Correction of Field Inhomogeneities in Low-Field MRI During Image Reconstruction

Image Distortion Correction

Bas Liesker

TU Delft

LUMC Leids Universitair Medisch Centrum

Correction of Field Inhomogeneities in Low-Field MRI During Image Reconstruction

Image Distortion Correction

by

Bas Liesker

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday May 26, 2021 at 13:00.

Student number:	4195590	
Project duration:	July 20, 2020 – May 26, 2021	
Thesis committee:	Dr. Ir. R. Remis,	TU Delft
	Dr. N.V. Budko,	TU Delft
	Dr. Ir. K. Koolstra,	LUMC
	BSc. T. O'Reilly,	LUMC
Supervisors:	Dr. Ir. R. Remis,	TU Delft
	Dr. Ir. K. Koolstra,	LUMC
	BSc. T. O'Reilly,	LUMC

This thesis is confidential and cannot be made public until May 26, 2021.

TU Delft

LUMC Leids Universitair Medisch Centrum

Abstract

Magnetic resonance imaging (MRI) scanners are a crucial diagnostic tool for radiologists. They are able to render two- and three-dimensional images of the body without exposure to harmful radiation. MRI systems are, however, costly to build and maintain. This adversely impacts access to these scanners in developing regions. In an effort to combat this problem, a low-field MRI scanner is being developed.

Conventional MRI scanners utilize a superconducting solenoid to generate the main magnetic field. The low-field scanner, on the other hand, induces the main magnetic field through a Hallbach array of permanent neodymium magnets. While beneficial for production and maintenance costs, as well as portability, the Hallbach array is not able to generate a perfectly homogeneous magnetic field.

The inhomogeneities present in the main magnetic field result in distortion of the images when reconstructed using conventional fast Fourier transform (FFT) methods. To counteract this, a reconstruction method that utilizes field information needs to be employed. In this thesis, existing methods to determine and utilize the field information to correct image distortion are explored. From this analysis, it is evident that model-based (MB) methods are most suitable for reconstruction of data from the low-field scanner. Current MB methods are only implemented for two-dimensional reconstruction. The goal of this thesis is to expand these methods to three-dimensional reconstruction.

A novel MB method for three-dimensional reconstruction is presented. This new method is able to circumvent memory constraints that arise from reconstruction of large data sets.

Though the new method requires several hours to reconstruct a $128 \times 128 \times 30$ data set, visual inspection indicates that an accurate result is achieved.

Acknowledgements

With the completion of this thesis, I have reached a milestone in my life. It has been a long and difficult journey, but worth every moment. There are several people who have helped me along the way who I want to acknowledge.

First and foremost, I would like to thank Dr. Ir. Rob Remis, whom without, I would not have been introduced to this project. As supervisor, he showed me which bureaucratic hoops I needed to jump through during the project, and motivated me to reach my deadlines. Together with Dr. Ir. Rob Remis, Dr. Ir. Kirsten Koolstra was present along every step of the way. She was always available for all questions and problems I had regarding the project. For that, I would like to thank her. I would also like to thank BSc. Tom O'Reilly, whom without, the low-field project would not be the success that it is. Furthermore, I am thankful for Dr. Neil Budko, who was able to fill a position in the thesis committee at the last moment.

I am grateful for all the friends I have made along the way, both in my study and student associations. Without them, my time in Delft would have been noticeably less enjoyable.

Despite my family living almost on the other side of the world, they have continually shown their support, for which I am eternally thankful.

Last but not least, I would like to thank my partner, who has always been there for me. She has motivated me to reach the point I am at now. I can always count on her support, even in stressful times like these.

Bas Liesker
Delft, May 18, 2021

Contents

List of Figures

List of Tables

List of Acronyms

AWGN	additive white Gaussian noise
CG	conjugate gradient
CGLS	conjugate gradient least squares
CPR	conjugate phase reconstruction
CS	compressed sensing
CSF	cerebrospinal fluid
DFT	discrete Fourier transform
EM	electromagnetic
FFT	fast Fourier transform
FID	free induction decay
FoV	field of view
MB	model-based
MFI	multifrequency interpolation
MRI	magnetic resonance imaging
NMR	nuclear magnetic resonance
PDF	probability density function
PET	positron emission tomography
RF	radio frequency
SNR	signal-to-noise ratio

1 Introduction

In the past century, the field of radiology has been intensely studied and researched, and now forms an integral part of the medical system. Radiology focuses mainly on diagnostics, by forming images of the body using a plethora of techniques. These techniques range from classic x-ray photography, to positron emission tomography (PET), to magnetic resonance imaging (MRI). Most MRI systems currently used in a clinical setting have main magnetic field strengths ranging from 1.5 to 7.0 teslas (T). While these systems have high diagnostic capabilities, they come with some drawbacks. This chapter will investigate the history and basic principles of MRI systems, after which it will introduce low-field MRI scanners and why they are of importance.

