\mathbf{r}^+$ correspond to projections in the negative and positive halfspaces with respect to the origin, respectively. Equivalence is defined in the sense that for all $a_{i'j'} \in \mathbf{A}_\mathbf{r}^-$, there exists an $a_{ij} \in \mathbf{A}_\mathbf{r}^+$, such that $a_{i'j'} = a_{ij}$. By storing only either $\mathbf{A}_\mathbf{r}^-$ or $\mathbf{A}_\mathbf{r}^+$ we obtain an additional memory reduction factor of two and thus an overall memory reduction factor of eight.

Therefore, the entries that need to be stored in $\mathbf{A}$ are only those associated with projection views $\mathbf{r} = (\theta_m, \phi_n^m)$ in the first octant of the system geometry, that is the set

$$\mathcal{P} = \{\mathbf{r} = (\theta_m, \phi_n^m), 0 \leq \theta_m < \frac{\pi}{2}, 0 \leq \phi_n^m \leq \frac{\pi}{2}\} \tag{4.19}$$

Notice that the north pole projection, $\theta = 0$, does not have rotations, and that projections in the xy plane have only a $\pi/2$ rotation.

Figure 4.4 shows an schematic representation of how matrix $\mathbf{A}$ is partitioned in blocks representing different projection views. The dark shaded areas are blocks of $\mathbf{A}$ that are actually stored and subsequently used in matrix-vector multiplications to produce their own results and as well as the results for blocks not stored, shown here as light shaded areas.

In order to demonstrate the significant memory reductions for storing a sparse matrix such as $\mathbf{A}$ based on symmetries, we briefly describe the data structure used to store it. With reference to Figure 4.5, the sparse matrix data representation consists of four tables.

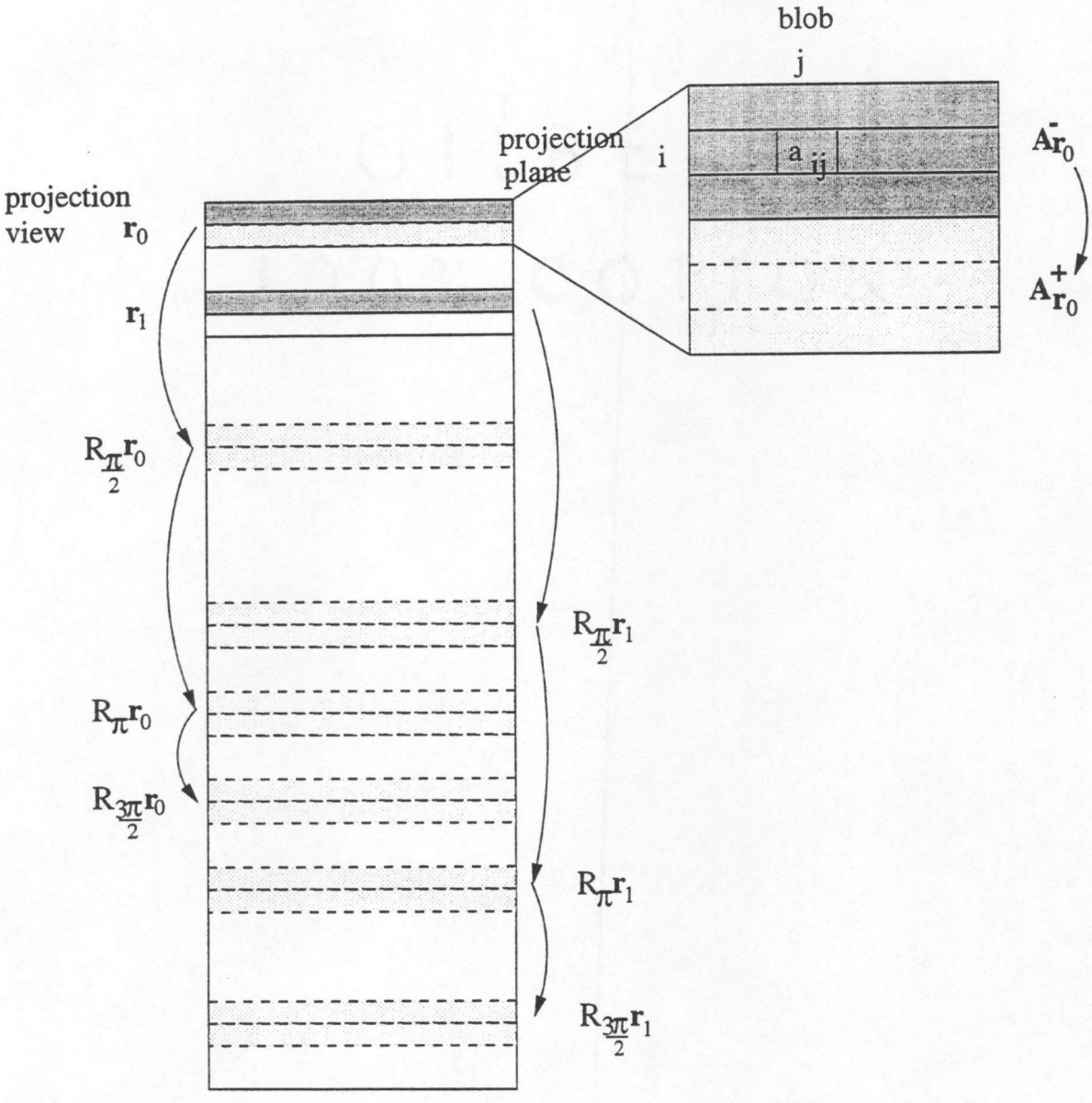


Figure 4.4: Schematic representation of the memory savings obtained for matrix **A** based on rotation and mirroring operations.

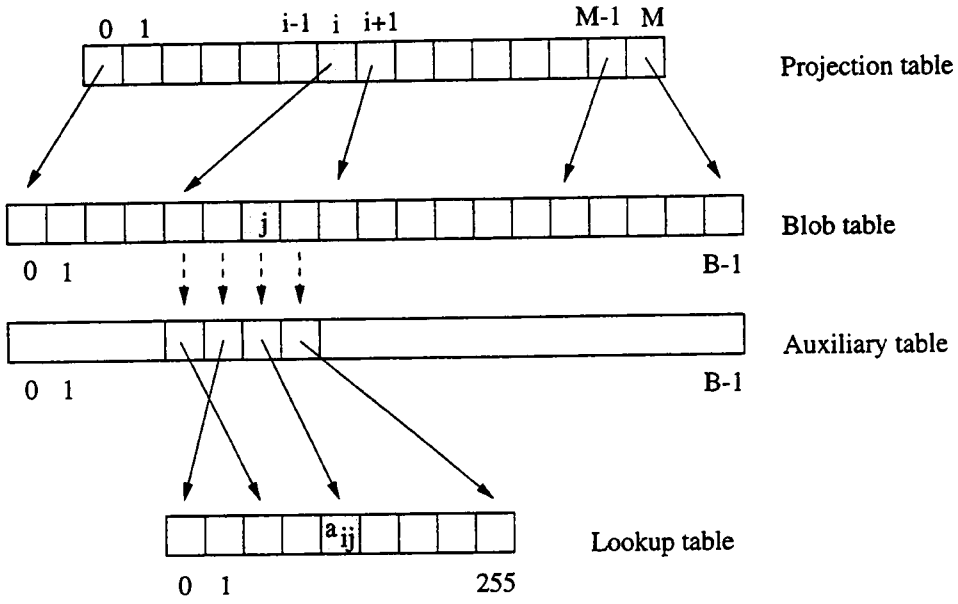


Figure 4.5: Sparse matrix data representation.

Entry i in the projection table represents a specific projection plane $i = (\theta_m, \phi_n^m, d\Delta_\tau)$, and through it, we access the blob table which stores the indices of the blobs, e.g. j, that are intersected by projection plane i. The length of the projection table is $M + 1$, but the length of the blob table is unknown and an estimate thereof must be computed before building $\mathbf{A}$. Both the projection and blob table entries are represented with integers. Instead of storing $a_{ij} = \Psi(\theta_m, \phi_m^n, d\Delta_\tau)$ explicitly for each pair (i, j), we have an auxiliary table with indices to a lookup table that stores Ψ values for only 256 values of $d\Delta_\tau$. This allows the entries of the auxiliary table to be represented by bytes instead of single or double precision numbers. Assuming the length of the blob table is B and that the lookup table entries are in double-precision, then the number of bytes needed to store $\mathbf{A}$ is:

$$\text{bytes} = 4(M + 1 + B) + B + 8 \times 256 \simeq 5B.$$

Table 4.1 shows the number of non-zero elements stored and the memory requirements when using and not using symmetries for various blob radii, fixed α, $N_\theta = 51$, $N_k = 64$, and $N = 64 \times 64 \times 64 = 262,144$ blobs. The distance between blob centers is assumed to

Table 4.1: Number of nonzero entries and memory requirements in gigabytes for various blob radius with and without symmetries.

Blob radius	*Symmetries off*		*Symmetries on*	
	No. nonzero elements (10^6)	Memory (Gb)	No. nonzero elements (10^6)	Memory (Gb)
1.0	1,195	5.57	164	0.77
1.2	1,353	6.30	186	0.87
1.4	1,510	7.04	207	0.97
1.6	1,670	7.78	229	1.07
1.8	1,827	8.51	251	1.17
2.0	1,984	9.24	272	1.27

be 1.0. On average, we find that the memory savings using symmetries is of the order of 86%.

4.5 Iterative algorithms

4.5.1 Least squares

In this section we consider the least square solution

$$\mathbf{c}^\star = \underset{\mathbf{c}}{\operatorname{argmin}} \ \|\mathbf{A}\mathbf{c} - \mathbf{p}\| \tag{4.20}$$

to the under-determined system of equations (4.15).

The most reliable algorithms for solving (4.20) are based on reducing $\mathbf{A}$ to a canonical form via orthogonal transformations [11]. Specially attractive is Paige and Saunders's LSQR algorithm [32] which is based on the bidiagonalization procedure of Golub and Kahan [10].

If $\mathbf{A}$ is of full rank, LSQR finds the unique minimum norm solution among all solutions of $\mathbf{A}\mathbf{c} = \mathbf{p}$. When $\mathbf{A}$ is under-determined, the solution satisfies the normal equations of

the second kind:

$$\mathbf{A}\mathbf{A}^T\mathbf{z} = \mathbf{p} \tag{4.21}$$

where $\mathbf{c} = \mathbf{A}^T\mathbf{z}$. Thus, at convergence, LSQR is equivalent to a conjugate gradient (CG) algorithm applied to solve (4.21). Furthermore, if $\mathbf{A}$ is rank deficient, LSQR will solve (4.20) whereas the conjugate gradient will not. One disadvantage of LSQR and CG algorithms is that they do not necessarily yield nonnegative solutions if the iterations are not explicitly constrained to do so. Some strategies to guarantee non-negative solutions include (1) active set of variables, (2) bent line search, and (3) interior point methods [21]. Although the nonnegativity assumption is natural for the BNCT application, since we cannot have a negative concentration of boron anywhere in the body, we do not to include them here for the sake of simplicity.

At iteration k, LSQR approximates matrix $\mathbf{A}$ by a lower bidiagonal matrix $\mathbf{B}_k$, which is used to formulate a (smaller) least-squares subproblem. Specifically, the quantities computed are:

$$\mathbf{V}_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k] \in \mathbb{R}^{N\times k}, \quad \mathbf{U}_{k+1} = [\mathbf{u}_1, \ldots, \mathbf{u}_{k+1}] \in \mathbb{R}^{M\times(k+1)} \tag{4.22}$$

and

$$\mathbf{B}_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix} \in \mathbb{R}^{(k+1)\times k} \tag{4.23}$$

such that $\mathbf{U}_{k+1}^T\mathbf{U}_{k+1} = \mathbf{I}$, $\mathbf{V}_k\mathbf{V}_k^T = \mathbf{I}$, $\mathbf{A}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{B}_k$, $\beta_1\mathbf{U}_{k+1}\mathbf{e}_1 = \mathbf{b}$, and $\mathbf{A}^T\mathbf{U}_{k+1} = \alpha_{k+1}\mathbf{v}_{k+1}\rho_{k+1}^T + \mathbf{V}_k\mathbf{B}_k^T$. The kth approximation to $\mathbf{c}^*$ is given by $\mathbf{c}_k = \mathbf{V}_k\mathbf{y}_k$ where

$\mathbf{y}_k \in \mathbb{R}^k$ is the solution to the least square subproblem

$$\mathbf{y}_k = \underset{\mathbf{y}\in\mathbb{R}^k}{\operatorname{argmin}} \|\mathbf{B}_k\mathbf{y} - \beta_1\mathbf{e}_1\| \tag{4.24}$$

for $\beta_1 = \|\mathbf{p}\|$ and $\mathbf{e}_1 = [1, 0, \ldots, 0]^T \in \mathbb{R}^{k+1}$. The LSQR algorithm solves this subproblem by computing the QR decomposition of $\mathbf{B}_k$ which can be accumulated as a product of Givens rotations. Paige and Saunders show that several simplifications can be made, perhaps most importantly allowing the approximation $\mathbf{c}_k$ to be found recursively thus eliminating the need for storing the entire matrix $\mathbf{V}_k$.

Figure 4.6 outlines the LSQR algorithm. Steps (a)-(c) initialize the bidiagonalization process. At each iteration, steps (d)-(e) continue the bidiagonalization, while steps (f)-(i) eliminate the subdiagonal of $\mathbf{B}_k$ by applying Givens rotations and compute several quantities that are used to update the solution $\mathbf{c}_k$ in step (j). Finally, (k) updates the "search" direction. The scalars β_{k+1} and α_{k+1} in steps (b), (d) and (e) are chosen such that $\mathbf{u}_{k+1}$ and $\mathbf{v}_{k+1}$ are unit vectors.

4.5.2 Maximum likelihood

A conceptually different way of solving (4.15) is the Richardson-Lucy maximum likelihood (RL-ML) algorithm [35, 27] which models the problem from a statistical point of view. The input data and the image are assumed to be random variables with known density functions and the objective is to maximize the log-likelihood of observing the data given the current estimate of the solution, that is

$$\mathbf{c}^* = \underset{\mathbf{c}}{\operatorname{argmax}} \log P(\mathbf{p}|\mathbf{c}) \tag{4.25}$$

where P is the likelihood function. Two important properties of the maximum-likelihood (RL-ML) estimate implemented by the RL-ML scheme are preservation of mass between iterations, i.e $\sum_i \langle \mathbf{a}^i, \mathbf{c}^k \rangle = \sum_i p_i$, and nonnegativity in the solution. However, the RL-ML has slower convergence rate than the LSQR and CG algorithms [21]. The algorithm, shown

Algorithm LSQR (Paige and Saunders):

1. Initialize

(a) $\mathbf{c}_0 = 0;$

(b) $\beta_1 \mathbf{u}_1 = \mathbf{p}; \quad \alpha_1 \mathbf{v}_1 = \mathbf{A}^T \mathbf{u}_1; \quad \mathbf{w}_1 = \mathbf{v}_1;$

(c) $\bar{\phi}_1 = \beta_1; \quad \bar{\rho}_1 = \alpha_1;$

2. for $k = 1, 2, \ldots$ until convergence

(d) $\beta_{k+1} \mathbf{u}_{k+1} = \mathbf{A}\mathbf{v}_k - \alpha_k \mathbf{u}_k;$

(e) $\alpha_{k+1} \mathbf{v}_{k+1} = \mathbf{A}^T \mathbf{u}_{k+1} - \beta_{k+1} \mathbf{v}_k;$

(f) $\rho_k = \sqrt{(\bar{\rho}_k)^2 + (\beta_{k+1})^2};$

(g) $z_k = \bar{\rho}_k / \rho_k; \quad s_k = \beta_{k+1} / \rho_k;$

(h) $\theta_{k+1} = s_k \alpha_{k+1}; \quad \bar{\rho}_{k+1} = -z_k \alpha_{k+1};$

(i) $\phi_k = z_k \bar{\phi}_k; \quad \bar{\phi}_{k+1} = s_k \bar{\phi}_k;$

(j) $\mathbf{c}_k = \mathbf{c}_{k-1} + (\phi_k / \rho_k) \mathbf{w}_k;$

(k) $\mathbf{w}_{k+1} = \mathbf{v}_{k+1} - (\theta_{k+1} / \rho_k); \mathbf{w}_k$

Figure 4.6: The LSQR algorithm.

Algorithm RL-ML (Richardson-Lucy):

1. Initialize

$$\boxed{\begin{array}{l}\text{(a)}\ \ \mathbf{c}^0 = \mathbf{1}; \\ \text{(b)}\ \ \mathbf{t} = \mathbf{A}^T\mathbf{1};\end{array}}$$

2. for $k = 1, 2, \ldots$ until convergence

$$\boxed{\begin{array}{l}\text{(c)}\ \ \mathbf{q}_k = \mathbf{A}\mathbf{c}_k; \\ \text{(d)}\ \ \hat{\mathbf{q}}_k = \mathbf{p} \oslash \mathbf{q}_k; \\ \text{(e)}\ \ \hat{\mathbf{c}}_k = \mathbf{A}^T\hat{\mathbf{q}}_k; \\ \text{(f)}\ \ \mathbf{c}_{k+1} = (\mathbf{c}_k \otimes \hat{\mathbf{c}}_k) \oslash \mathbf{t};\end{array}}$$

Figure 4.7: The RL-ML algorithm.

in Figure 4.7, consists of a sequence of so-called forward and backward projections from the image domain to the projection domain and vice versa, respectively. The solution estimate is initialized with a positive value and, at each iteration, updated in a multiplicative fashion.

The operators $\oslash$ and $\otimes$ denote elementwise division and multiplication, respectively. Step (a) initializes the solution estimate with a positive value and step (b) computes a normalization vector used later in the computation. Step (c) forward projects the current estimate of the image, $\mathbf{c}_k$, to compute an estimate $\mathbf{q}_k$ of the projection data. The elementwise ratio of projection data $\mathbf{p}$ and estimate $\mathbf{q}_k$ is computed in step (d) and then backprojected into $\hat{\mathbf{c}}_k$ in step (e). Finally, step (f) updates the solution with $\hat{\mathbf{c}}_k$ and $\mathbf{t}$.

In closing, we finally mention that the RL-ML algorithm is identical in form to the expectation maximization maximum likelihood (EM-ML) algorithm [37, 22] which is used extensively in SPECT/PET tomography.

4.6 Distributed computation

An efficient implementation of any of the iterative algorithms described in the previous section for three-dimensional image reconstruction requires the use of distributed or parallel computation. As indicated by Table 4.1, the number of non-zero entries in system matrix $\mathbf{A}$ is of the order of two billion elements for a blob radius of 2, even for a rather small problem. This translates into approximately 1.3 gigabytes of memory and 4 billion floating-point operations per iteration. Thus, even if we could fit this matrix in a single processor's memory, the execution time of the algorithm would be prohibitive with today's technology. We resort, therefore, to a distributed solution.

The parallelism in the algorithms can be embedded entirely in the matrix-vector multiplication operations $\mathbf{y} = \mathbf{A}\mathbf{x}$ and $\mathbf{x} = \mathbf{A}^T\mathbf{y}$. All other storage and overhead considerations are negligible when compared to the vast amount of computation and memory that these two operations demand. In the image reconstruction literature, these operations are usually called *forward* and *backward* projections because they correspond to mapping the image into the projection domain and vice versa, respectively.

In order to distribute the forward and backward operations, processor i is assigned a subset $\mathcal{P}_i = \{\mathbf{r}_{i_0}, \mathbf{r}_{i_1}, \ldots, \mathbf{r}_{i_{n-1}}\}$ of $\mathcal{P}$ from Equation (4.19). We call $\mathcal{P}_i$ a "generating" set in the sense that from it, other projections are accessed by rotation and mirroring operations. The corresponding blocks of matrix $\mathbf{A}$ that processor i stores are the generating matrices $\{\mathbf{A}^-_{\mathbf{r}_{i_0}}, \cdots, \mathbf{A}^-_{\mathbf{r}_{n-1}}\}$. By distributing the problem in this way, processors have a full view of the image but only a partial view of the projection data. This means that the forward projection of a single processor is complete and does not need data from other processors. On the other hand, a backward projection of a processor is partial and a global sum reduction is needed to compute the final image.