1.1. Magnetic Resonance Imaging

In 1972, the first images using nuclear magnetic resonance (NMR) were taken by Paul Lauterbur [1]. This laid the foundation to what is now known as MRI. Since then, the development in the MRI field has resulted in scanners that can image the brain with an isotropic resolution of 0.7mm [2]. This in turn has allowed doctors to diagnose afflictions such as small tumors and lesions that were previously undetectable.

MRI scanners, such as the one depicted in Figure 1.1, apply a strong homogeneous magnetic field (main magnetic field, B_0) to the patient inside the bore of the system. This magnetizes the body until it is removed from the scanner. Once magnetized, spatial variation is introduced in the form of linear gradient fields in the three Cartesian directions (G_x, G_y, and G_z). This allows for position-based signal acquisition.

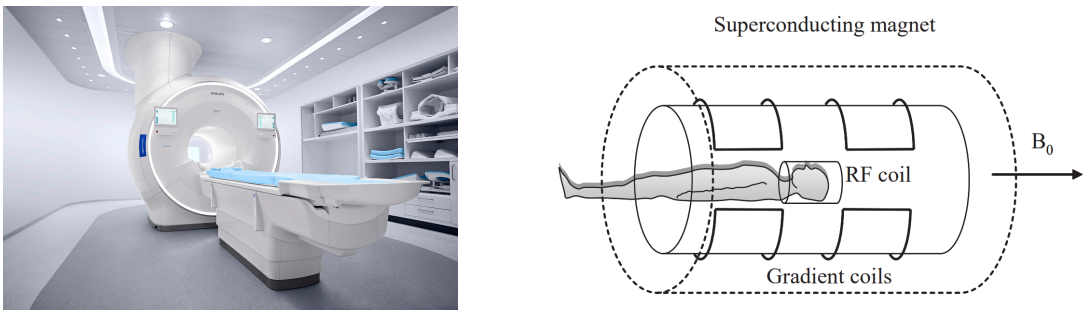

Figure 1.1: An example of a clinical 3.0T MRI scanner from Philips [3] (left) and the internal coils of a typical MRI system (right), taken from Smith and Webb [4].

To obtain signals from the body, radio frequency (RF) pulses are emitted to excite precessing protons, which then slowly return to their equilibrium state. As the protons return to their initial state, they emit electromagnetic (EM) energy that can be measured using coils. These coils can be designed such that they both transmit and receive RF pulses. Thanks to the gradient fields superimposed on the main magnetic field, the RF coils receive position-dependent signals, which, when transformed to the spatial domain, result in an image.

In high-field ($\geq$ 1.5T) MRI systems, the main magnetic field is produced by a superconducting solenoid around the bore. To achieve this, liquid helium continuously retains the temperature of the solenoid to below 9.3K, even when the scanner is not actively in use [5]. This produces a stable homogeneous field throughout the bore.

The gradient fields are produced by a separate set of coils, one pair for each direction on the Cartesian plane. To produce the gradient along the direction of the bore, a set of solenoids is typically used, while along the other two directions, intricate coils are needed to produce a linear magnetic field [6].

Together with the RF coils, the aforementioned solenoid and gradient coils are the main internal components of a typical high-field MRI scanner. The external components consist mainly of amplifiers and signal processing equipment. Lawrence Wald et al. estimated the relative costs of the main components for a 1.5T scanner with market price of $800,000 USD [5]. Figure 1.2 summarizes this cost distribution.

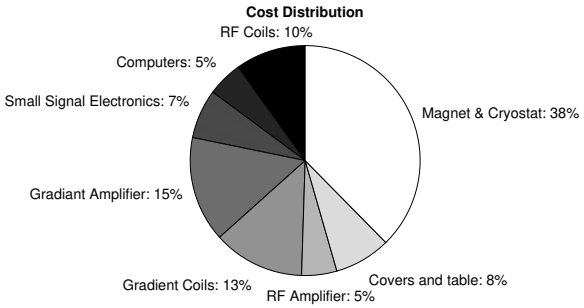

Figure 1.2: The cost distribution of the main components of a 1.5T MRI scanner. Adapted from Wald et al. [5].

Alongside the expenditure of acquiring an MRI system, there are significant costs for maintaining and housing the scanner. Each scanner needs to be housed in a specialized room that electromagnetically shields the system, both from outgoing and incoming signals. This is necessary in order to prevent interference with external equipment and disturbances in the acquired signals. These rooms can cost up to $100,000 USD [5].

The high costs of obtaining and maintaining MRI scanners has lead to a gap in the number of scanners per capita between the developed and developing world (see Figure 1.3). This discrepancy adversely affects the average lifespan of citizens from developing countries [7].

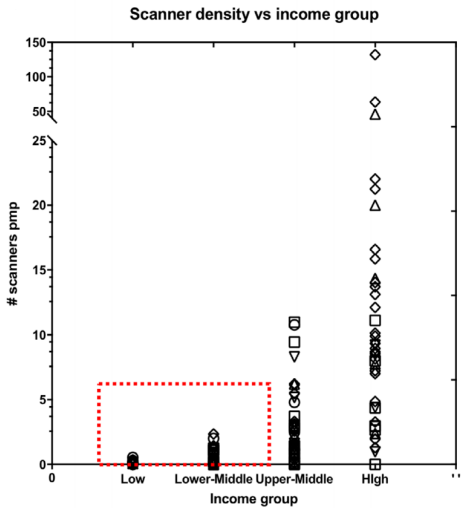

Figure 1.3: Scanner availability (scanners per million people) based on income. Taken from Geethanath et al. [7].

1.2. Low-Field MRI

In an attempt to make MRI systems more affordable and accessible, low-field (< 0.5T) scanners are currently actively researched and developed [5], [8]–[10]. The most significant difference between high-field and low-field systems is the removal of the superconducting solenoid. To produce the main magnetic field, some of these systems instead opt for an array of permanent neodymium magnets in a cylindrical Halbach formation, such as the system developed by Cooley et al. [9]. Another possible technique for inducing a strong magnetic field is the neodymium dipole design, introduced by Miyamoto et al. [11]. Despite the strong field produced by this design (between 0.2 and 0.35T [5]), the mass of the permanent magnets limits the mobility of these systems. In an effort to increase portability, Nakagomi et al. have recently developed a car-mounted adaptation of the dipole system with a 200kg permanent magnet [12]. Low-field scanners with a Hallbach array, on the other hand, have a lighter permanent magnet design, such as the one developed by O'Reilly et al., which weighs 75kg [10].

With the replacement of the superconducting solenoid and removal of its cooling system, the manufacturing and maintenance costs have been significantly reduced. As an added benefit, these scanners employ shielding directly around the system, eliminating the need for a specialized room for housing the scanner. This results in a smaller footprint and increased portability.

Due to cost savings and the limitations of permanent magnets, low-field scanners come with a significant drawback: decreased signal-to-noise ratio (SNR) and therefore lower resolution compared to high-field scanners. This limits the diagnostic capabilities of the scanner. Despite this drawback, these scanners can still be used as an early detection system to find physiological conditions that, without a scanner, might go unnoticed.

1.3. Neonatal Hydrocephalus

One possible use case of low-field MRI scanners is the detection of neonatal hydrocephalus. Hydrocephalus is characterized by the accumulation of cerebrospinal fluid (CSF) within the brain [13]. This build-up of fluid is caused by an obstruction in the brain [14].

Due to lower sanitary conditions and less access to health care, hydrocephalus is significantly more prevalent in children in developing countries as opposed to children in developed countries [14]. Low-field scanners, with their reduced costs and increased portability, provide an opportunity for early detection of neonatal hydrocephalus in developing countries. Previously, these systems have already been used in the diagnosis of hydrocephalus in dogs [15], suggesting the possibility of detection in children.

With the current development of low-field scanners, the SNR and accuracy of acquired images is gradually increasing. In time, this will lead to diagnostic capabilities of increasingly complex physiological conditions.

1.4. Thesis Objective

In this thesis, the low-field scanner developed by O'Reilly et al. [10], depicted in Figure 1.4, will be investigated. Currently, the scanner exhibits field disturbances due to inhomogeneities that are more prevalent than desired. There are three main components that contribute to the inhomogeneity of the fields:

- Imperfect Halbach array
- Non-linearity of the gradient coils
- Thermal effects

These field inhomogeneities lead to spatial distortion of the imaged volumes.

Current reconstruction techniques are either ineffective at completely correcting image distortion due to field disturbances, or require additional scanning time. Hence, this thesis will focus on reconstruction efficiency, both in terms of processing time and accuracy. The techniques presented by Koolstra et al. [16] and de Leeuw den Bouter et al. [17] will form the basis for the new algorithms. The objective of this thesis is to answer the following question:

How can image distortion due to field inhomogeneities effectively be corrected in 3D imaging volumes?

Figure 1.4: The low-field (50mT) scanner developed by O'Reilly et al. [10].

The goal is to correct for the aforementioned field disturbances, as well as to implement the new reconstruction technique on the scanner.