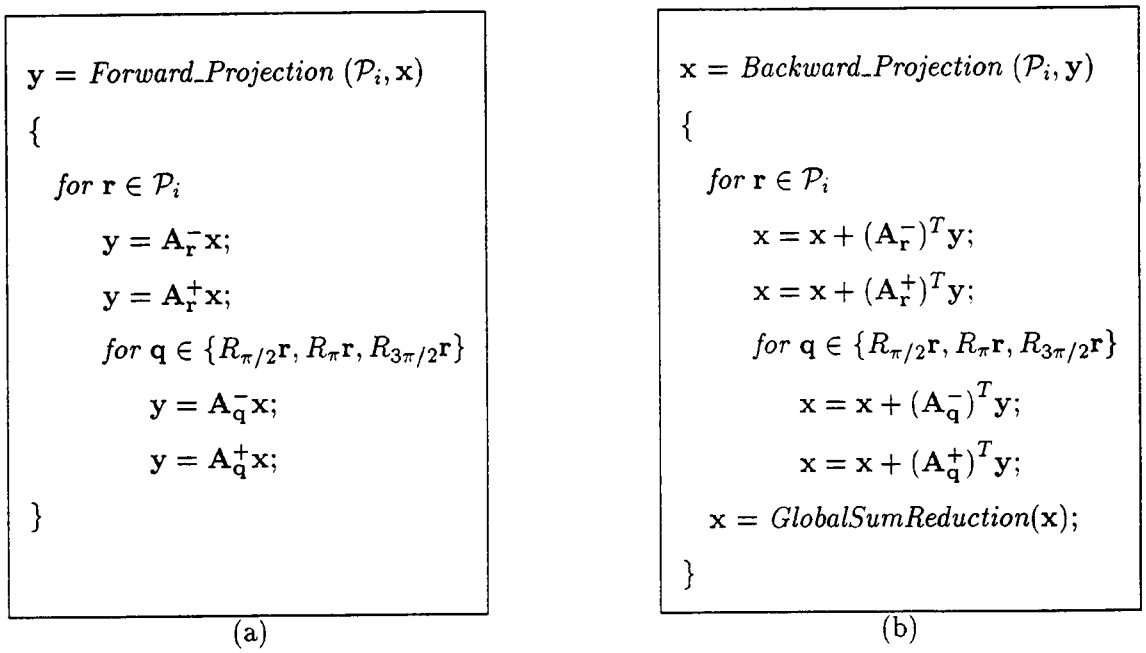


Figure 4.8: Forward and backward projection procedures for a symmetry based system matrix.

4.6.1 Forward and backward projections

The symmetry based *Forward* and *Backward* procedures are shown in Figure 4.8 (a) and (b), respectively. Both algorithms start by computing the matrix-vector multiply corresponding to direction $\mathbf{r}$ which always includes a mirroring operation. Then, the matrix-vector multiply for rotated and mirrored directions are carried out. If $\mathbf{r}$ is one of the poles then $R_{n\pi/2} = \varnothing$, for any n, and if $\mathbf{r}$ is in the xy plane then $R_\pi\mathbf{r} = R_{3\pi/2}\mathbf{r} = \varnothing$. Besides the global reduction, the main difference between these two procedures is that the forward operation computes final results for entries in $\mathbf{y}$, whereas a backward computes only partial results. This is the result of partitioning $\mathbf{A}$ by projection planes (or rows), as opposed to blobs (or columns).

Matrices $\mathbf{A}_\mathbf{r}^+$, $\mathbf{A}_\mathbf{q}^+$ and $\mathbf{A}_\mathbf{q}^-$ are not stored. Their entries are efficiently accessed from $\mathbf{A}_\mathbf{r}^-$ by lookup-tables. No additional storage is needed for the various transpose matrices.

After a forward projection, LSQR must compute $\alpha^{k+1} = \langle \mathbf{u}^{k+1}, \mathbf{u}^{k+1} \rangle^{1/2}$ to normalize $\mathbf{u}^{k+1}$ in step (d). Each processor computes its contribution to the inner product followed

by a global sum operation across the nodes. Thus instead of communicating an entire vector, we only communicate a number. RL-ML does not require any communication after a forward projection.

4.6.2 Load balancing

One important aspect in the implementation of a distributed algorithm is to distribute the workload among processors as evenly as possible to minimize the idle time and consequently total execution time of the algorithm. In our case, the workload distribution is a function of how the reduced set of directions is assigned to individual processors.

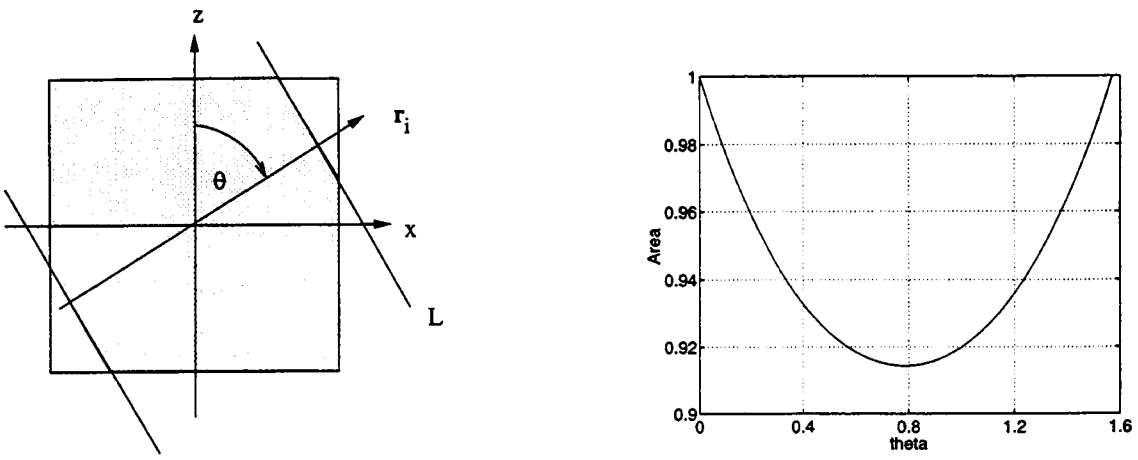
At first sight, it might appear that any workload assignment that distributes the $\mathbf{r}$'s evenly across the processors will produce a balanced workload because of the spherical nature of the system geometry. This is not the case, however, for the following two reasons:

1. Directions $\mathbf{r}_i = (\theta_m, \phi_n^m)$ in the xy plane, i.e. for which $\theta_m = \pi/2$, are rotated only once instead of three times.

2. The number of elements in $\mathbf{A}_{\mathbf{r}_i}^+$ (or $\mathbf{A}_{\mathbf{r}_i}^-$), and consequently the number of multiplications in a forward and a backward operation, is a function of where $\mathbf{r}_i$ is positioned in the geometry with respect to the image.

To see why the number of elements in $\mathbf{A}_{\mathbf{r}_i}^+$ is a function of θ and ϕ, consider the 2D perspective of the geometry when $\phi = 0$, for all θ, shown in Figure 4.9 (a). The square represents the image, line L represents the outermost projection plane at direction $\mathbf{r}_i$, and the shaded region represents the area of intersection between all projection planes at $\mathbf{r}_i$ and the image. It is seen that the area is a measure of the workload or number of floating-point operations associated with $\mathbf{r}_i$. The area of intersection as function of θ is:

$$A(\theta) = \Big(\frac{\sin\theta\cos\theta + \sin\theta + \cos\theta - 1}{2\sin\theta\cos\theta}\Big). \tag{4.26}$$

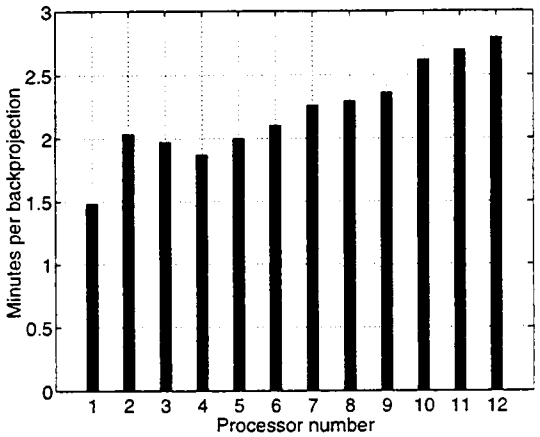


(a) (b)

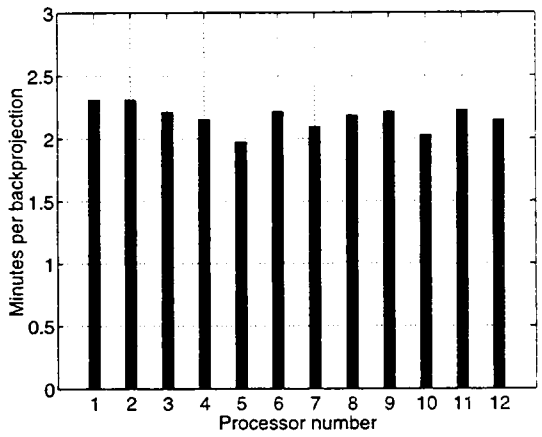
Figure 4.9: Description of the dependency of the area on θ: (a) area of intersection between image and all projection planes at $\mathbf{r}_i$, (b) plot of the area as a function of θ.

Figure 4.9 (b) plots $A(\theta)$ for $0 \leq \theta \leq \pi/2$. The shape of the function for fixed θ and varying ϕ is similar, that is it is maximum at $\phi = 0$ and $\phi = \pi/2$ and minimum at $\phi = \pi/4$. It can be shown that $\lim_{\theta \to 0} = \lim_{\theta \to \pi/2} = 1$.

A raster distribution of projections from $\theta = 0$ to $\theta = \pi/2$ and for each θ from $\phi = 0°$ to $\phi = \pi/2$ would assign a processor only projections that are close to each other and therefore enclosing similar (either a small or large) areas. Thus two processors may be assigned very different workloads depending on which projections they are assigned during the raster distribution. To overcome this problem, the following heuristic assignment is used. First, the projections are sorted in ascending order according to Equation (4.26), and then a processor is assigned an equal number of projections from the higher and lower ends of this ordered set. Also, the assignment of projections at $\theta = \pi/2$ is treated similarly, but the projections are sorted as function of ϕ instead. Figure 4.10 shows the wall-clock time needed by a backward operation for the raster and heuristic assignments. There are approximately 0.5 minutes saved per iteration using the heuristic method. For 60 iterations this translates into a 30 minutes saving for RL-ML and 1 hour for LSQR, which is considerable.



(a)



(b)

Figure 4.10: Wall-clock time needed by a backward operation for two workload distributions: (a) raster, (b) heuristic.

Chapter 5

Three-Dimensional Focus of Attention

5.1 Introduction and overview

In this chapter we follow the approach of Focus-Of-Attention (FOA) [14, 13] and thus the theory of support functions [33, 19] to introduce a basic form of *prior* knowledge into the reconstruction.

The basic idea is to introduce a spatial constraint for the object to be reconstructed. Since the object must lie inside a closed region in the image, it is natural to assume that all projection planes that have a zero projection value do not intersect this region and therefore can be discarded from the reconstruction. Likewise, all blobs outside this region will have no spin density to contribute to projection data and can also be eliminated. Thus, FOA effectively eliminates unknowns and equations from the linear system given by Equation (4.15). A smaller system of equations implies a reduced amount of storage, computation, and communication time.

The "boundary" projection planes at a given view, i.e. planes beyond which there is no activity, are formally called support planes. The distance from the origin to a support plane is called a support measurement and the set of all support measurements is called a

support vector. Support planes define a mask that indicates which projection planes are to be considered during the reconstruction. They also define the convex hull of the object which in our case corresponds to the smallest convex object containing all the blobs with non zero spin density. The FOA strategy then consists of the following steps:

1. Determine a consistent support vector and a set of support planes, or **p**-mask, that indicates the set of projection planes to focus on.

2. Compute the convex hull, or **b**-mask, that indicates the set of blobs to focus on.

3. Compute the **A** matrix for the reduced set of projection planes and blobs.

For ideal data, step 1 is easily accomplished by inspection. However, due to the presence of noise in the data, the support measurements are usually inconsistent with one another, in the sense that they do not correspond to a convex hull. To overcome this problem, we apply some important results from theory of support functions to enforce consistency among the support planes. An initial estimate of the true support vector is manipulated by an optimization process such that, at termination, all support planes are consistent with one another.

In section 5.2 we briefly review the theory of support functions and consistency conditions as found in [19]. In section 5.3 we describe how this theory is applied to our 3D reconstruction problem with special emphasis on computational issues that render a feasible solution. As in chapter 4, we make use of symmetries and distributed computing. In section 5.4 we assess the computational impact that FOA introduces to iterative reconstruction. The theoretical results from [19] have not been previously applied to three or more dimensions because of the demanding computational requirements [18].

5.2 Support functions

5.2.1 Support functions and support hyperplanes

The support function $h(\mathbf{u}) \in \mathbb{R}$ of an object $\mathcal{O} \subset \mathbb{R}^n$ is given by:

$$h(\mathbf{u}) \equiv \sup_{\mathbf{x} \in \mathcal{O}} \mathbf{x}^T \mathbf{u} \tag{5.1}$$

where $\mathbf{u} \in \mathbb{R}^n$. The support hyperplane of $\mathcal{O}$ at direction $\mathbf{u}$ is given by the set of points

$$L(\mathbf{u}) = \left\{ \mathbf{x} \in \mathbb{R}^n | \mathbf{x}^T \mathbf{u} = h(\mathbf{u}) \right\}. \tag{5.2}$$

The inner product in Equation (5.1) represents the distance from $\mathbf{x}$ to the hyperplane passing through the origin and orthogonal to $\mathbf{u}$. Thus a support function measures the maximum projection of any point in $\mathcal{O}$ onto the $\mathbf{u}$-axis.

In two dimensions the support hyperplane is a support line as shown in Figure 5.1 (a). The corresponding support function is shown in Figure 5.1 (b). The unit direction vectors can be parameterized according to the angle θ they form with the x-axis. Notice that $L(\mathbf{u})$ is the tangent to the object and orthogonal to unit vector $\mathbf{u}$. Any other line at the same direction will either contain interior points from the object or none. Then the support function $h(\mathbf{u})$ of an object measures the distance from the origin to support lines at all directions.

Notice that according to Equation (5.1) an object $\mathcal{O}$ has a unique support function, but a function can be the support function for many different objects. It is also true that an object $\mathcal{O}$ has a unique convex hull. On the other hand, a support function $h(\mathbf{u})$ uniquely defines a convex hull which is the same for all the objects having $h(\mathbf{u})$ as support function. Therefore, given a support function we can only reconstruct the convex hull of the object for which it is the support function. The actual object boundary cannot be determined.

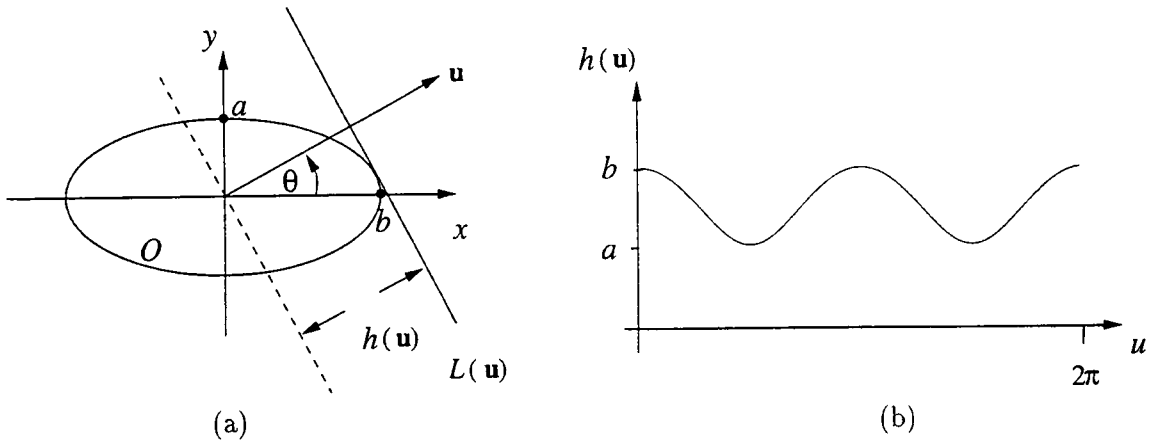


Figure 5.1: Example of an object and its support function. (a) object and support geometry, (b) support function.

5.2.2 Consistency conditions

The conditions under which a function is a support function are now characterized. A function $h(\mathbf{u})$ is the support function of a convex object if and only if it is defined for all vectors $\mathbf{u}$ and satisfies:

1. $h(0) = 0$
2. $h(\lambda\mathbf{u}) = \lambda h(\mathbf{u})$ for $\lambda > 0$
3. $h(\mathbf{u}+\mathbf{v}) \leq h(\mathbf{u}) + h(\mathbf{v})$, for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$.

These conditions are global in the sense that they must be satisfied for all vectors (infinite number) $\mathbf{u}$ and $\mathbf{v}$ which take values on a continuous range. In the two-dimensional case, Rademacher observes that the sub-additivity condition (3) can be replaced by a determinantal condition on $h(\mathbf{u})$. Karl et al [19] generalizes this result to n dimensions provided that one of the vectors is in the positive cone of the others. Specifically, they show that a function $h(\mathbf{u})$ is a support function if and only if it is defined for all $\mathbf{u}$, conditions (1) and (2) are satisfied and the following determinantal inequality is satisfied for all $(n+1)$-tuples of unit vector $\mathbf{u}_i$, with one in the full positive cone of the others:

$$\begin{vmatrix} h(\mathbf{u}_1) & \mathbf{u}_1^T \\ h(\mathbf{u}_2) & \mathbf{u}_2^T \\ \vdots & \vdots \\ h(\mathbf{u}_{n+1}) & \mathbf{u}_{n+1}^T \end{vmatrix} \begin{vmatrix} 1 & \mathbf{u}_1^T \\ 1 & \mathbf{u}_2^T \\ \vdots & \vdots \\ 1 & \mathbf{u}_{n+1}^T \end{vmatrix} \geq 0. \tag{5.3}$$

A vector $\mathbf{u}$ is in the positive cone formed by $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$, if it can be written as a linear combination of the $\mathbf{u}_i$'s with nonnegative coefficients, that is:

$$\mathbf{u} \in \text{cone}^+\{\mathbf{u}_1, \ldots, \mathbf{u}_n\} \Leftrightarrow \mathbf{u} = \sum_{i=1}^{n} c_i \mathbf{u}_i$$

with $c_i \geq 0$. A cone is said to be full if the set $\{\mathbf{u}_i\}$ is independent.