1.5. Thesis Structure

Chapter 2 will introduce the structure of the obtained signal. Following that, Chapter 3 explains the existing methods used to correct for image distortion. Next, Chapter 4 investigates how model-based (MB) reconstruction performs on single- and multi-slice data. Chapter 5 highlights what changes need to be made to two-dimensional reconstruction in order to be able to process three-dimensional data sets. Afterwards, Chapter 6 will investigate the effect of regularization, after which Chapter 7 will give a comparison of two- and three-dimensional reconstruction on various data sets. Finally, Chapter **??** will briefly inform how the algorithm was implemented and tested, followed by Chapter 8, which will form a conclusion to the thesis, and highlight some recommendations.

2

Signal Model

To acquire an image using MRI, a signal describing the volume in question is required. The signal describes the susceptibility to magnetization, which in turn describes the proton density distribution, resulting in contrast between different media (such as fat and water). This chapter will describe how inducing specific magnetic fields leads to a signal that can be transformed into an accurate image of a volume. The chapter begins with the Bloch equation and concludes with the basic signal model.

2.1. Bloch Equation

The Bloch equation forms the basis of signal acquisition. It describe the magnetization of a volume based on the applied magnetic field. Table 2.1 lists the symbols utilized in the Bloch equation when a Cartesian reference frame is used.

Table 2.1: The symbols used in the Bloch equation in the Cartesian reference frame.

Symbol	Description
$\mathbf{M}$	Magnetization
$\mathbf{B}$	Magnetic Field
$\mathbf{i}_x$, $\mathbf{i}_y$, and $\mathbf{i}_z$	Basis Vectors of unit length
$\mathbf{r}$	Position Vector
t	Time
T_1	Longitudinal Relaxation Time
T_2	Transverse Relaxation Time
γ	Gyromagnetic Ratio

Initially ($t = 0$), when only the background field is present, the magnetization of the volume is:

$$\mathbf{M}^0(\mathbf{r}, 0) = \mathbf{M}_x^0(\mathbf{r})\mathbf{i}_x + \mathbf{M}_y^0(\mathbf{r})\mathbf{i}_y + \mathbf{M}_z^0(\mathbf{r})\mathbf{i}_z \tag{2.1}$$

The main magnetic field, $\mathbf{B}_0$, is a static background field along the longitudinal direction (z-direction).

$$\mathbf{B}_0(\mathbf{r}) = B_0(\mathbf{r})\mathbf{i}_z \tag{2.2}$$

In the time interval $t = [0, t_f)$, the magnetic field is the sum of the background field and the applied field, $\mathbf{B}_a$.

$$\mathbf{B}(\mathbf{r}, t) = \mathbf{B}_0(\mathbf{r}) + \mathbf{B}_a(\mathbf{r}, t) \tag{2.3}$$

Bloch's equation is [18]:

$$\frac{\partial \mathbf{M}(\mathbf{r},t)}{\partial t} + \gamma \mathbf{B}(\mathbf{r},t) \times \mathbf{M}(\mathbf{r},t) + \frac{1}{T_2(\mathbf{r})}\mathbf{M}_{\perp}(\mathbf{r},t) + \frac{1}{T_1(\mathbf{r})}\mathbf{M}_{\parallel}(\mathbf{r},t) = \frac{1}{T_1(\mathbf{r})}M^{eq}(\mathbf{r})\mathbf{i}_z \tag{2.4}$$

The longitudinal and transverse magnetization are, respectively:

$$\begin{aligned} \mathbf{M}_{\parallel}(\mathbf{r},t) &= \mathbf{M}_z(\mathbf{r},t)\mathbf{i}_z \\ \mathbf{M}_{\perp}(\mathbf{r},t) &= \mathbf{M}_x(\mathbf{r},t)\mathbf{i}_x + \mathbf{M}_y(\mathbf{r},t)\mathbf{i}_y \end{aligned} \tag{2.5}$$

The equilibrium magnetization is:

$$M^{eq}(\mathbf{r}) = \rho(\mathbf{r})\frac{\gamma^2\hbar^2}{4k_B T}B_0(\mathbf{r}) \tag{2.6}$$

where ρ is the spin density, $\hbar$ Planck's constant divided by 2π, k_B the Boltzmann's constant, and T the absolute temperature. For protons, the gyromagnetic ratio is approximately: $42.577 \cdot 10^6$ Hz T^{-1} or $2.675 \cdot 10^8$ rad $\mathrm{s}^{-1}\mathrm{T}^{-1}$ [19].

2.1.1. Rotating Reference Frame

To find the general set of solutions to the Bloch equation, a rotating reference frame is introduced.

$$\begin{aligned} \mathbf{i}'_x &= cos(\omega_0(\mathbf{r})t)\mathbf{i}_x - sin(\omega_0(\mathbf{r})t)\mathbf{i}_y \\ \mathbf{i}'_y &= sin(\omega_0(\mathbf{r})t)\mathbf{i}_x + cos(\omega_0(\mathbf{r})t)\mathbf{i}_y \\ \mathbf{i}'_z &= \mathbf{i}_z \end{aligned} \tag{2.7}$$

The magnetization in the rotating reference frame is then:

$$\begin{aligned} M'_x &= \mathbf{M}\cdot\mathbf{i}'_x = cos(\omega_0(\mathbf{r})t)M_x - sin(\omega_0(\mathbf{r})t)M_y \\ M'_y &= \mathbf{M}\cdot\mathbf{i}'_y = sin(\omega_0(\mathbf{r})t)M_x + cos(\omega_0(\mathbf{r})t)M_y \\ M'_z &= \mathbf{M}\cdot\mathbf{i}'_z = M_z \end{aligned} \tag{2.8}$$

Now in vector form:

$$m = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} \qquad m' = \begin{bmatrix} M'_x \\ M'_y \\ M'_z \end{bmatrix} \tag{2.9}$$

To transform to and from the rotating reference frame, the rotation matrix, R, is used.

$$\begin{aligned} m' &= Rm \\ m &= R^T m' \end{aligned} \tag{2.10}$$

$$R = \begin{bmatrix} cos(\omega_0(\mathbf{r})t) & sin(\omega_0(\mathbf{r})t) & 0 \\ -sin(\omega_0(\mathbf{r})t) & cos(\omega_0(\mathbf{r})t) & 0 \\ 0 & & 1 \end{bmatrix} \tag{2.11}$$

The components of the Bloch equation in the stationary frame are:

$$\begin{aligned}\frac{\partial M_x}{\partial t} - \omega_0(\mathbf{r})M_y + \frac{1}{T_2} &= 0\\ \frac{\partial M_y}{\partial t} + \omega_0(\mathbf{r})M_x + \frac{1}{T_2} &= 0\\ \frac{\partial M_z}{\partial t} + \frac{M_z}{T_1} &= \frac{M^{eq}}{T_1}\end{aligned} \tag{2.12}$$

In matrix form:

$$\frac{\partial m}{\partial t} + \omega_0(\mathbf{r})\begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}m + \begin{bmatrix}\frac{1}{T_2} & 0 & 0\\ 1 & \frac{1}{T_2} & 0\\ 0 & 0 & \frac{1}{T_1}\end{bmatrix}m = \begin{bmatrix}0\\ 0\\ \frac{M^{eq}}{T_1}\end{bmatrix} \tag{2.13}$$

Substitute $m = Rm'$:

$$R\frac{\partial m'}{\partial t} + \frac{\partial R}{\partial t}m' + \omega_0(\mathbf{r})\begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}Rm' + \begin{bmatrix}\frac{1}{T_2} & 0 & 0\\ 1 & \frac{1}{T_2} & 0\\ 0 & 0 & \frac{1}{T_1}\end{bmatrix}Rm' = \begin{bmatrix}0\\ 0\\ \frac{M^{eq}}{T_1}\end{bmatrix} \tag{2.14}$$

Since:

$$\frac{\partial R}{\partial t} + \omega_0(\mathbf{r})\begin{bmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}R = 0 \tag{2.15}$$

The system becomes:

$$R\frac{\partial m'}{\partial t} + \begin{bmatrix}\frac{1}{T_2} & 0 & 0\\ 1 & \frac{1}{T_2} & 0\\ 0 & 0 & \frac{1}{T_1}\end{bmatrix}Rm' = \begin{bmatrix}0\\ 0\\ \frac{M^{eq}}{T_1}\end{bmatrix} \tag{2.16}$$

Premultiply by R^T:

$$\frac{\partial m'}{\partial t} + \begin{bmatrix}\frac{1}{T_2} & 0 & 0\\ 1 & \frac{1}{T_2} & 0\\ 0 & 0 & \frac{1}{T_1}\end{bmatrix}m' = \begin{bmatrix}0\\ 0\\ \frac{M^{eq}}{T_1}\end{bmatrix} \tag{2.17}$$

The individual components of the system are:

$$\begin{aligned}\frac{\partial M'_x}{\partial t} + \frac{M'_x}{T_2} &= 0\\ \frac{\partial M'_y}{\partial t} + \frac{M'_y}{T_2} &= 0\\ \frac{\partial M'_z}{\partial t} + \frac{M'_z}{T_1} &= \frac{M^{eq}}{T_1}\end{aligned} \tag{2.18}$$