Condition (5.3) does not require any special ordering of the vectors. In fact, they can always be arranged so that $\mathbf{u}_{n+1}$ is in the positive cone of the remaining vectors, in which case the determinantal condition can be written as:

$$h(\mathbf{u}_{n+1}) - \mathbf{u}_{n+1}^T \begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \vdots \\ \mathbf{u}_n^T \end{bmatrix}^{-1} \begin{bmatrix} h(\mathbf{u}_1) \\ h(\mathbf{u}_2) \\ \vdots \\ h(\mathbf{u}_n) \end{bmatrix} \leq 0. \tag{5.4}$$

The left hand-side of Equation (5.4) is the signed distance, in the direction of $\mathbf{u}_{n+1}$, from the point determined by the intersection among the supporting hyperplanes given by normal $\mathbf{u}_i$, $i = 1, \ldots, n$, to the supporting hyperplane given by normal $\mathbf{u}_{n+1}$. Figure 5.2 illustrates these ideas for the three-dimensional case. In this example, $\mathbf{u}_4$ is assumed to be in the positive cone$^+\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$. In (a), the signed distance D from the point of intersection among the planes with normals $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ to the plane with normal $\mathbf{u}_4$ is negative, which implies that the set of support values for the 4-tuple $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\}$ is consistent and therefore $h(\mathbf{u})$ is a support function. In (b), D is positive which means

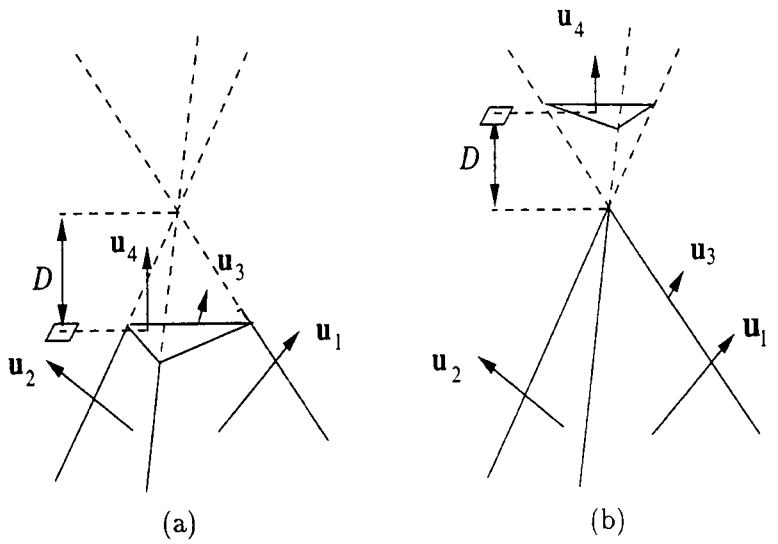


Figure 5.2: Geometrical interpretation of the determinantal condition in 3D: (a) condition is satisfied, (b) condition is not satisfied. (Adapted from [19].)

that they are inconsistent and $h(\mathbf{u})$ is not a support function.

It is worth emphasizing the fact that the determinantal condition is still global in the sense that all possible $n+1$-tuples must satisfy it.

5.2.3 Support samples: the discrete case

Consistency from global tests

In practical applications one can only measure support values along a discrete set of directions $\{\mathbf{u}_i \in \mathbb{R}^n, i = 0, \ldots, N_s - 1\}$. Due to the presence of noise, a group of such measurements may not be consistent which means that there is no object from which all such observations could have been obtained. The problem is then to determine whether a discrete set of support values is consistent and if not, to enforce consistency.

Let $\{h(\mathbf{u}_i), i = 0, \ldots, N_s - 1\}$ be a set of support samples associated with directions $\mathbf{u}_i$. Assume that the intersection of all support half-spaces is nonempty and assume that for every positive $(n+1)$-tuple cone of directions the cone is full. Then the set of samples is consistent if and only if the determinantal condition (5.3) is satisfied for every positive $(n+1)$-tuple cone of the set. The determinantal condition is also called a test.

Although we now have a finite set of measurements, the condition is still global and all positive cones must be checked. This results in an intractable computational problem in three and more dimensions. In [19] it is shown that it is possible to determine global consistency from local consistency tests and therefore substantially reduce the complexity of the problem. In order to do so, we first introduce the concept of local family.

Local family and consistency from local tests

Given a set $\mathcal{S} = \{\mathbf{u}_i \in \mathbb{R}^n, i = 0, \ldots, N_s - 1\}$ of unit vectors and a member $\mathbf{u}_k$ from $\mathcal{S}$ the local family corresponding to $\mathbf{u}_k$ is the set of all $(n+1)$-tuples of vectors from $\mathcal{S}$ such that $\mathbf{u}_k$ is the only vector from $\mathcal{S}$ in their full positive cone. Formally, we define a local family as the set:

$$\mathcal{L}(\mathbf{u}_k) = \{\text{cone}^+_{k_1}, \ldots, \text{cone}^+_{k_p}\} \tag{5.5}$$

where $\mathbf{u}_k \in \text{cone}^+_{k_j}$.

For instance, consider the example shown in Figure 5.3 for the 3D case. In (a), the local family of $\mathbf{u}_4$ is $\mathcal{L}_4 = \{\text{cone}^+\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}, \text{cone}^+\{\mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}, \text{cone}^+\{\mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}\}$. Adding a new vector $\mathbf{u}_{11}$ as shown in (b) changes the local family to $\mathcal{L}_4 = \{\text{cone}^+\{\mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}, \text{cone}^+\{\mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}\}$. The local families for the other vectors are not shown here.

A local family $\mathcal{L}$ is said to be consistent for direction $\mathbf{u}_k$ if and only if all possible $(n+1)$-tuples from $\mathcal{L}$ containing $\mathbf{u}_k$ in the full positive cone of the $(n+1)$-tuple satisfy the determinantal condition (5.3). Thus, consistency of a local family is the consistency of each full positive cone in the family. For the example, in Figure 5.3 (a), $\mathcal{L}_4$ is consistent if and only if $\text{cone}^+\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4\}$, $\text{cone}^+\{\mathbf{u}_4, \mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}$, and $\text{cone}^+\{\mathbf{u}_4, \mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}$ are all consistent.

The final result states that a set of support samples $h(\mathbf{u}_i)$ with associated direction vectors $\mathbf{u}_i$ is (globally) consistent if and only if for each direction $\mathbf{u}_k$ all elements of the corresponding local family are consistent.

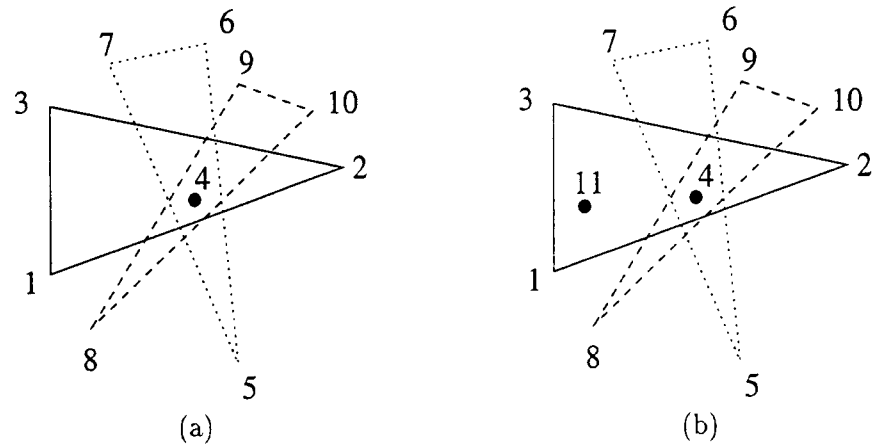


Figure 5.3: Local family example. In (a) $\mathcal{L}_4 = \{\text{cone}^+\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}, \text{cone}^+\{\mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}, \text{cone}^+\{\mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}\}$. The introduction of a new vector in (b) changes the family to $\mathcal{L}_4 = \{\text{cone}^+\{\mathbf{u}_5, \mathbf{u}_6, \mathbf{u}_7\}, \text{cone}^+\{\mathbf{u}_8, \mathbf{u}_9, \mathbf{u}_{10}\}\}$. (Adapted from [19].)

Summarizing, the steps taken to determine if a set of support samples $\{h(\mathbf{u}_i), i = 0, \ldots, N_s - 1\}$ is consistent are:

1. For each direction vector $\mathbf{u}_k$ find its local family $\mathcal{L}(\mathbf{u}_k)$.

2. For each cone in $\mathcal{L}(\mathbf{u}_k)$ determine whether the test (5.4) is satisfied.

If any cone in any local family does not satisfy the test, then the entire set of support measurements is not consistent.

Since each test in (5.4) is linear in the support samples $h(\mathbf{u}_i)$, the above procedure can be written in matrix form as:

$$\mathbf{Qh} \leq \mathbf{0} \tag{5.6}$$

where $\mathbf{Q} \in \mathbb{R}^{N_t \times N_s}$ is a highly sparse matrix with only $(n+1)$ non-zero entries in each row. N_t is the total number of tests, and N_s is the number of spherical directions in S. Equation (5.6) is the basis for constrained support reconstruction that we explore in the next section.

5.3 Application in 3D constrained reconstruction

5.3.1 Formulation of the problem

From the projection geometry defined in chapter 4 we have that the set of support directions is the set of spherical directions $S = \{\mathbf{u}_i, i = 0, \ldots, N_s - 1\}$ defined in Equation (4.5).

Since a support measurement at direction $\mathbf{u}_i$ can be understood as the largest possible projection of any point from the object onto the axis defined by $\mathbf{u}_i$, it is natural to use the projection data to compute an initial estimate of the support vector. From a Radon projection $p(\mathbf{r}_i, d\Delta_\tau)$ at direction $\mathbf{r}_i$ we obtain two support measurements, one for spherical direction $\mathbf{u}_i$ and the other for spherical direction $TR_\pi\mathbf{u}_i$ as shown in Figure 5.4. These values are computed by means of moving-average based thresholding on $p(\mathbf{r}_i, d\Delta_\tau)$ from left-to-right for $\mathbf{u}_i$ and from right-to-left for $TR_\pi\mathbf{u}_i$.

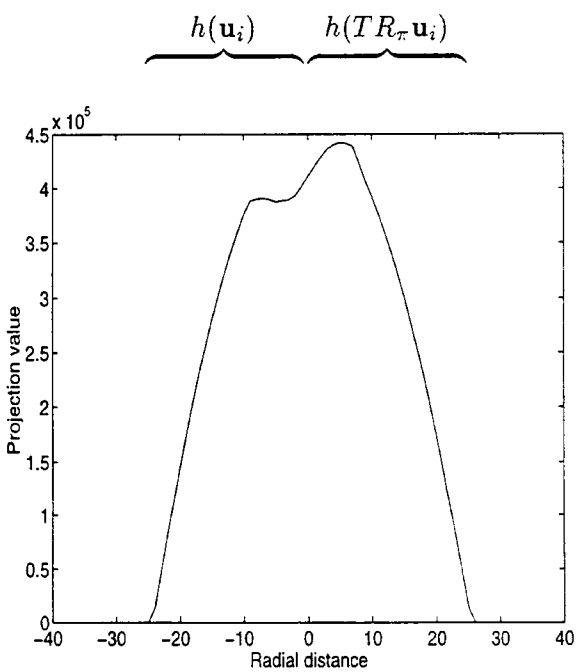


Figure 5.4: The initial estimate of the support vector is computed with a moving-average based thresholding over each projection.

Let $\mathbf{h}_0$ be the initial estimate (usually noisy and inconsistent) of the true support vector obtained by the moving average based thresholding along spherical directions $\mathbf{u}_i$. Let $\mathbf{Q} \in \mathbb{R}^{N_t \times N_s}$ be the matrix which embodies the consistency tests for the local families. Then, the closest consistent support vector $\mathbf{h}^\star$ can be found by solving the following quadratic minimization problem:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\mathbf{h}^T\mathbf{h} - \mathbf{h}^T\mathbf{h}_0 \\ &\text{subject to} \quad \mathbf{Q}\mathbf{h} \leq \mathbf{0}. \end{aligned} \tag{5.7}$$

Global consistency of a set of measurements requires local consistency of all tests for all local families from $\mathbf{S}$. For each vector $\mathbf{u}_i \in \mathbf{S}$ we must find all triplets $(\mathbf{u}_j, \mathbf{u}_k, \mathbf{u}_l)$ such that $\mathbf{u}_i$ is the unique vector in the positive cone formed by $\{\mathbf{u}_j, \mathbf{u}_k, \mathbf{u}_l\}$. This results in a $O(N_s^3)$ search for just one family and $O(N_s^4)$ for the all of them, without considering the test for uniqueness. When $N_s = 3,290$ ($N_\theta = 51$ as in chapter 4) there are approximately 35×10^9 (thirty-five billion!) possible triplets to consider for each local family. To attack this problem, we take an approach similar to the one we used for the system matrix in the previous chapter. First, we explain how symmetries can reduce the number of local families that must be found. Secondly, we distribute the resulting set of local families across a network of processors, and finally we describe some simple heuristics that substantially speed-up the overall computation time.

5.3.2 Finding local families

Assume that the local family $\mathcal{L}(\mathbf{u}_k)$ for a given direction $\mathbf{u}_k$ has been found and let $\{\mathbf{u}_k, \mathbf{u}_{k_1}, \mathbf{u}_{k_2}, \mathbf{u}_{k_3}\}$ be one 4-tuple in $\mathcal{L}(\mathbf{u}_k)$, that is $\mathbf{u}_k \in \text{cone}^+\{\mathbf{u}_{k_1}, \mathbf{u}_{k_2}, \mathbf{u}_{k_3}\}$. Then by definition,

$$\mathbf{u}_k = \sum_{i=1}^{3} \alpha_{k_i} \mathbf{u}_{k_i} \tag{5.8}$$

and $\alpha = [\alpha_{k_1}, \alpha_{k_2}, \alpha_{k_3}]^T > 0$. Applying a rotation operator $R_{\pi/2}$ to (5.8), we obtain:

$$R_{\pi/2}\mathbf{u}_k = \sum_{i=1}^{3} \alpha_{k_i} R_{\pi/2}\mathbf{u}_{k_i}. \tag{5.9}$$

Therefore, if $\mathbf{u}_k \in \text{cone}^+\{\mathbf{u}_{k_1}, \mathbf{u}_{k_2}, \mathbf{u}_{k_3}\}$, then $R_{\pi/2}\mathbf{u}_k \in \text{cone}^+\{R_{\pi/2}\mathbf{u}_{k_1}, R_{\pi/2}\mathbf{u}_{k_2}, R_{\pi/2}\mathbf{u}_{k_3}\}$. Moreover, if $\mathbf{u}_k$ is the unique vector in former cone, then $R_{\pi/2}\mathbf{u}_k$ is the unique vector in the latter cone. To prove this property, let $\mathbf{v} \neq R_{\pi/2}\mathbf{u}_k$ such that $\mathbf{v} \in \text{cone}^+\{R_{\pi/2}\mathbf{u}_{k_1}$, $R_{\pi/2}\mathbf{u}_{k_2}, R_{\pi/2}\mathbf{u}_{k_3}\}$. Then there exists a $R_{\pi/2}^{-1}\mathbf{v} \neq \mathbf{u}_k$ in $\text{cone}^+\{\mathbf{u}_{k_1}, \mathbf{u}_{k_2}, \mathbf{u}_{k_3}\}$, which is a contradiction. Similar analysis follows for other the rotation and reflection operators.

Therefore, by finding the local family $\mathcal{L}(\mathbf{u}_k)$, we can find the local families $\mathcal{L}(R_{n\pi/2}\mathbf{u}_k)$, for $n = 1, 2, 3$, and $\mathcal{L}(R_{n\pi/2}T\mathbf{u}_k)$ for $n = 0, 1, 2, 3$, by rotations and reflections of their vectors. More importantly, though, are the facts that the set of coefficient vectors α for a local family are invariant to rotation and reflection of the family and that they (augmented with a constant -1) correspond to rows of the $\mathbf{Q}$ matrix. This can be easily seen by rewriting Equation (5.8) as

$$\mathbf{u}_k^T = \alpha^T \begin{bmatrix} \mathbf{u}_{k_1}^T \\ \mathbf{u}_{k_2}^T \\ \mathbf{u}_{k_3}^T \end{bmatrix} \tag{5.10}$$

$$\alpha^T = \mathbf{u}_k^T \mathbf{F}^{-1} \tag{5.11}$$

where $\mathbf{F} = [\mathbf{u}_{k_1}\ \mathbf{u}_{k_2}\ \mathbf{u}_{k_2}]^T$.

By restricting the search for local families to the subset of vectors in the first octant of the system geometry, i.e., vectors $\mathbf{u}_k = (\theta_m, \phi_n^m)$ for which $0 \leq \theta_m \leq \pi/2$ and $0 \leq \phi_n^m < \pi/2$, the storage requirement for $\mathbf{Q}$ and the computing time for finding the local families is reduced by a factor of eight. However, the search for local families for the reduced set of directions still requires an exhaustive search for triplets in the entire set S. This is a very demanding computational problem that requires the use of several processors for

```
Find_Local_Family (S_i)
{
(a)  for u_k ∈ S_i
(b)      unique(u_k) = TRUE;
(c)      for all triplets {u_l, u_m, u_n} ∈ U, l ≠ m ≠ n ≠ k
(d)          F = [u_l  u_m  u_n]^T;
(e)          if (u_k^T F^{-1} ≥ 0)
(f)              for u_q ∈ U, q ≠ l ≠ m ≠ n ≠ k
(g)                  if (u_q^T F^{-1} >= 0)
(h)                      unique(u_k) = FALSE;
(i)                      break;
(j)              if unique(u_k)
(k)                  L_{u_k} = L_{u_k} ∪ {{u_k, u_l, u_m, u_n}};
(l)                  add row u_k^T F^{-1} to Q;
}
```

Figure 5.5: Procedure *Find_Local_Family.*

acceptable execution time.

The subset of directions is distributed across *nprocs* processors so that each processor is responsible of finding the local families for a reduced number of directions. Let $\mathcal{S}_i = \{\mathbf{u}_{i_o}, \ldots, \mathbf{u}_{i_{n-1}}\}$ be the subset of support directions from the first octant assigned to processor i. Figure 5.5 shows the procedure that individual processors execute asynchronously (without the need to communicate) to find the local families.

The positivity condition in step (e) determines whether $\mathbf{u}_k$ is in the positive cone of $\{\mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_l\}$. Step (f) begins the determination of uniqueness for $\mathbf{u}_k$. The positivity test in step (g) determines whether $\mathbf{u}_k$ is the only vector in the positive cone. If no $\mathbf{u}_q$ satisfies test (g), then $\mathbf{u}_k$ is the unique vector in the cone and $\{\mathbf{u}_k, \mathbf{u}_l, \mathbf{u}_m, \mathbf{u}_n\}$ is added to the local family of $\mathbf{u}_k$ and the corresponding row entry in $\mathbf{Q}$ is updated in steps (k) and (l),

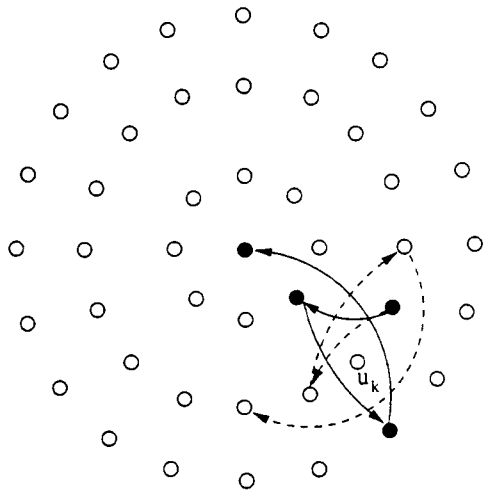


Figure 5.6: Illustration of uniqueness search for $\mathbf{u}_k$. The dashed arrows show the alternating search for vectors with same elevation angles. The solid lines show the order in which elevation angles are searched. The black dots correspond to the starting search vectors at each elevation.

respectively. Several algorithmic improvements can be applied to this procedure. For instance, since $\mathbf{F}$ is a 3×3 matrix, the entries in its inverse can be computed individually and therefore the matrix-vector multiply $\mathbf{u}_k\mathbf{F}^{-1}$ in step (e) can be done such that a new column in $\mathbf{F}^{-1}$ is computed if and only if the multiplication of $\mathbf{u}_k$ with the previous column yielded a positive value. Similar improvements are introduced when computing $\mathbf{u}_m^T\mathbf{F}^{-1}$ for the uniqueness test in (g). The search involved in the uniqueness computation can be made substantially faster by searching the spherical directions closest to $\mathbf{u}_k$ first. This is based on the heuristic that, for most cases, if $\mathbf{u}_k$ is not the unique vector in a cone, it is likely that another vector is located close to $\mathbf{u}_k$. Ideally, we would like to search S along growing concentric circular regions with center $\mathbf{u}_k$, but the overhead introduced by this method is more costly than the savings obtained. A simpler, but still powerful method, consists in testing vectors with closer elevation and polar angles to $\mathbf{u}_k$ as shown in Figure 5.6. First, vectors $\mathbf{u}_m$ at the same elevation θ_i are tested, alternating between vectors on the right and on the left of $\mathbf{u}_k$. If no vector is found to satisfy the positive cone condition at θ_i, vectors at θ_{i+1}, θ_{i-1}, θ_{i+2} and so on are tested in the same fashion until a vector is found or S is exhausted. The starting vector at any elevation angle is the one with polar

Table 5.1: Statistics for algorithm *Find_Local_family* with 24 processors using symmetries.