The solutions to the first-order partial differential equations are of the form:

$$\begin{aligned} M_x' &= Ae^{(-t/T_2)} \\ M_y' &= Be^{(-t/T_2)} \\ M_z' &= Ce^{(-t/T_1)} + \left(1 - e^{(-t/T_1)}\right) M^{eq} \end{aligned} \tag{2.19}$$

Using the initial conditions, the solution to the Bloch equation in the rotating reference frame becomes:

$$\begin{aligned} M_x' &= M_x^0 e^{(-t/T_2)} \\ M_y' &= M_y^0 e^{(-t/T_2)} \\ M_z' &= M_z^0 e^{(-t/T_1)} + \left(1 - e^{(-t/T_1)}\right) M^{eq} \end{aligned} \tag{2.20}$$

In the stationary reference frame, the solution to the Bloch equation is (using equation 2.10):

$$\begin{aligned} M_x &= e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_x^0 + sin(\omega_0(\mathbf{r})t)M_y^0\right] \\ M_y &= e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_y^0 - sin(\omega_0(\mathbf{r})t)M_x^0\right] \\ M_z &= M_z^0 e^{(-t/T_1)} + \left(1 - e^{(-t/T_1)}\right) M^{eq} \end{aligned} \tag{2.21}$$

2.2. Voltage Signal

After spending some time in the main magnetic field, the volume of interest enters a state of equilibrium magnetization. An RF pulse forces the volume to leave this state, after which it will gradually return to equilibrium. During this transition back to equilibrium, the precessing protons emit energy in the form of EM waves. This results in a free induction decay (FID) signal which can be measured by the same (or different) set of coils that transmitted the RF pulse.

The induced potential difference, V, is given by [18]:

$$\begin{aligned} V(t) &= -\int_{\mathbf{r}\in\mathbb{D}} \frac{\partial \mathbf{M}(\mathbf{r},t)}{\partial t} \cdot \mathbf{B}_r(\mathbf{r})\, d\mathbf{r} \\ V(t) &= -\int_{\mathbf{r}\in\mathbb{D}} \left(\frac{\partial M_x}{\partial t}B_{r,x} + \frac{\partial M_y}{\partial t}B_{r,y} + \frac{\partial M_z}{\partial t}B_{r,z}\right) d\mathbf{r} \end{aligned} \tag{2.22}$$

where $\mathbf{B}_r$ is the received field. Using equation 2.21, the derivatives are:

$$\begin{aligned} \frac{\partial M_x}{\partial t} &= \frac{-1}{T_2} e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_x^0 + sin(\omega_0(\mathbf{r})t)M_y^0\right] + \omega_0(\mathbf{r}) e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_y^0 - sin(\omega_0(\mathbf{r})t)M_x^0\right] \\ \frac{\partial M_y}{\partial t} &= \frac{-1}{T_2} e^{(-t/T_2)} \left[sin(\omega_0(\mathbf{r})t)M_x^0 + cos(\omega_0(\mathbf{r})t)M_y^0\right] - \omega_0(\mathbf{r}) e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_x^0 + sin(\omega_0(\mathbf{r})t)M_y^0\right] \\ \frac{\partial M_z}{\partial t} &= \frac{1}{T_1} e^{(-t/T_1)} \left(M^{eq} - M_z^0\right) \end{aligned} \tag{2.23}$$

Since $\omega_0(\mathbf{r}) \gg \frac{1}{T_2}$ and $\omega_0(\mathbf{r}) \gg \frac{1}{T_1}$:

$$\begin{aligned} \frac{\partial M_x}{\partial t} &\approx \omega_0(\mathbf{r}) e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_y^0 - sin(\omega_0(\mathbf{r})t)M_x^0\right] \\ \frac{\partial M_y}{\partial t} &\approx -\omega_0(\mathbf{r}) e^{(-t/T_2)} \left[cos(\omega_0(\mathbf{r})t)M_x^0 + sin(\omega_0(\mathbf{r})t)M_y^0\right] \\ \frac{\partial M_z}{\partial t} &\approx 0 \end{aligned} \tag{2.24}$$

The voltage signal then becomes:

$$V(t) \approx -\int_{\mathbf{r}\in\mathbb{D}} \omega_0(\mathbf{r}) e^{(-t/T_2)} \Big(\left[cos(\omega_0(\mathbf{r})t)M_y^0 - sin(\omega_0(\mathbf{r})t)M_x^0\right] B_{r,x} - \left[cos(\omega_0(\mathbf{r})t)M_x^0 + sin(\omega_0(\mathbf{r})t)M_y^0\right] B_{r,y} \Big)\, d\mathbf{r} \tag{2.25}$$