N_θ	No. of $\mathcal{L}$'s		No. of Cones			Wall-clock time	
	Total	Considered	Smallest	Largest	Total	Min.	Max.
21	550	75	506	903	59,774	0.05h	0.04h
31	1,210	160	1,221	2,122	304,344	1.0h	0.8h
41	2,126	277	2,164	3,885	945,292	9.2h	8.3h
51	3,290	425	3,414	6,093	2,290,094	71.3h	57.8h

direction closest to $\mathbf{u}_k$'s polar direction.

Table 5.1 shows some results concerning the finding of local families when using 24 143-MHz UltraSPARC processors. From left to right, we have: the number of elevation angles, the total number of local families (number of spherical directions), the number of local families in the first octant of the system geometry, the number of cones (tests) in the smallest and largest local families, the total number of tests to be carried out (number of rows of $\mathbf{Q}$), and the minimum and maximum wall-clock execution times for any processor to finish. The number of cones were obtained using symmetries. Notice that the number of local families that are actually searched for is not exactly the total number of local families divided by 8; the family at the north pole has no rotations, and the families at the equator have no reflections. As expected, increasing the number of spherical directions by a factor of k, increases of the execution time by a factor of k^4. Although processors do not communicate during the execution, the total time is still bounded from below by the maximum execution time among them. The difference between the minimum and maximum execution times appears large, especially for $N_\theta = 51$. This suggests an unbalanced workload distribution. However, each processor is assigned either n or $n+1$ $\mathbf{u}$'s and by simple calculations from the table, one finds that these difference between minimum and maximum execution times is on average the time needed to find the local family of just one $\mathbf{u}$.

5.3.3 Optimization

It is convenient to solve Equation (5.7) by formulating the following dual problem with a simpler constraint set [1]:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\mathbf{a}^T\mathbf{P}\mathbf{a} + \mathbf{r}^T\mathbf{a} \\ &\text{subject to} && \mathbf{a} \geq \mathbf{0} \end{aligned} \tag{5.12}$$

where $P \equiv (p_{ij}) = \mathbf{Q}\mathbf{Q}^T$, and $\mathbf{r} = -\mathbf{Q}\mathbf{h}_0$. If $\mathbf{a}^\star$ is the solution of Equation (5.12), then $h^\star = \mathbf{h}_0 - \mathbf{Q}^T\mathbf{a}^\star$ is the solution of Equation (5.7). Incorporating the additional constraint $\mathbf{h} \geq \mathbf{h}_0$ such that to guarantee the solution's convex hull fully covers the initial estimate, we obtain:

$$\begin{bmatrix} \mathbf{Q} \\ -\mathbf{I} \end{bmatrix} \mathbf{h} \leq \begin{bmatrix} \mathbf{0} \\ -\mathbf{h}_0 \end{bmatrix} \tag{5.13}$$

The new dual becomes:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\bar{\mathbf{a}}^T\bar{\mathbf{P}}\bar{\mathbf{a}} + \bar{\mathbf{r}}^T\bar{\mathbf{a}} \\ &\text{subject to} && \bar{\mathbf{a}} \geq \mathbf{0} \end{aligned} \tag{5.14}$$

where

$$\bar{\mathbf{a}} = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} \quad \bar{\mathbf{P}} = \begin{bmatrix} \mathbf{Q} \\ -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^T & -\mathbf{I} \end{bmatrix} \quad \bar{\mathbf{r}} = \begin{bmatrix} \mathbf{r} \\ \mathbf{0} \end{bmatrix} \tag{5.15}$$

and $\mathbf{a}$ is from (5.12) and $\mathbf{b} \in \mathbb{R}^{N_s}$ is a new dummy vector. If $\bar{\mathbf{a}}^\star$ solves Equation (5.14) then,

$$\mathbf{h}^\star = \mathbf{h}_0 - (\mathbf{Q}^T\mathbf{a}^\star - \mathbf{b}^\star) \tag{5.16}$$

solves Equation (5.7).

If matrix $\mathbf{Q}$ is sufficiently small to fit in just one processor's memory, the solution to Equation (5.14) can be found with a Gauss-Seidel iteration. If that is not case or if speed is a critical factor, a modified Jacobi iteration, which is more easily parallelized, can be used instead. For completeness, we will describe both algorithms. Before proceeding, however, the following remarks which are true for both algorithms are in order:

1. Because of efficiency considerations, and due to symmetries in the geometry, matrix $\mathbf{Q}$ is built to store tests for only 1/8 of the local families. Tests for other families must be obtained through rotations and reflections.

2. The solution (5.16) obtained is an ideal support vector. In real applications the vector element values must be quantized to correspond to actual distances to the centers of projection planes, that is the support vector must be made a physically realizable solution. It is the quantized support vector that is used afterwards by the reconstruction algorithm, not the continuous solution obtained from the optimization.

3. It is convenient to use a stopping criteria in terms of the support vector rather than in terms of the dual vector. In particular, the algorithms is stopped when $\|\mathbf{h}_{k+1} - \mathbf{h}_k\| \leq \epsilon$, where ϵ is a parameter defined by the user.

Gauss-Seidel method

When the jth coordinate is updated, the Gauss-Seidel iteration has the form:

$$\bar{a}_i = \bar{a}_i - \min\{\bar{a}_i, \frac{1}{\bar{p}_{ii}}(\bar{r}_i + \sum_{k=0}^{N_t-1} \bar{p}_{ik}\bar{a}_k)\} \tag{5.17}$$

$$\bar{a}_k = \bar{a}_k \quad \forall k \neq i \tag{5.18}$$

for $i = 0, \ldots, N_t - 1$. Equation (5.18) means that the update of the current variable uses the most recent values of all the other variables.

Let

$$\mathbf{y} = -\mathbf{Q}^T\mathbf{a} + \mathbf{b} \tag{5.19}$$

and let $\mathbf{q}_i^T$, and $\bar{\mathbf{p}}_i^T$ denote the ith row of matrices $\mathbf{Q}$ and $\bar{\mathbf{P}}$, respectively. Then,

$$\bar{\mathbf{P}}\bar{\mathbf{a}} = -\begin{bmatrix} \mathbf{Q}\mathbf{y} \\ -\mathbf{y} \end{bmatrix}$$

and

$$\bar{\mathbf{p}}_i^T\bar{\mathbf{a}} = \begin{cases} -\mathbf{q}_i^T\mathbf{y} & i = 0, \ldots, N_t - 1 \\ \mathbf{y} & i = N_t, \ldots, N_t + N_s - 1 \end{cases}$$

Therefore, iterations (5.17) and (5.18) can be rewritten in terms of vectors $\mathbf{a}$ and $\mathbf{b}$ as follows:

$$a_j = a_j - \min\{a_j, \frac{1}{p_{jj}}(r_j - \mathbf{q}_j^T\mathbf{y})\} \tag{5.20}$$

$$u_i = u_i \quad \forall i \neq j \tag{5.21}$$

$$\mathbf{y} = -\mathbf{Q}^T\mathbf{a} + \mathbf{b} \tag{5.22}$$

for $j = 0, \ldots, N_t - 1$, and

$$b_j = b_j - \min\{b_j, y_j\} \tag{5.23}$$

$$y_j = y_j - \min\{b_j, y_j\} \tag{5.24}$$

for $j = 0, \ldots, N_s - 1$. Step (5.20) updates a single component of vector $\mathbf{a}$ and step (5.22) makes the corresponding changes to $\mathbf{y}$ to reflect the new value of $\mathbf{a}$. These two steps are iterated simultaneously until all components in $\mathbf{a}$ have been updated. It is

possible to further simplify step (5.21) and to avoid the matrix-vector multiply every time a component is changed. Let $\mathbf{e}_j$ be the jth unit vector and $m = \min\{a_j, (r_j - \mathbf{q}_j^T\mathbf{y})/p_{jj}\}$ from (5.20). Then, after updating the jth component of $\mathbf{a}$, we have

$$\mathbf{y} = -\mathbf{Q}^T\mathbf{a} + \mathbf{b} \tag{5.25}$$

$$= -\mathbf{Q}^T(\mathbf{a} - m\mathbf{e}_j) + \mathbf{b} \tag{5.26}$$

$$= \mathbf{y} + m\mathbf{Q}^T\mathbf{e}_j \tag{5.27}$$

$$= \mathbf{y} + m\mathbf{q}_j \tag{5.28}$$

After a full sweep of vector $\mathbf{a}$, steps (5.23) and steps (5.24) execute similar operations on vectors $\mathbf{b}$ and $\mathbf{y}$. Here, we have used the fact that $\bar{r}_i = 0$ and $\bar{p}_{ii} = 1$ for $i = N_t, \ldots, N_t + N_s - 1$.

From Equations (5.19) and (5.16) the support vector that solves Equation (5.7) is $\mathbf{h}^* = \mathbf{h}_0 + \mathbf{y}$.

Jacobi method

The major disadvantage of the above Gauss-Seidel iteration is that it does not lend itself well to parallelization. The degree of parallelism of a Gauss-Seidel iteration is a function of the sparsity structure of matrix $\mathbf{P}$. When the lower triangular part of $\mathbf{P}$ has a certain structure, it is possible to partition the matrix using coloring and other numbering schemes. However, in the absence of any information about the structure of the matrix and the fact that we cannot use any other partitioning but the one defined by geometrical symmetries, we prefer to use the following linearized projected Jacobi method:

$$\bar{a}_j(t+1) = \max\left\{0, \bar{a}_j(t) - \frac{\gamma}{\bar{p}_{jj}}\left(\bar{r}_j + \sum_{k=0}^{N_t-1} \bar{p}_{jk}\bar{a}_k(t)\right)\right\}, \tag{5.29}$$

for $j = 0, \ldots, N_t - 1$. γ is a step size parameter controlling the convergence rate of the iteration and t is iteration number that changes after all variables have been updated. This Jacobi iteration scheme is more suitable for parallelization, but has the disadvantage that γ must be found experimentally to obtain reasonable convergence rates.

Following the same steps as we did for Gauss-Seidel the final iteration becomes:

$$a_j(t+1) = a_j(t) - \min\{a_j(t), \frac{\gamma}{p_{jj}}(r_j(t) - \mathbf{q}_j^T \mathbf{y}(t))\} \tag{5.30}$$

for $j = 0, \ldots, N_t - 1$, and

$$b_j(t+1) = b_j(t) - \min\{b_j(t), \gamma\, y_j(t)\} \tag{5.31}$$

$$\mathbf{y}(t+1) = -\mathbf{Q}^T \mathbf{a}(t+1) + \mathbf{b}(t+1) \tag{5.32}$$

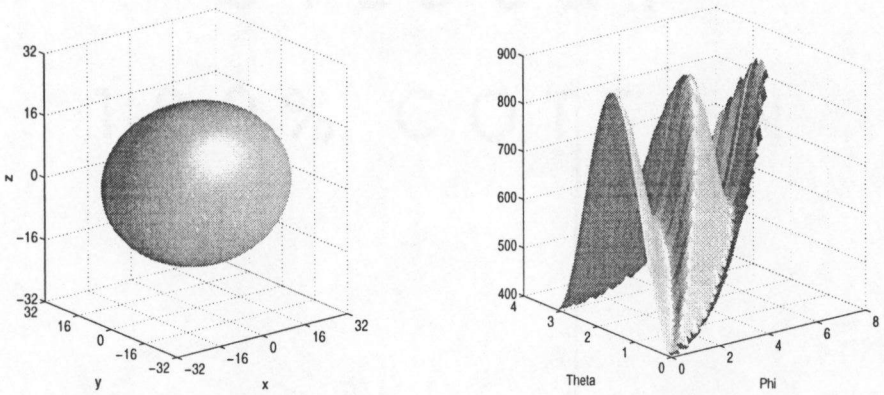
for $j = 0, \ldots, N_s - 1$. As before, the solution that solves Equation (5.7) is $\mathbf{h}^* = \mathbf{h}_0 + \mathbf{y}$.

An example

To study the performance of the Jacobi iteration, we generate support measurements for an ellipsoid with semi-axis $a = 30$, $b = 25$, $c = 20$, shown in Figure 5.7 (a). The support function of an ellipsoid can be computed analytically and is equal to:

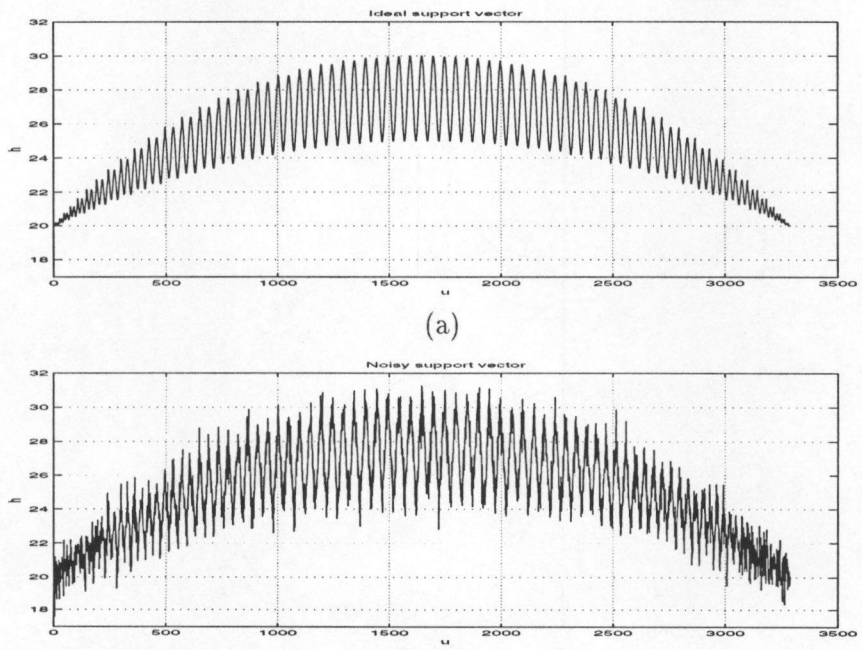
$$h(\theta, \phi) = \sqrt{a^2 \sin^2\theta \cos^2\phi + b^2 \sin^2\theta \sin^2\phi + c^2 \cos^2\theta}$$

A support vector from this support function was sampled using the spherical sampling method with $N_\theta = 51$ and a total of $N_s = 3,290$ spherical directions. The support vector and its 1D representation are plotted in Figure 5.7 (b) and Figure 5.8 (a), respectively. To create an inconsistent of support vector, we added Gaussian noise with zero mean and variance 0.0005, to the true support vector (see Figure 5.8 (b)). Finally, the linearized Jacobi method was distributed among 24 processors and executed for values of $\gamma = 0.0003$, 0.001, 0.0015, and 0.0020. The iterations were stopped when $||\mathbf{h}_{k+1} - \mathbf{h}_k|| <= 10^{-2}$.



(a) (b)

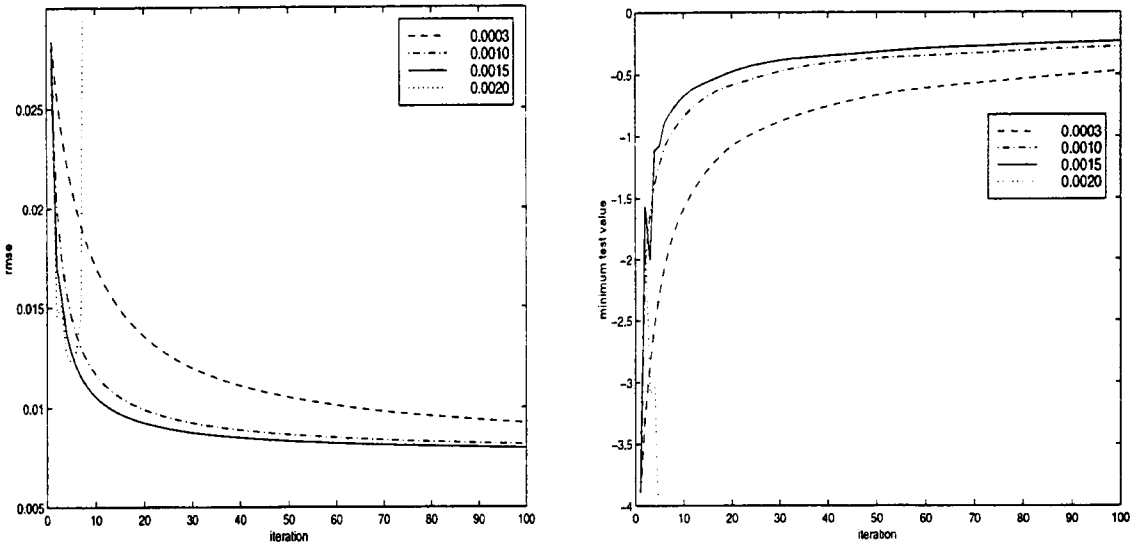
Figure 5.7: Ellipsoid and ideal support function.



(a)

(b)

Figure 5.8: Ellipsoid support vectors. (a) Consistent, (b) inconsistent due to added Gaussian noise.

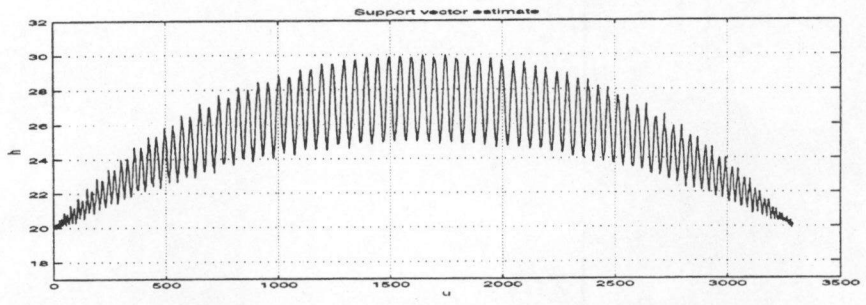


(a) (b)

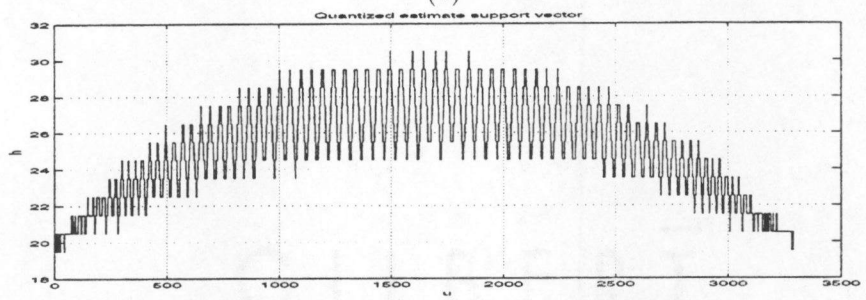
Figure 5.9: Error measures for linearized Jacobi algorithm. (a) Root mean square error between ideal and estimated support vector. (b) Minimum test value.

Two error measures are plotted in Figure 5.9 for the first 100 iterations: (a) $\|\mathbf{h}^* - \mathbf{h}^k\|/\|\mathbf{h}^k\|$, the normalized error between the ideal support vector and the current estimate, and (b) $\min \mathbf{Q}\mathbf{h}_k$, the minimum test value. The number of iterations needed for each γ to satisfy the stopping criterion are: 233, 169, and 153 for $\gamma = 0.0003$, 0.0010, and 0.0015 respectively. Experimentally, we find $\gamma = 0.0015$ to yield good convergence rates and γ values five times smaller than 0.0015 still provide reasonable speed. On the other hand, for values of γ slightly larger than 0.0015 the algorithm diverges quickly and very early in the iteration. Notice also the abrupt jump in $\min \mathbf{Q}\mathbf{h}_k$ for $\gamma = 0.0015$. In general, in this particular implementation, there is a very narrow range of values for γ that provide good convergence rates and the algorithm seems very sensitive for values close to the optimum. One way to overcome these problems is with a hybrid Gauss-Seidel and Jacobi method in which the variables are partitioned in subsets and at each iteration only variables in one subset are updated with a linearized Jacobi iteration [1].

The support vector and its quantized version after 100 iterations are shown in Figure 5.10.



(a)



(b)

Figure 5.10: Estimate support vectors after 100 iterations of linearized Jacobi with $\gamma = 0.0015$: (a) ideal estimate, (b) quantized estimate.

5.4 Computational impact

The FOA preprocessing scheme is data dependent because it is entirely based on the projection data. Therefore, to assess the computational benefits of using a FOA, we generate projection data for two objects of different sizes and shapes: a sphere of radius 25 and an ellipsoid with semi-axis $a = 25$, $b = 20$, $c = 23$. As in Chapter 4, the image is assumed to be $64 \times 64 \times 64 = 262,144$ blobs and there are $1645 \times 64 = 105,280$ projection planes. The distance between adjacent blobs and the distance between projection plane centers is 1.0. The relative sizes of the sphere and ellipsoid with respect to the entire volume are 75% and 81%, respectively. Each object was modeled with three different blob radii: 1.25, 1.5, and 2.0.

Table 5.2 summarizes the computational requirements for a generic reconstruction problem (without FOA) using 12 167-Mhz UltraSPARC processors that communicate through a 155 Mbps ATM network. The size of $\mathbf{A}$ is the total number of non-zero elements

Table 5.2: Computational requirements for a generic reconstruction problem.

Blob radius	**A** matrix (millions)	**A** init. (seconds)	No. mult. (billions)	Comm. (seconds)
1.25	191	36.9	2.78	2.3
1.50	218	39.5	3.18	2.3
2.00	272	46.1	3.97	2.3

stored using symmetries. The time required to initialize the matrix is the maximum time across all processors. The number of multiplications is the total number of multiplications involving $\mathbf{A}$ and $\mathbf{A}^T$ per iteration across all processors. Finally, The communication time is the time spent in communicating the image elements that are indicated by the **b**-mask, which for the generic case is always 64^3 blobs or 2 Mbytes, assuming double-precision representation.

5.4.1 On memory

The memory reduction obtained using FOA is the result of eliminating equations from the system described by Equation (4.15) according to the **p** and **b**-masks. An equation is discarded when the intersection of the corresponding projection plane with the **b**-mask is empty. However, when using symmetries we cannot use the **b**-mask as is to achieve full storage reduction because a projection plane that does not intersect the **b**-mask at a given view, may intersect it after having applied a rotation or mirroring operation. Therefore, we create and use an **s**-mask which is the smallest spherical hull that fully covers the **b**-mask; the radius of this hull is given by the largest quantized value in $\mathbf{h}^*$. The **s**-mask is only used to build matrix **A**.

Table 5.3 lists the memory requirements when using the FOA preprocessing scheme. In the case of the ellipsoid, 84% of the blobs and 38% of the projection planes can be eliminated, on average. This translates into memory savings of approximately 75% with respect to matrix **A**. Although the savings for the sphere are not quite as high, 68% is still a substantial reduction factor.

Table 5.3: FOA based memory requirements for two different objects.

Object	Blob radius	**A** matrix (millions)	**b**-mask (thousands)	s-mask (thousands)	**p**-mask (thousands)
Ellipsoid	1.25	46.7	37.9	59.7	65.6
	1.50	54.9	38.9	61.5	61.5
	2.00	72.8	44.4	65.7	65.6
Sphere	1.25	59.3	75.7	75.7	82.3
	1.50	69.6	77.9	77.9	82.3
	2.00	102.4	83.1	92.1	83.7

5.4.2 On computation and communication

Table 5.4 summarizes the computation and communication times when using FOA. The savings in initializing **A** are approximately 50% for both the ellipsoid and the sphere. The reason why these values do not agree with the savings in matrix size is because a larger portion of the initialization time is spent in memory reallocation than in filling the matrix entries. The major impact using FOA, however, are the significant savings in number of multiplications per iteration. For the ellipsoid, we have an 83% reduction and for the sphere a 68%. Notice that for the ellipsoid, the computational savings of 83% is larger than the memory savings of 75%. This is due to the fact that the memory savings are a function of the **s**-mask and the computation savings are a function of the **b**-mask. In the case of the sphere, the memory and computation savings are similar (68%) because the **s** and **b** masks are identical.

The time spent in communicating the image elements indicated by the **b**-mask is reduced in 55% for the ellipsoid and 47% for the sphere. Again, these reductions do not follow the reductions in size for the **b**-mask. The reasons for this discrepancy are not well understood. They might be due to overheads associated with the way the communication software (MPI) handles global reductions. Besides, these times are wall-clock times and they include overheads from all layers of the communication.

Although the values in Table 5.4 are independent of the algorithm and number of

Table 5.4: Computation and communication timings using FOA

Object	Blob radius	A init. (seconds)	No. mult. (billions)	Comm. (seconds)
Ellipsoid	1.25	15.89	0.42	0.98
	1.50	17.19	0.51	0.96
	2.00	19.79	0.72	1.16
Sphere	1.25	18.11	0.87	1.17
	1.50	19.34	1.01	1.24
	2.00	23.43	1.35	1.27

iterations used to find the solution, it is clear that FOA constitutes a very powerful pre-processing scheme for reducing the computational requirements of the 3D reconstruction algorithm at hand. On average, for a small object like the ellipsoid, the expected savings in computation and communication are 80% and 55%, respectively. For larger objects, like the sphere, the corresponding savings are 66% and 46%.

Chapter 6

Experimental Results

6.1 Method

6.1.1 Computing environment

Distributed versions of the LSQR and RL-ML algorithms with and without FOA for three dimensional image reconstruction have been implemented. The code is written in the C programming language and the inter-processor communication is handled by the MPICH implementation of the MPI standard for parallel and distributed computing. Although the implementation is independent of the hardware specifics, one must consider the minimum memory requirements shown in Table 4.1 for different problem sizes. We use a network of twelve SUN UltraSparc II workstations connected through a 155 Mbps ATM switch. Each machine is equipped with two 167 Mhz processors and 256 Mbytes of RAM. Only one processor per machine is used.

6.1.2 Data material

The experiments in this Chapter are for the geometry parameters shown in Table 6.1.

Since BNCT is still in a research stage of development, real data is difficult to obtain. There are only few places in the world that routinely perform BNCT experiments and the subset thereof that use MRI for monitoring do not sample the data as described in

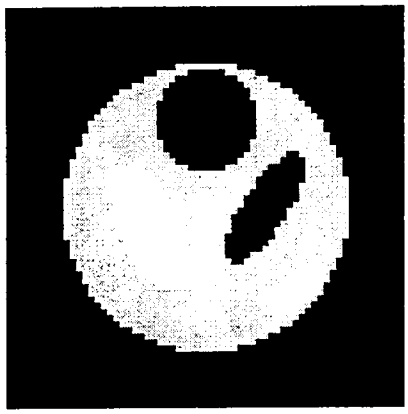
Table 6.1: Projection geometry parameters.

Symbol	Value	Description
N_θ	51	Number of elevation angles in spherical sampling of **k**-space.
N_s	3290	Number of spherical trajectories in **k**-space.
N_k	64	Number of sampled points per spherical trajectory.
$N'_\theta = (N_\theta - 1)/2 + 1$	26	Number of elevation angles in projection reconstruction
$N_v = N_s/2$	1,645	Number of view directions
$M = N_s N_k$	105,280	Number of projection planes
$N = 64 \times 64 \times 64$	262,144	Number of blobs

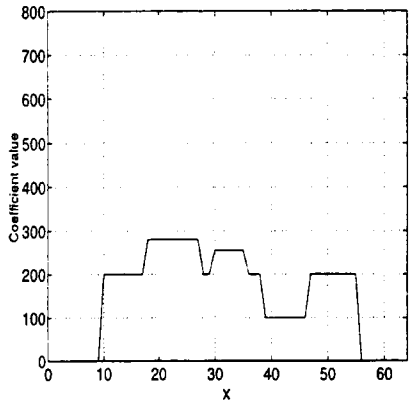
Chapter 4. The experimental study presented next, therefore, is based on a noise-free 3D simulated coefficient phantom consisting of six regions of interest (ROIs) of constant values. The central slice, i.e. $z = 31$, is shown in Figure 6.1 (a). The corresponding central profile, i.e. $y = 31$, is shown in Figure 6.1 (b). The ROI outlines, values, and number of pixels are shown in Figure 6.2.

The coefficient phantom is used to generate images according to Equation (3.1) and for the following (R, α) pairs of Kaiser-Bessel window parameters: $(1.25, 3.6)$, $(1.5, 6.4)$, and $(2.0, 10.4)$. The radius R is relative to the grid increment along the x, y, and z directions which is set to 1.0. The images can be generated to have any resolution desired, but for simplicity and throughout the experiments, we show images sampled at the grid points (blob centers). Figure 6.3 shows the resulting images for $z = 31$ and central profiles for $z = 31$, $y = 31$. Similarly, projection data is generated by forward projecting the coefficient phantom according to Equation (3.3) for the same (R, α) pairs along 1645 view directions each one consisting of 64 projection planes. The distance between the centers of adjacent projection planes is also 1.0.

Since the projection data is ideal, the support estimate from the threshold based moving average was used as the true support vector for FOA preprocessing. The **b**, **s**, and **p** masks have similar sizes as the sphere of radius 25 used in chapter 5.

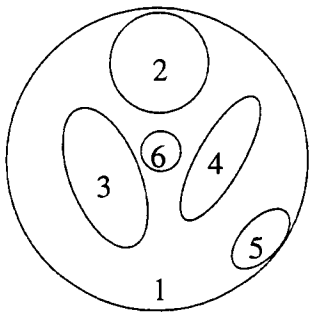


(a)



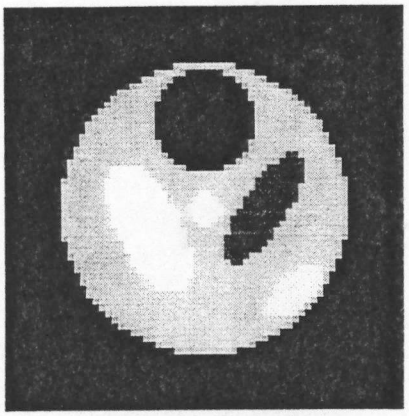
(b)

Figure 6.1: Simulated coefficient phantom used in the experiments. (a) Slice $z = 31$ and (b) corresponding central profile $z = 31$, $y = 31$.

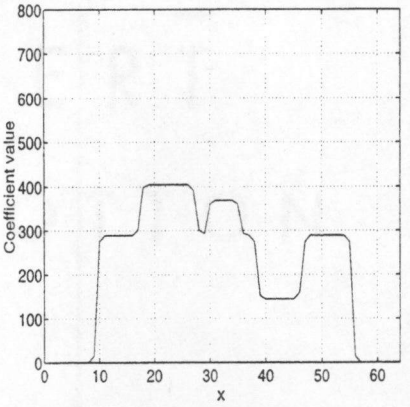


ROI	Value	No. elements
1	200	50644
2	10	2176
3	280	1360
4	100	668
5	230	240
6	255	344

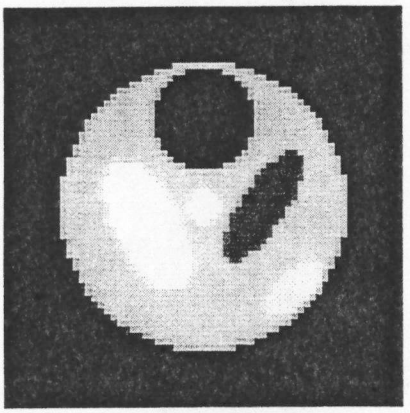
Figure 6.2: Region-of-interest (ROI) specifications.



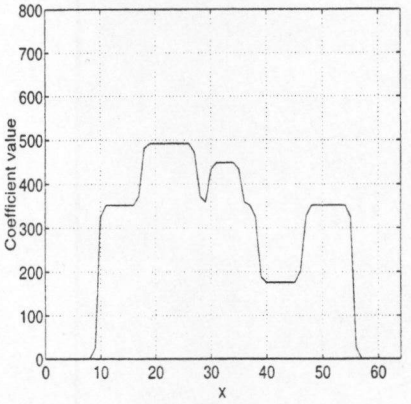
(a)



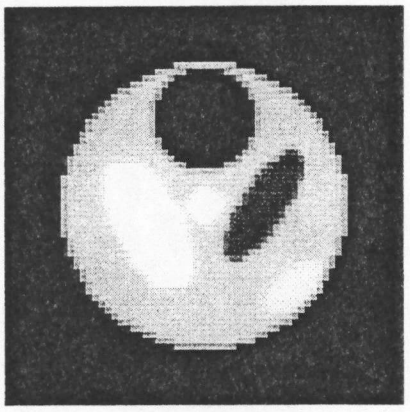
(b)



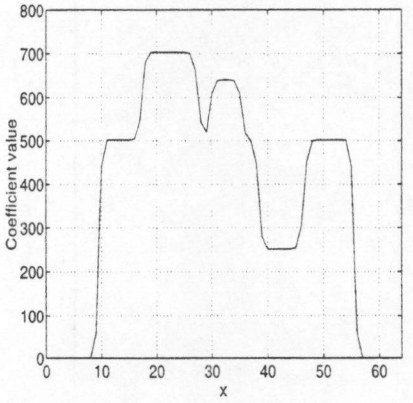
(c)



(d)



(e)



(f)

Figure 6.3: Simulated phantom images and central profiles. Images correspond to $z = 31$ and profiles correspond to $z = 31$, $y = 31$. For (a) and (b) $R = 1.25$, $\alpha = 3.6$, (c) and (d) $R = 1.5$, $\alpha = 6.4$, and (d) and (e) $R = 2.0$, $\alpha = 10.4$.

6.2 Image quality

6.2.1 Reconstructions

The reconstruction results obtained after 80 iterations of the ML-RL and LSQR algorithms with and without FOA and for the three pairs (R, α) are shown in Figures 6.4, 6.5, and 6.6. For simplicity, only images corresponding to slices at $z = 31$ are only shown. Since each pair (R, α) models a different image and different projection data according to Equations (3.1) and (3.3), it is not possible to make cross comparisons of image quality among different (R, α) pairs.

The general trend indicated by these reconstructions is that LSQR does a better job in preserving sharp transitions than RL-ML which tends to blur the image making it impossible to distinguish transitions among regions with similar values. Notice for instance, how ROI 5, the lower right ellipsoid near the phantom's border, has almost disappeared in all RL-ML reconstructions. ROI 5 can still be differentiated in LSQR images. Another interesting aspect regarding RL-ML results is the consistency in the visual appearance of images with and without FOA for all R and α. On the other hand, LSQR adds some speckling to the images, specially in those obtained with FOA and radius $R = 1.25$. The degree of speckling decreases for larger radius because of the increasing overlap among neighboring blobs. LSQR images also show a ringing phenomenon highly notorious in ROI 2 and around the inner border of ROI 1 for $R = 2.0$, $\alpha = 10.4$.

These observations are corroborated by plots of the central profiles $z = 31, y = 31$ (central row, left to right) and $z = 31, x = 31$ (central column, top to bottom) with FOA shown in Figures 6.7 and 6.8, respectively. Reconstructions for other values of R and α with and without FOA display similar trends and are not shown here. The RL-ML profiles are very smooth and do not closely follow sharp transitions nor do they reach high peaks. The LSQR profiles easily approximate high peaks, but constant regions are approximated with oscillations that are the source of the mentioned ringing the images. This is clearly seen in the deep valley in Figures 6.7 (b) and 6.8 (b).

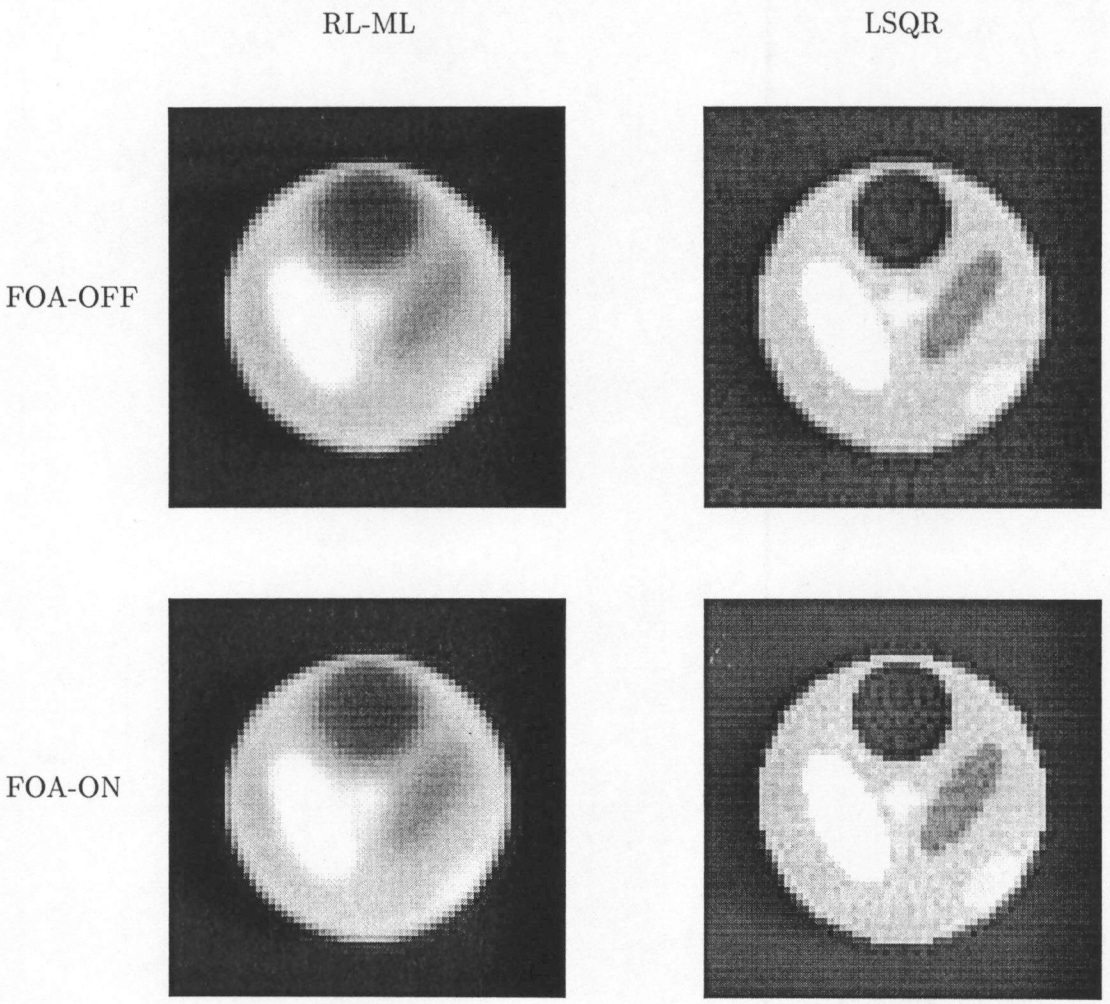


Figure 6.4: Reconstructed images obtained by the RL-ML and LSQR algorithms with and without FOA with $R = 1.25$, $\alpha = 3.6$.