Now let $M_0^{\pm} = M_x^0 \pm jM_y^0$ and $B_r^{\pm} = B_{r,x} \pm jB_{r,y}$. The signal then takes the form:

$$V(t) \approx \frac{j}{2} \int_{\mathbf{r}\in\mathbb{D}} \omega_0(\mathbf{r}) e^{(-t/T_2)} \left(e^{(-j\omega_0(\mathbf{r})t)} M_0^+ B_{r,x}^- - e^{(j\omega_0(\mathbf{r})t)} M_0^- B_{r,x}^+ \right)\, d\mathbf{r} \tag{2.26}$$

The signal is then modulated and amplified, after which it is passed through a low-pass filter:

$$\begin{aligned} V_{mod}(t) &= 2e^{(j\omega_{mod}t)} V(t) \\ V_{mod}(t) &= j \int_{\mathbf{r}\in\mathbb{D}} \omega_0(\mathbf{r}) e^{(-t/T_2)} \left(e^{(-j(\omega_0(\mathbf{r})-\omega_{mod})t)} M_0^+ B_{r,x}^- - e^{(j(\omega_0(\mathbf{r})+\omega_{mod})t)} M_0^- B_{r,x}^+ \right)\, d\mathbf{r} \\ V_{LPF}(t) &= j \int_{\mathbf{r}\in\mathbb{D}} \omega_0(\mathbf{r}) e^{(-t/T_2)} e^{(-j\Delta\omega_0(\mathbf{r})t)} M_0^+ B_{r,x}^-\, d\mathbf{r} \end{aligned} \tag{2.27}$$

with $\Delta\omega_0(\mathbf{r}) = \omega_0(\mathbf{r}) - \omega_{mod}$.

2.3. Signal Structure

With the voltage model in hand, the signal structure can be derived. The received signal is of the form:

$$S(t) = \int_{\mathbf{r}\in\mathbb{D}} \omega_0(\mathbf{r}) e^{-t/T_2} e^{-j\Delta\omega_0(\mathbf{r})t} M_0^+ B_{r,x}^-\, d\mathbf{r} \tag{2.28}$$

The sensitivity of the receiver coil, $B_{r,x}^-$, is assumed to be spatially invariant, hence eliminating this term from the integrand.

$$S(t) \propto \int_{\mathbf{r}\in\mathbb{D}} \omega_0(\mathbf{r}) e^{-t/T_2} e^{-j\Delta\omega_0(\mathbf{r})t} M_0^+\, d\mathbf{r} \tag{2.29}$$

In the presence of magnetic fields, the magnetization of the volume can be described with a spin density map, $\rho(\mathbf{r})$. The signal in terms of spin density is:

$$S(t) \propto \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-t/T_2} e^{-j\Delta\omega_0(\mathbf{r})t}\, d\mathbf{r} \tag{2.30}$$

Ignoring transverse relaxation effects, the intermediate signal model becomes:

$$\begin{aligned} S(t) &= \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\Delta\omega_0(\mathbf{r})t}\, d\mathbf{r} \\ S(t) &= \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t}\, d\mathbf{r} \end{aligned} \tag{2.31}$$

2.3.1. Spatial Encoding

Gradient fields are linear fields in the x-, y-, and z-direction, produced by their respective coils. These fields introduce spatial encoding in the acquired signal. The fields are defined as:

$$G_x = \frac{\partial B_z}{\partial x} \qquad G_y = \frac{\partial B_z}{\partial y} \qquad G_z = \frac{\partial B_z}{\partial z} \tag{2.32}$$

The frequency-encoded one-dimensional signal then becomes:

$$S(t) = \int_{x\in\mathbb{D}} \rho(x) e^{-j\gamma\Delta B_0(x)t} e^{-j\gamma G_x x t}\, dx \tag{2.33}$$

With the addition of phase encoding, the signal takes the form of:

$$S(t,\tau) = \int_{x\in\mathbb{D}} \int_{y\in\mathbb{D}} \rho(x,y) e^{-j\gamma\Delta B_0(x,y)t} e^{-j\gamma G_x x t} e^{-j\gamma G_y y \tau}\, dydx \tag{2.34}$$

Or equivalently, when the position is vectorized, the signal per phase encoding step becomes:

$$S_\tau(t) = \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau}\, d\mathbf{r} \tag{2.35}$$

The total signal in k-space notation is:

$$S(t) = \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j2\pi\mathbf{k}(t)\cdot\mathbf{r}}\, d\mathbf{r} \tag{2.36}$$