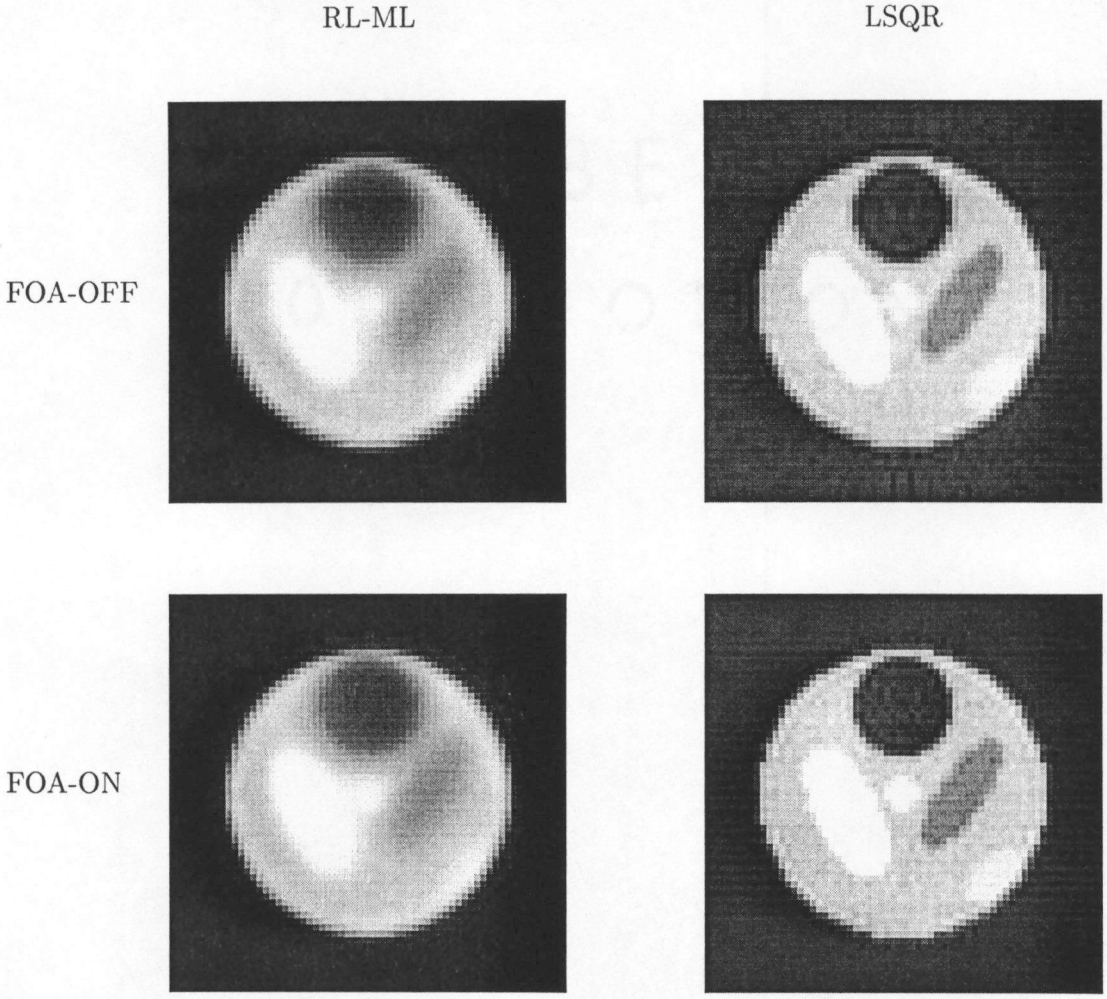


Figure 6.5: Reconstructed images obtained by the RL-ML and LSQR algorithms with and without FOA with $R = 1.5$, $\alpha = 6.4$.

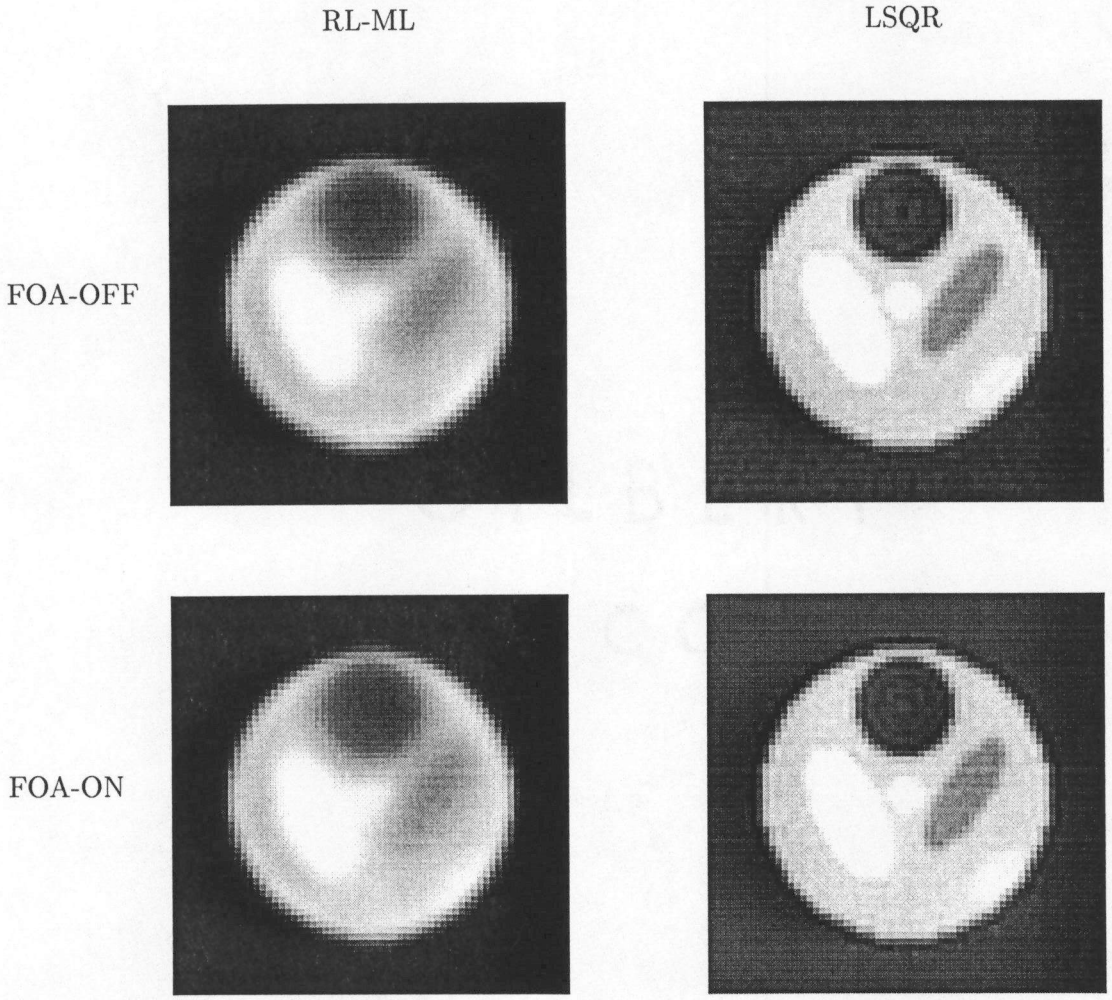
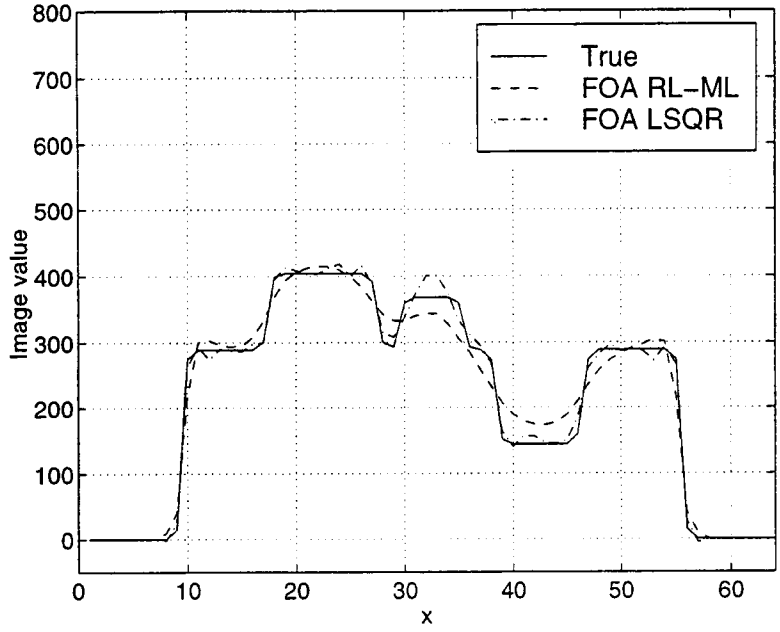
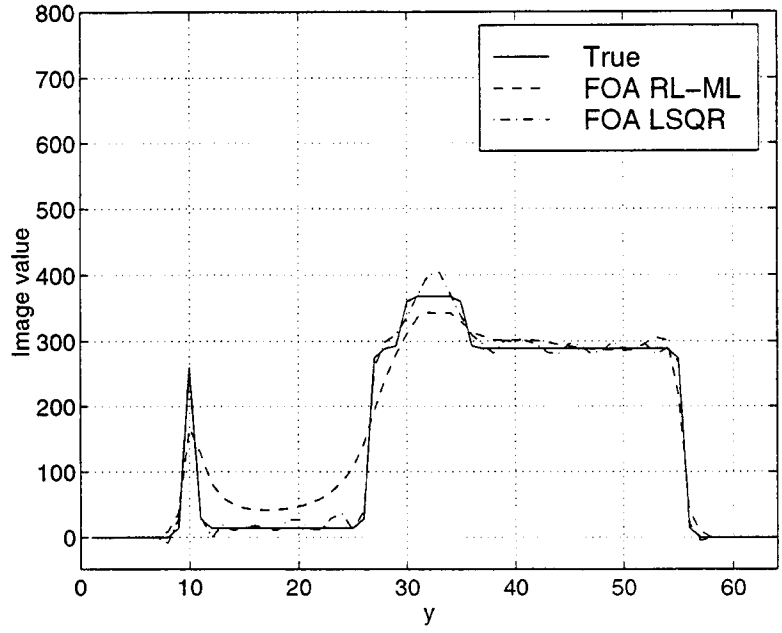


Figure 6.6: Reconstructed images obtained by the RL-ML and LSQR algorithms with and without FOA with $R = 2.0$, $\alpha = 10.4$.

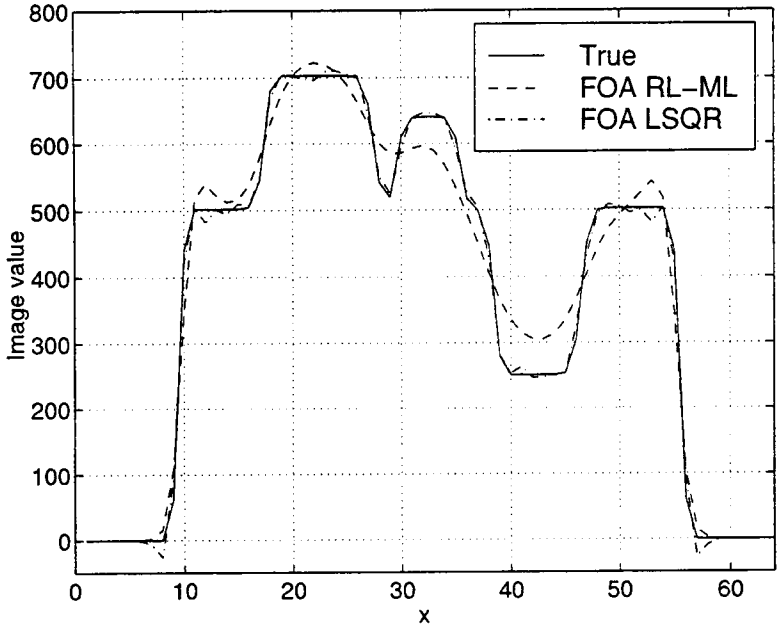


(a)

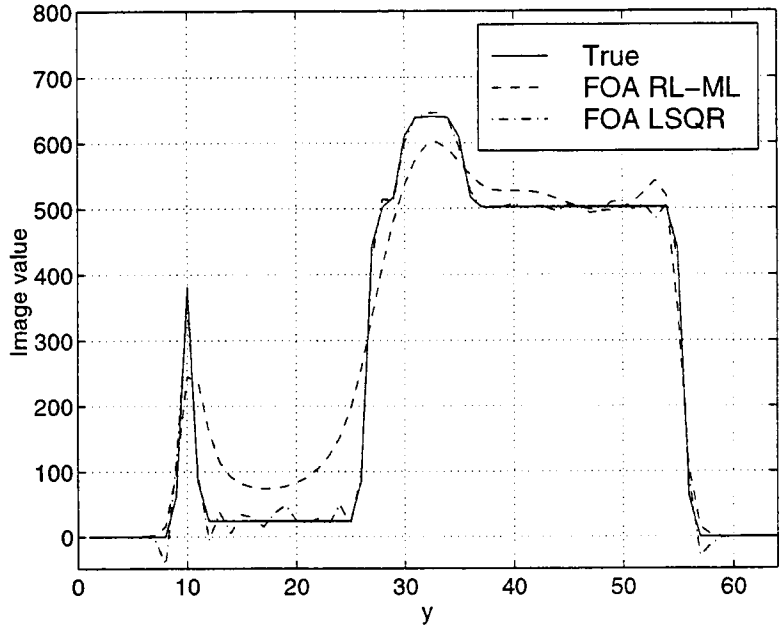


(b)

Figure 6.7: Central profiles obtained by the RL-ML and LSQR algorithms with FOA with $R = 1.25$ and $\alpha = 3.6$. For (a) $z = 31, y = 31$, and (b) $z = 31, x = 31$.



(a)



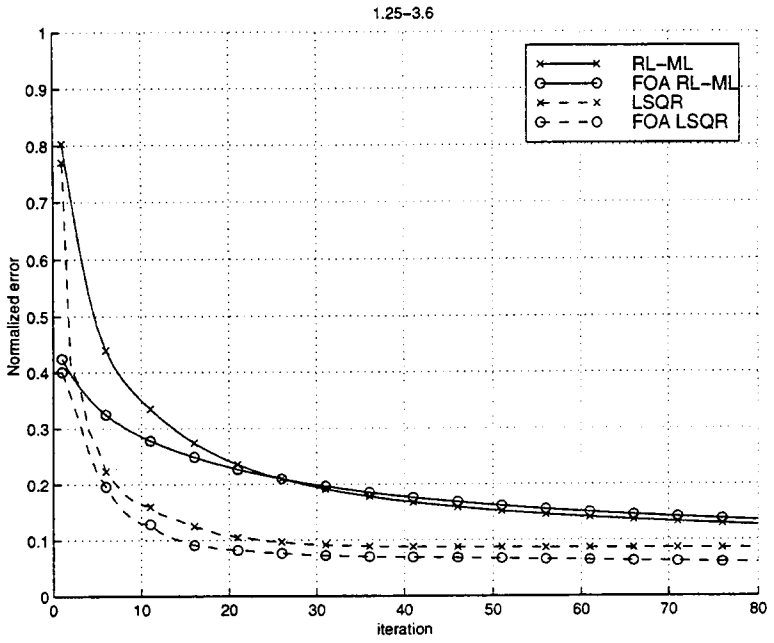
(b)

Figure 6.8: Central profiles obtained by the RL-ML and LSQR algorithms with FOA with $R = 2.0$ and $\alpha = 10.4$. For (a) $z = 31, y = 31$, and (b) $z = 31, x = 31$.

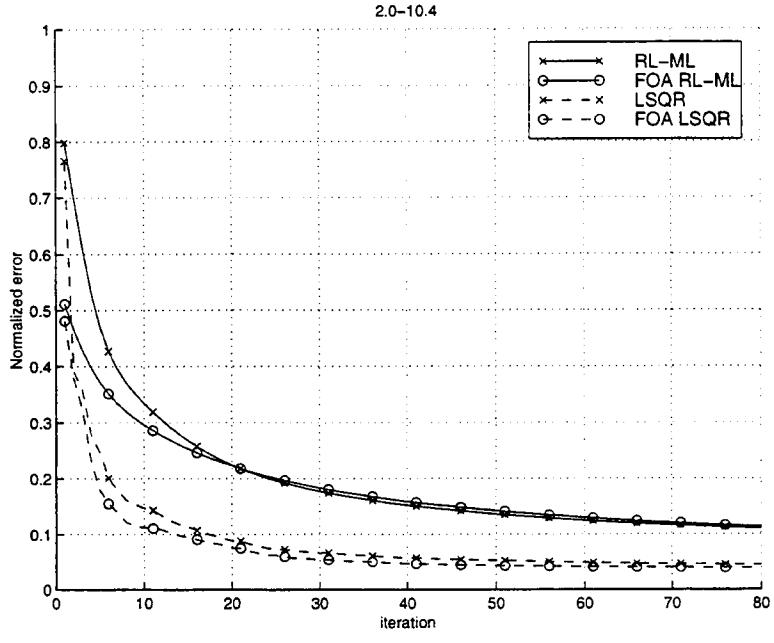
6.2.2 Error measures

Figure 6.9 shows plots of the normalized error between the true and reconstructed images. Solid lines are for RL-ML and segmented lines for LSQR. Circles and crosses indicate whether FOA is used or not, respectively. Clearly, the LSQR error is always smaller than the RL-ML error. As exemplified in Figure 6.9 (b), RL-ML requires 80 iterations to obtain the same error that LSQR achieves in 15 iterations. The error value with $R = 2.0, \alpha = 10.4$ at iteration 30 is approximately 18% for RL-ML and 7% for LSQR, which are relatively high considering that these reconstructions are from ideal data. After 40 iterations LSQR has basically converged to an error value of 5%. Also, notice that the use of FOA does not alter the error function in all cases.

A local analysis of image quality [25] can be carried out by comparing statistical quantities such as the mean region value and standard deviation for the ROIs outlined in Figure 6.2. Figures 6.10 and 6.11 plot for each ROI and for every other 11 iterations the sample mean plus-minus one standard deviation when FOA is used in RL-ML and LSQR ($R = 2.0$ and $\alpha = 10.4$). The true mean is indicated by a dotted line. Other cases show similar behavior. It is clear that LSQR convergences faster to the true mean value than RL-ML. However, they both have a hard time decreasing the standard deviation of ROIs 1 and 2 which are the two largest ROIs in the image having very different mean values. Observe how RL-ML converges easily to the true mean values of ROIs 1, 3, 5, and 6, but it does not do so for ROIs 2 and 4. It appears then that differences in the error functions between RL-ML and LSQR come mainly from approximating ROIs 2 and 4, which are regions with low values surrounded by regions with high values. Since RL-ML coefficients are always positive, it is more difficult to compensate for large changes between regions. A region with high value will be approximated with a large coefficient and the resulting blobs will contribute to adjacent regions always with a positive value. This problem is accentuated by larger blob radii. LSQR has the possibility of using negative coefficients to overcome this problem. But this is also the source of undesirable negative image values. Notice also that both algorithms produce similar statistics whether FOA is used or not.



(a)



(b)

Figure 6.9: Plots of the normalized error between the true and reconstructed images by the RL-ML and LSQR. For (a) $R = 1.25$, $\alpha = 3.6$, (b) $R = 2.0$, $\alpha = 10.4$.

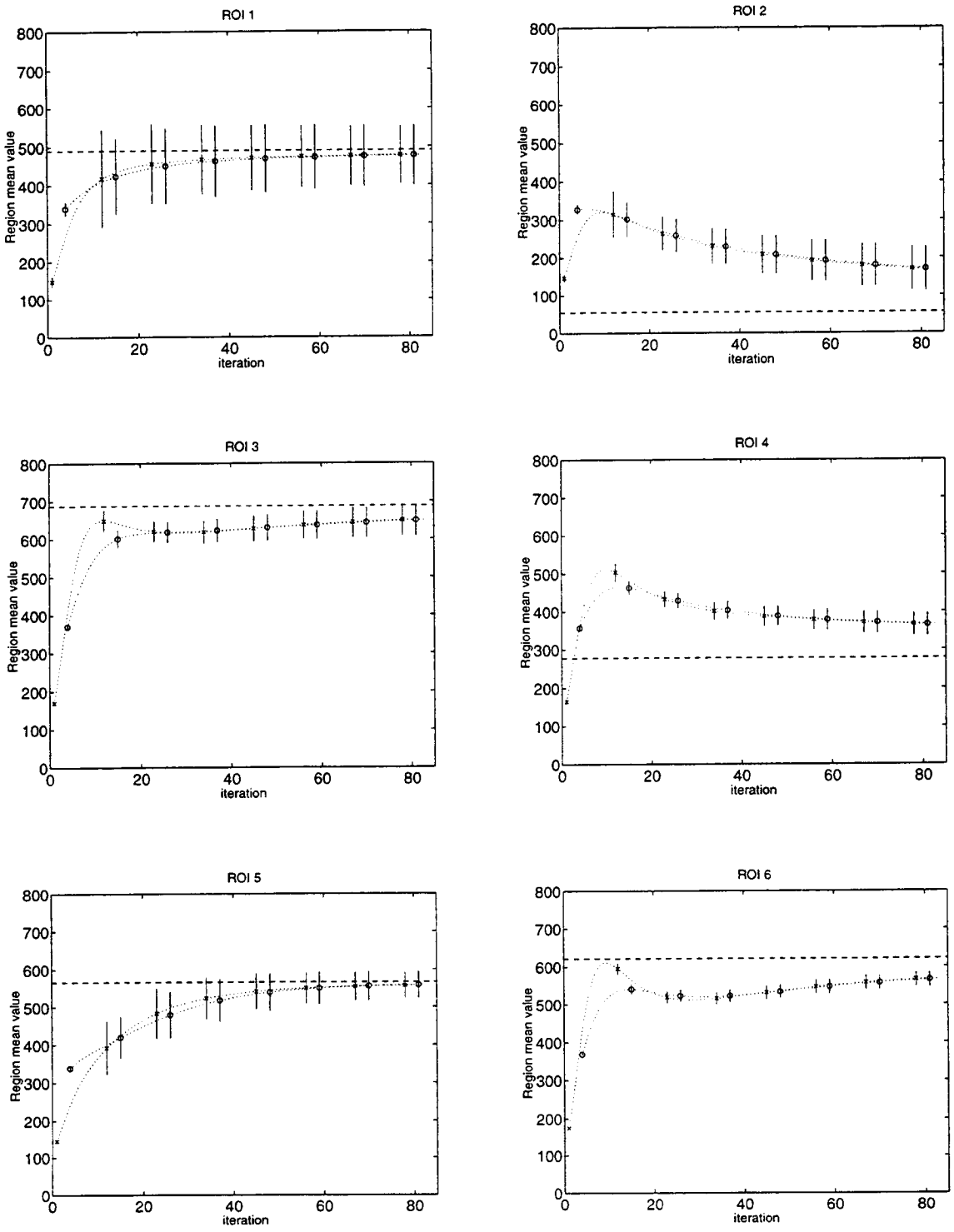


Figure 6.10: Mean plus-minus one standard deviation results obtained with the RL-ML algorithm for each of the six ROI's. Markers ◯ and × indicates results with and without FOA, respectively.

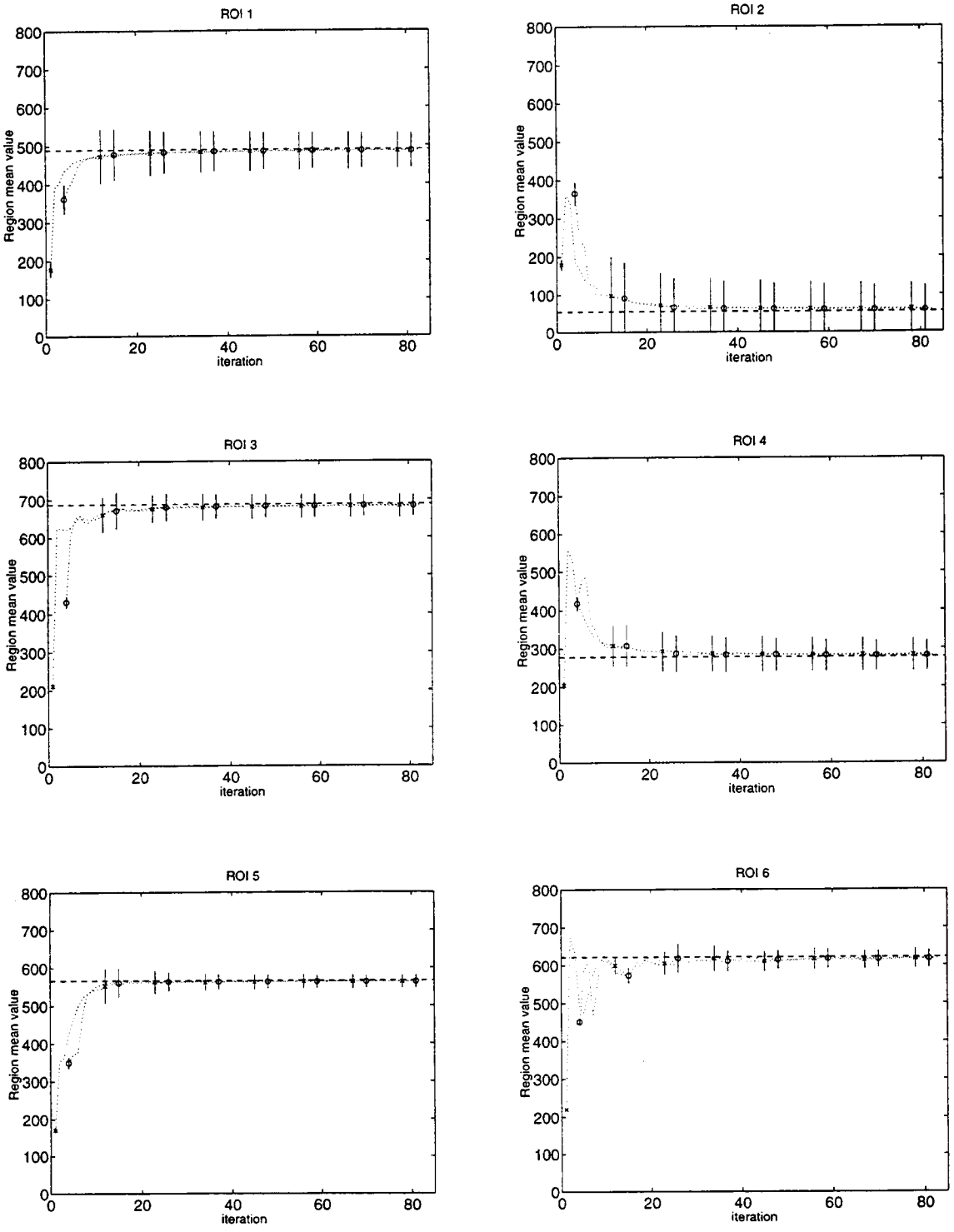


Figure 6.11: Mean plus-minus one standard deviation results obtained with the LSQR algorithm for each of the six ROI's. Markers ◯ and × indicates results with and without FOA, respectively.

6.2.3 Computational performance

An algorithm-independent computational evaluation has been provided in Chapter 5. Here, the overall performance of the RL-ML and LSQR algorithms are compared.

Figure 6.12 shows the mean wall-clock time in seconds per iteration for each of the 12 processors used. The time has two components: (1) computing time, that includes the time of forward and backward operations, matrix-vector and other floating-point operations, and (2) overall communication time required to carry out the global sum operation.

In general, the algorithms have the same overall performance and the workload is well balanced. RL-ML is slightly faster than LSQR because it requires fewer operations per iteration and only one global sum reduction. Besides the reduction needed by the backward operation in both algorithms, LSQR needs to communicate a partial inner product across the nodes. But overall both algorithms execute 40 iterations in 3 hours using 12 processors.

Table 6.2 summarizes the overall performance for the RL-ML algorithm. The wall-clock times are the mean values per iteration in minutes and the speed-up is computed as the ratio of the wall-clock time without FOA over the wall-clock time with FOA. The results for LSQR are essentially the same. On average, FOA speeds up the reconstruction problem by a factor of 2 or, alternatively, achieves a 50% time reduction. This percentage is smaller than the one computed in Chapter 5 because of the additional computing and communication times being measured now. The combined use of FOA and geometrically induced symmetries has also substantially reduced the memory requirement to the point where the problem can now be run in a small network of processors. Figure 6.13 shows the overall timings of ML-RL with FOA when using 4, 8, and 12 processors. Clearly, the reconstructions are compute intensive and communication accounts for less than 10% of the total time. RL-ML and LSQR take 3 hours to execute 40 iterations with only 4 processors. The times needed to initialize the $\mathbf{A}$ matrix are 23.4, 36.3, and 162.8 seconds, for 4, 8, and 12 processors, respectively.

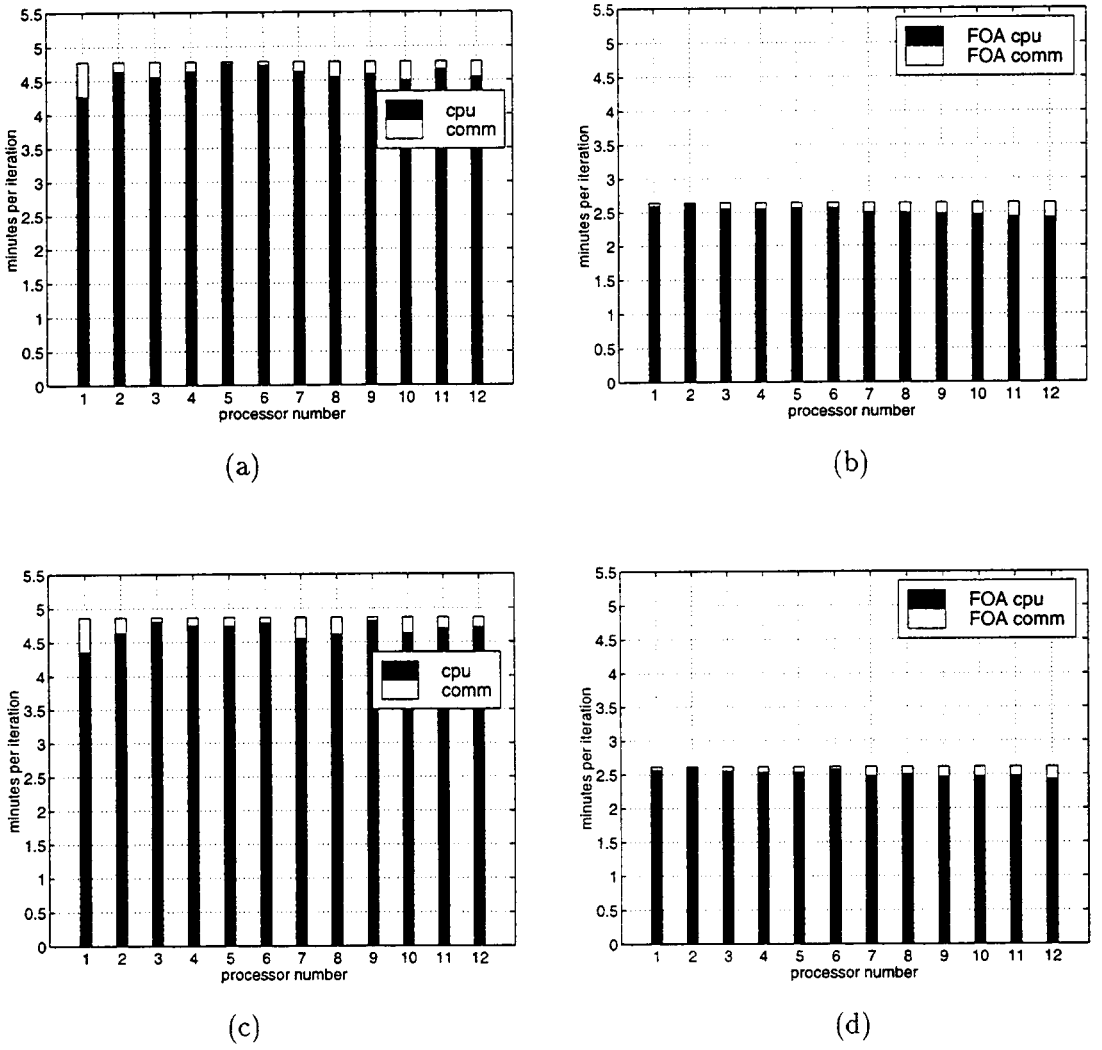


Figure 6.12: Mean wall-clock time per iteration for RL-ML and LSQR with and without FOA with $R = 2.0$, $\alpha = 10.4$. (a) and (b) RL-ML timings, (c) and (d) LSQR timings.

Table 6.2: Overall mean wall-clock time per iteration for RL-ML.

R	Mean wall-clock time		FOA
	FOA OFF	FOA ON	Speed-up
1.25	3.0 min	1.4 min	2.15
1.50	3.7 min	1.7 min	2.17
2.00	4.8 min	2.6 min	1.80

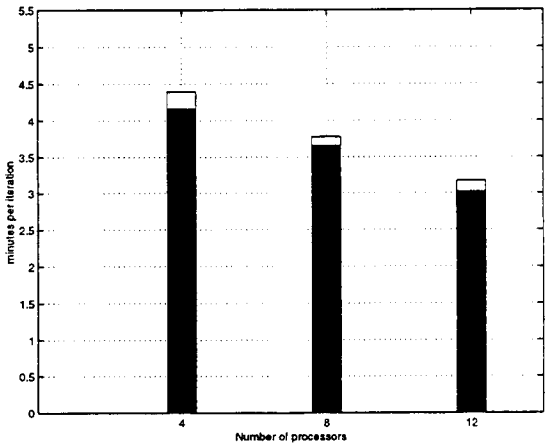


Figure 6.13: Mean wall-clock time per iteration for RL-ML with FOA with $R = 2.0$, $\alpha = 10.4$, for 4, 8, and 12 processors.

6.3 Summary

The experiments have shown that LSQR converges faster to the solution than RL-ML. Since RL-ML solutions are always positive and blobs overlap, it is difficult for RL-ML to approximate sharp changes across smooth regions. On the other hand, LSQR coefficients can be negative and therefore the contributions of overlapping blobs can be suppressed. However, LSQR tends to produce negative image values and an undesirable ringing effect. The local analysis has also revealed that, although LSQR converges faster than RL-ML to the true ROIs' mean value, they both have a hard time decreasing the variances of the same regions in the image.

The image quality results reported here must be viewed in the following context: (1) the reconstruction problem is highly undersampled, ill-conditioned and does not have a unique solution; (2) the RL-ML and LSQR are designed to minimize different objective functions and they do not necessarily converge to the same solution; and (3) the image, side-effects and artifacts are also dependent on our particular choice of basis functions.

The experiments have shown that the computational performance of RL-ML and LSQR are similar. LSQR requires an additional global reduction after a forward projection, but since it is only a single number that is transmitted, the overall performance of the algorithm is not affected. FOA preprocessing helps to speed-up the algorithms at least by a factor of two and the image quality is not affected. When applying FOA and symmetries, the largest problem can be run on as few as four workstations with an expected average wall-clock time of 4.5 minutes per iteration. The 3D reconstruction problem has been identified as compute intensive where communication accounts for only 10% of the overall execution time.

Chapter 7

Conclusion

7.1 System description

This dissertation has developed a computational framework for 3D iterative image reconstruction from boron data as an alternative to the traditional method of Fourier reconstruction [9]. As shown in Figure 7.1, it consists of three major subsystems: data collection, Focus-Of-Attention (FOA) preprocessing, and iterative reconstruction.

Data collection: This subsystem is responsible for acquiring the MRI boron data in accord with the properties of the boron signal. Due to short relaxation times of the boron signal, the object must be sampled along spherical trajectories of the k-space. Sampling always begins at the origin and extends to high frequencies along straight lines. The spherical trajectory has been modified to (1) guarantee that the k-space is sampled along straight lines passing through the origin and (2) induce geometrical symmetries. It is shown that these two sampling constraints can be satisfied by having an odd number of elevation angles, each one with a power of 4 number of polar angles. Using the projection theorem, planar projection data is obtained by 1D Fourier inversion of each sampled line in k-space.

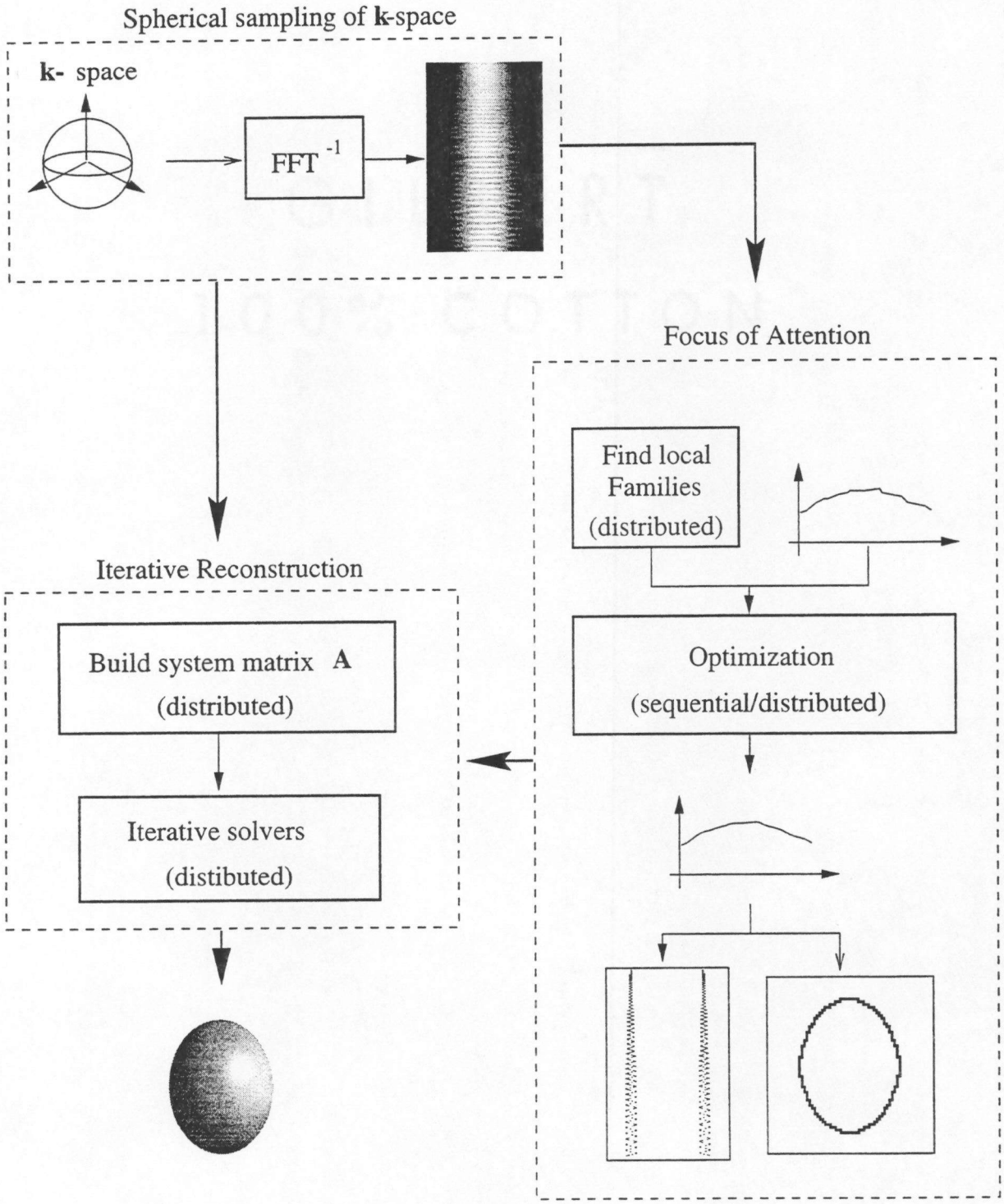


Figure 7.1: Schematic system description for iterative reconstruction of 3D images from boron data.

FOA preprocessing: The FOA subsystem is responsible for: (1) finding a consistent support vector of the object to be reconstructed and (2) defining three indicator masks that are used to reduce the size of the linear system of equations. The FOA subsystem receives from the data collection subsystem the spherical sampling geometry and the Radon projection data. The first step in FOA is to compute an estimate of the support vector, $\mathbf{h}_0$, from the projection data by means of a threshold-based moving average over each projection. Also, the local family $\mathcal{L}(\mathbf{u})$ and local consistency tests for all spherical directions $\mathbf{u}$ are computed. Local consistency tests are stored in matrix $\mathbf{Q}$. However, due to rotation and reflection symmetries induced in the sampling geometry, only a reduced number of local families and tests must be found. It is shown that the entries for other families and tests are implicitly represented by $\mathbf{Q}$. The finding of local families is a data-independent process and is carried out only once for a given sampling geometry. With $\mathbf{h}_0$ and $\mathbf{Q}$ in place, an optimization phase begins which finds the consistent support vector $\mathbf{h}^\star$ closest to $\mathbf{h}_0$ in a least-squares sense. An additional constraint has been introduced to the optimization problem in order to guarantee that the solution's convex hull fully covers the initial estimate. The resulting equations for a sequential and a parallel implementation of the optimization problem have been derived. The solution $\mathbf{h}^\star$ is quantized to correspond to physically realizable projection planes and then used to compute a **p**-mask and a **b**-mask that indicate which projection planes and blobs to subsequently focus on, respectively. The **b**-mask indicates the object's convex hull and can be computed by intersecting all regions of the image that lie in the negative side of all "boundary" planes from the **p**-mask. Another mask computed during this phase is the **s**-mask which corresponds to the smallest spherical hull that fully contains the **b**-mask.

Iterative reconstruction: This subsystem solves the support constrained reconstruction problem with distributed, iterative algorithms based on symmetries. The inputs to this subsystem are: the spherical sampling geometry and the projection data from data collection, and $\mathbf{h}^\star$, **b**, **p**, and **s**-masks from the FOA subsystem. The image is modeled as

a linear combination of spherically symmetric basis functions whose 3D Radon transform is derived. These basis functions are specially attractive for 3D reconstruction because they facilitate the handling of symmetries. The reconstruction problem is now one of solving a large and sparse linear system of equations. The system matrix **A** is built using the rotation and mirroring symmetries of the projection geometry, as well as the **p** and **s**-masks. When FOA is not applied, the **p**-mask corresponds to the entire set of projection planes and the **b**-mask and the **s**-mask to the entire image. If FOA is applied, equations and unknowns are eliminated according to the **p** and **s**-masks. An equation is eliminated when the intersection of the corresponding projection plane with the **s**-mask is empty. Two iterative solvers are proposed for solving the system of equations and their distributed implementations are provided. They are also based on symmetries, the **p**-mask, the **b**-mask. The **s**-mask is not needed during iterations. Full FOA-based computational savings are achieved.

7.2 Contributions and discussion

The major contributions of this dissertation are:

- A new signal sampling strategy that allows simultaneously the acquisition of Radon projection data and the introduction of geometrical symmetries.
- Derivation of the 3D Radon transform of the Kaiser-Bessel window function.
- The use of symmetries in a distributed environment to reduce the memory requirement of the system.
- A 3D FOA preprocessing scheme that introduces a basic form of prior knowledge into the reconstruction and reduces the computational requirements.
- Experimental results that study the merit of using Kaiser-Bessel functions and FOA in 3D Radon based projection reconstruction and the performance of two iterative algorithms.

The sampling algorithm is a modification to an already known sampling strategy. However, the modifications are significant in the sense that they greatly simplify the solution of other aspects of the system. For instance, without the symmetries the system requires around 10 gigabytes of memory for a modest reconstruction problem, an amount currently not available in small networks of workstations. When this much memory becomes available, it is likely that we will want to increase the image resolution or sampling density to obtain better quality images. We believe, therefore, that the experience gained in developing and exploiting symmetries will be valuable not only in MRI but also in other areas of 3D image reconstruction.

In addition, symmetries are useful during the FOA preprocessing phase. The time needed to find the local families is a major bottleneck in FOA. It takes two to three days to find the local families of only 1/8 of the total number of direction vectors for $N_\theta = 51$. In a geometry lacking any kind of symmetries, this time would increase to magnitudes that would make the application of the theory of support vectors impractical. This dissertation seems to be the first in applying the theory of support functions to a 3D problem. The solutions proposed for the various computational problems encountered may serve as guidelines for other applications of the theory to three and more dimensions.

Kaiser-Bessel functions have been studied before for use in 3D X-ray projection reconstruction problems like PET/SPECT. We have extended this study to 3D Radon projection reconstruction in the context of MRI. The experiments show that blobs seem to be good candidates when reconstructing smooth regions, but they are probably not the best choice for representing images with many sharp transitions, especially when reconstructing with the RL-ML or other positivity constrained algorithms. In LSQR, the combination of negative and positive coefficients permits a more accurate representation of sharp transitions.

The experiments also show that the use of FOA in 3D reconstruction is beneficial. FOA produces images with similar qualities as for generic reconstruction and it reduces substantially the computational requirements. On average, and for medium size objects,

FOA reduces the memory requirements and the workload in 75% and communication in 50%. An overall reduction of 50% can be expected when considering the iterations as a whole process. The combined effect of symmetries, FOA and distributed computation allows the problem to be solved in only four small workstations.

7.3 Future work

One aspect of paramount importance is the comprehensive study of the properties of the boron signal. Ultimately, this may lead to choosing different basis functions, sampling algorithms, or reconstruction algorithms. Although there are several studies in the literature evaluating the performance of blobs in image reconstruction, there still exists the need of fine tuning the blob's parameter selection to the boron application. This will be possible only when real data becomes available. Real data will also help in defining more accurate and complex priors for constrained reconstruction.

Important aspects for future research related to iterative algorithms are the introduction of a positivity constraint in LSQR [21] and techniques to improve the convergence rates of it and RL-ML [8]. Clearly, it is impossible to have negative concentrations of boron in the imaged region. It would be valuable to have a basis function together with a positive constraint so that the final image values actually correspond to absolute values of boron concentrations [17]. A positive solution is probably the first step toward quantitation in iterative reconstruction.

LSQR's convergence may be improved with preconditioning techniques [36]. Preconditioners based on matrix factorization do not appear applicable when the matrix is only partially stored and symmetries must be kept. Even if the preconditioner has the same sparsity degree as the original matrix, the memory requirement would at least double which would exceed the memory capacity available. Alternatively, Fourier based preconditioners [4], whose action can be algorithmically implemented, seem to be good candidate for further study. With respect to RL-ML's convergence, ordered subsets [15] has proven

to be an effective mechanism to speed-up the algorithm in SPECT/PET computed imaging. Although the technique is easily applicable, one must take into consideration the restrictions imposed by the symmetries in assigning the subsets to different processors.

Three-dimensional image reconstruction is a compute intensive task. From a computer science perspective there are plenty of opportunities for the development of reconstruction algorithms that make efficient use of emerging computer technologies.

Bibliography

Bibliography

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[2] M. A. Brown and R. C. Semelka. *MRI: Basic Principles and Applications.* Wiley-Liss, 1995.

[3] Y. Censor. Finite series-expansion reconstruction methods. *Proc. of the IEEE*, 71(3):409–419, 1983.

[4] R. H. Chan, J. G. Nagy, and R. J. Plemmons. FFT-based preconditioners for Toeplitz-block least squares problems. *SIAM J. Numer. Anal.*, 30:1740–1768, 1993.

[5] C. M. Chen, S.-Y. Lee, and Z. H. Cho. Parallelization of the EM algorithm for 3D PET image reconstruction. *IEEE Trans. Medical Imaging.*, 10:513–522, 1991.

[6] Z.-H. Cho, J. P. Jones, and M. Singh. *Foundations of Medical Imaging.* Wiley, 1993.

[7] Z.-H. Cho, H. S. Kim, H. B. Song, and J. Cumming. Fourier transform nuclear magnetic resonance tomographic imaging. *Proc. of the IEEE*, 70(10):1152–1173, 1982.

[8] N. H. Clinthorne, Tin-Su Pan, Ping-Chun Chiao, W. L. Rogers, and J. A. Stamos. Preconditioning methods for improved convergence rates in iterative reconstructions. *IEEE Trans. Medical Imaging.*, 12:78–83, 1993.

[9] G. H. Glover, J. M. Pauly, and K. Bradshaw. Boron-11 imaging with a 3D reconstruction method. *J. Magn. Reson. Imag.*, 2:47–52, 1992.

[10] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal. Series B*, 2:205–224, 1965.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations.* John Hopkins, second edition, 1989.

[12] P. Grangeat and J. L. Amans, editors. *Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine.* Kluwer Academic Publishers, 1996.

[13] J. Gregor and D. A. Huff. A computational study of the focus-of-attention EM-ML algorithm for PET reconstruction. To appear in a special issue of *Parallel Computing* on biomedical applications.

[14] J. Gregor and D. A. Huff. A focus-of-attention preprocessing scheme for EM-ML PET reconstruction. *IEEE Trans. Medical Imaging.*, 16:218–223, 1997.

[15] H. M. Hudson and R. S. Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE Trans. Medical Imaging.*, 13:601–609, 1994.

[16] G. W. Kabalka, M. Davis, and P. Bendel. Boron-11 MRI and MRS of intact animals infused with a boron-neutron-capture agent. *J. Magn. Reson. Med.*, 8:231–237, 1988.

[17] G. W. Kabalka, C. Tang, and P. Bendel. The role of boron MRI in boron neutron capture therapy. *J. Neuro-Oncology*, 33:153–161, 1997.

[18] W. C. Karl. Personal communication.

[19] W. C. Karl, S. R. Kulkarni, G. C. Verghese, and A. S. Willsky. Local tests for consistency of support hyperplane data. *J. Math. Imag. and Vision*, 6:249–267, 1995.

[20] L. Kaufman. Solving emission tomography problems on vector machines. *Annals of Operations Research*, 22:325–353, 1990.

[21] L. Kaufman. Maximum likelihood, least squares, and penalized least squares for PET. *IEEE Trans. Medical Imaging.*, 2:200–214, 1993.

[22] K. Lange and R. Carson. EM reconstruction algorithms for emission and transmission tomography. *J. Computer Assisted Tomography*, 8(2):306–316, 1984.

[23] R. M. Lewitt. Multidimensional digital image representations using generalized Kaiser-Bessel window functions. *J. Opt. Soc. of Amer. A*, 7:1834–1846, 1990.

[24] R. M. Lewitt. Alternatives to voxels for image representation in iterative reconstruction algorithms. *Phys., Med. and Bio.*, 37:705–716, 1992.

[25] J. Llacer, E. Veklerov, K. J. Coakley, E. J. Hoffman, and J. Nunez. Statistical analysis of maximum likelihood estimator images of human brain FDG PET studies. *IEEE Trans. Medical Imaging.*, 12:215–231, 1993.

[26] G. L. Locher. Biological effects and therapeutic possibilities of neutrons. *Amer. J. Roentgenol Radium Ther.*, 36:1–13, 1936.

[27] L. B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79:745–765, 1974.

[28] R. B. Lufkin. *The MRI Manual.* Mosby Year Book, 1990.

[29] S. Matej and R. M. Lewitt. Practical considerations for 3D image reconstruction using spherically symmetric volume elements. *IEEE Trans. Medical Imaging.*, 15:68–78, 1996.

[30] Y. Nakagawa and H. Hatanaka. Boron neutron capture theraphy: clinical brain tumor studies. *J. Neuro-Oncology*, 33:105–115, 1997.

[31] F. Natterer. *The Mathematics of Computarized Tomography.* Wiley & Sons, 1986.

[32] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8:43–71, 1982.

[33] J. L. Prince and A. S. Willsky. Reconstructing convex sets from support line measurements. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 12(4):377–389, 1990.

[34] J. Radon. Uber die bestimmung von funktionen durch ire integralwerte tangs gewisser mannigfaltigkeiten. *Math. Phys. Klass.*, 69:262–277, 1917. Bertichte Saechsiche Akad.

[35] W. H. Richardson. Bayesian-based iterative method of image restoration. *J. Opt. Soc. of Amer.*, 62:55–59, 1972.

[36] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.

[37] L. A. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Medical Imaging.*, 1:113–122, 1982.

[38] D. D. Stark and W. G. Bradley Jr, editors. *Magnetic Resonance Imaging*, volume one. Mosby Year Book, second edition, 1992.

[39] J. R. Venhuizen. BNCT briefing book. http://id.inel.gov/cart/briefbook/, 1997. Idaho National Engineering and Enviromental Laboratory.

[40] G. N. Watson. *A treatise on the theory of Bessel functions.* Cambridge University Press, 2nd edition, 1945.

[41] P. Woodward and R. D. Freimarck, editors. *MRI for technologists.* McGraw-Hill, Inc, 1995.

Appendix

Appendix

The 3D Radon transform of blobs

In this appendix we derive the three-dimensional Radon transform of the Kaiser-Bessel window function for a plane given by orthogonal unit vector $\mathbf{r}$ and distance τ from the origin, that is

$$\Psi(\mathbf{r}, \tau) = [\mathcal{R}_3 \psi](\mathbf{r}, \tau)$$

where

$$\psi(s) = \begin{cases} \frac{1}{I_K(\alpha)} \left[\sqrt{1-(s/R)^2}\right]^K I_K\left(\alpha\sqrt{1-(s/R)^2}\right) & 0 \le s \le R \\ 0 & \text{otherwise} \end{cases}$$

Using the shifting property of the Radon transform and the fact that ψ is spherically symmetric, it suffices to compute the Radon transform of a blob centered at the origin along a plane parallel to the xy plane as shown Figure A.1. Since the function is zero outside the sphere of radius R, we only have to integrate the shaded circle of radius $l = \sqrt{R^2 - \tau^2}$. By symmetry of ψ, we integrate only one quadrant of the circle and multiply the result by four. The Radon transform can thus be written as:

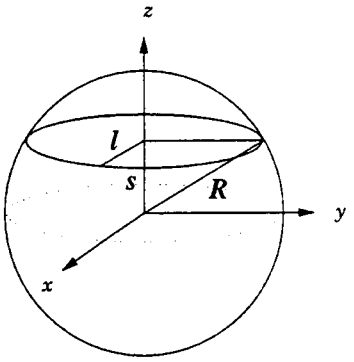


Figure A.1: Computing the three dimensional Radon transform. The filled area is the plane of integration; s is the distance from it to the origin and R is the blob radius.

$$\Psi(s) = 4 \int_0^l \int_0^{\sqrt{l^2-x^2}} \psi(\sqrt{s^2+x^2+y^2})\, dydx \tag{A.1}$$

$$= \frac{4}{I_K(\alpha)} \int_0^l q(x)\, dx \tag{A.2}$$

where

$$q(x) = \int_0^{\sqrt{R^2-(s^2+x^2)}} \left[\sqrt{1-\frac{s^2+x^2+y^2}{R^2}}\right]^K I_K\left[\alpha\sqrt{1-\frac{s^2+x^2+y^2}{R^2}}\right] dy \tag{A.3}$$

First we solve $q(x)$. Let $y = \sqrt{R^2-s^2-x^2}\cos\phi$ and $dy = -(R^2-s^2-x^2)\sin\phi\, d\phi$. The square root term simplifies to

$$\begin{aligned}
\sqrt{1-\frac{s^2+x^2+y^2}{R^2}} &= \sqrt{\frac{R^2-s^2-x^2-(R^2-s^2-x^2)\cos^2\phi}{R^2}} \\
&= \sqrt{\frac{(R^2-s^2-x^2)(1-\cos^2\phi)}{R^2}} \\
&= |\sin\phi|\sqrt{1-\frac{s^2+x^2}{R^2}} \\
&= \sin\phi\sqrt{1-\frac{s^2+x^2}{R^2}}
\end{aligned}$$

where the last step follows from the fact that $0 \le \phi = \cos^{-1}(\frac{y}{\sqrt{R^2-s^2-x^2}}) \le \pi/2$, for $0 \le y \le \sqrt{R^2-s^2-x^2}$.

Then,

$$\text{(A.4)}\quad q(x) = R\left[\sqrt{1-\frac{s^2+x^2}{R^2}}\right]^{K+1} \int_0^{\pi/2}\left[\sqrt{1-\frac{s^2+x^2}{R^2}}\right]^{K}\sin^K(\phi) I_K\left[\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}\sin\phi\right]\sin^{K+1}\phi\, d\phi.$$

Since $I_K(w) = (-i)^K J_K(iw)$ we have that

$$\text{(A.5)}\qquad q(x) = R\left[\sqrt{1-\frac{s^2+x^2}{R^2}}\right]^{K+1}(-i)^K\int_0^{\pi/2} J_K\left[i\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}\right]\sin^{K+1}\phi\, d\phi.$$

The integral in this equation is known as Sonine's integral whose general form and solution are [40]:

$$\int_0^{\pi/2} J_v(z\sin\phi)\cos^{2u+1}\phi\sin^{v+1}\phi\, d\phi = \frac{2^u\Gamma(u+1)}{z^{u+1}}J_{v+u+1}(z).$$

From (A.5) we find that

$$v = K$$
$$u = -\frac{1}{2}$$
$$z = i\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}$$

and therefore

$$q(x) = R\left[\sqrt{1-\frac{s^2+x^2}{R^2}}\right]^{K+1}(-i)^K\left\{\frac{2^{-1/2}\Gamma(1/2)}{\left(i\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}\right)^{1/2}}J_{K+\frac{1}{2}}\left(i\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}\right)\right\}.$$

Since $\Gamma(1/2) = \sqrt{\pi}$ and $(-i)^K = i^{-K}$, we obtain:

$$q(x) = R\sqrt{\frac{\pi}{2\alpha}}\left(\sqrt{1-\frac{s^2+x^2}{R^2}}\right)^{K+\frac{1}{2}}I_{K+\frac{1}{2}}\left(\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}\right).$$

Thus we have that Equation (A.2) reduces to:

$$\Psi(s) = \frac{4}{I_K(\alpha)}\left(R\sqrt{\frac{\pi}{2\alpha}}\right)\int_0^{\sqrt{R^2-s^2}}\left(\sqrt{1-\frac{s^2+x^2}{R^2}}\right)^{K+\frac{1}{2}} I_{K+\frac{1}{2}}\left(\alpha\sqrt{1-\frac{s^2+x^2}{R^2}}\right) d\,x.$$

Notice that this equation has the same form as Equation A.3 and can be solved following similar steps as above. Finally we obtain that

$$\Psi(s) = \frac{1}{I_K(\alpha)}\left\{2R\sqrt{\frac{\pi}{2\alpha}}\right\}^2\left(\sqrt{1-\left(\frac{s}{R}\right)^2}\right)^{K+1} I_{K+1}\left(\alpha\sqrt{1-\left(\frac{s}{R}\right)^2}\right). \tag{A.6}$$

Vita

Fernando Rodrigo Rannou was born in Santiago, Chile, on July 11, 1965. He attended Colegio San Marcos from 1970 to 1981, where he received the Licenciatura de Enseñanza Media. In March 1982, He entered the Engineering School at Universidad de Santiago de Chile (USACH). In the fifth year at USACH, he received the degree of Licenciado en Ciencias de la Ingeniería, and in March of 1988 he received the degree of Ingeniero Civil en Informática. The following Fall He joined the Departamento de Ingeniería en Informática as an instructor.

In January 1992, He entered the graduate program in the Department of Computer Science at The University of Tennessee, Knoxville. He was awarded the Master of Science degree in December of 1993 and the Doctor of Philosophy degree in May of 1998.

He is married to Jenny-Ann Marisol Morales Morrison.