2.3.2. Discretization

The continues range of spatial points over which the signal is acquired can be transformed to a set of discrete points, if the step size, $\Delta\mathbf{r}$, is sufficiently small. This discretization leads to a signal that, for every time sample, is a summation over the complete set of discrete spatial points. Hence, the signal is rewritten as:

$$\begin{aligned} S_\tau(t) &= \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau}\, d\mathbf{r} \\ S_\tau(t) &= \lim_{\Delta\mathbf{r}\to 0} \sum_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau}\, \Delta\mathbf{r} \\ S_\tau(t) &\propto \sum_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau} \end{aligned} \tag{2.37}$$

2.3.3. Linear System

The acquired signal can be written as a set of linear equations. Per phase encoding step, the system takes the form:

$$\mathbf{S}_\tau = E_\tau \boldsymbol{\rho} \tag{2.38}$$

Where the signal and original image are:

$$\mathbf{S}_\tau = \begin{bmatrix} S_\tau(t_0) \\ S_\tau(t_1) \\ \vdots \\ S_\tau(t_{N-1}) \end{bmatrix} \qquad \boldsymbol{\rho} = \begin{bmatrix} \rho(r_0) \\ \rho(r_1) \\ \vdots \\ \rho(r_{p-1}) \end{bmatrix} \tag{2.39}$$

with $p = M \cdot N$ the total number of encoding points, M the number of phase encoding steps, and N the number of readout samples per phase encoding step. The system matrix for one phase encoding step is:

$$E_\tau = \begin{bmatrix} e^{-j\gamma((\Delta B(r_0)+G_x(r_0))t_0+G_y(r_0)\tau)} & \dots & e^{-j\gamma((\Delta B(r_{p-1})+G_x(r_{p-1}))t_0+G_y(r_{p-1})\tau)} \\ e^{-j\gamma((\Delta B(r_0)+G_x(r_0))t_1+G_y(r_0)\tau)} & \dots & e^{-j\gamma((\Delta B(r_{p-1})+G_x(r_{p-1}))t_1+G_y(r_{p-1})\tau)} \\ \vdots & \vdots & \ddots \\ e^{-j\gamma((\Delta B(r_0)+G_x(r_0))t_{N-1}+G_y(r_0)\tau)} & \dots & e^{-j\gamma((\Delta B(r_{p-1})+G_x(r_{p-1}))t_{N-1}+G_y(r_{p-1})\tau)} \end{bmatrix} \tag{2.40}$$

The total system matrix is then obtained by stacking the system matrix of each phase encoding step.

3

Current Methods

The efficient fast Fourier transform (FFT), based on the algorithm introduced by Cooley et al., is the classic approach for transforming an acquired signal to the spatial domain [20]. This method is suited for recovering images when the main magnetic field is homogenious [21]. When the main magnetic field displays strong inhomogeneities, the FFT method can lead to deformation in the recovered image [16]. Hence, other techniques for image reconstruction are required.

As of this moment, there are several methods that attempt to correct for image distortion due to field inhomogeneities. In this chapter, some of the existing methods will be discussed, both in terms of advantages and short-comings.

3.1. Field Map

Existing image deformation correction methods currently depend on the field map of the B_0 field. This field map describes the field at every coordinate. Ideally, the main magnetic field is spatially invariant, thus leading to a field map that is constant throughout the volume. However, due to inhomogeniety in the B_0 field, there is a spatial dependence in the field strength. An example of such a field map is illustrated in Figure 3.1. Future field maps will be depicted in hertz.

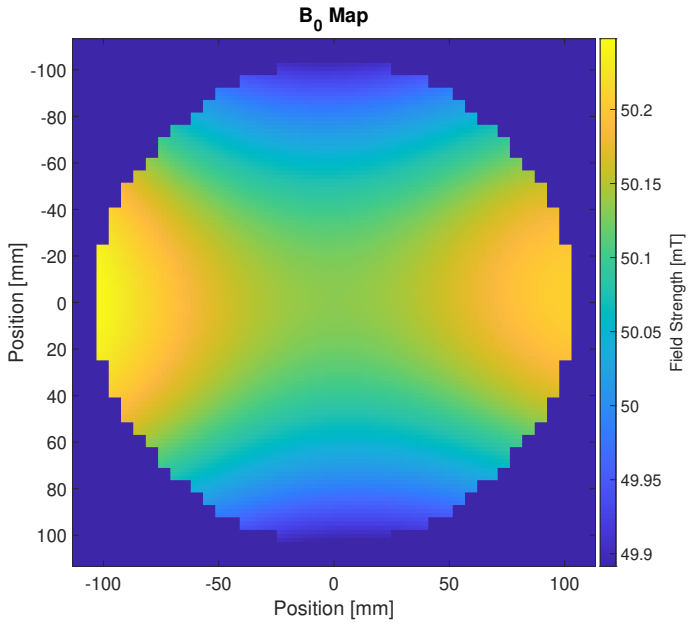

Figure 3.1: An example of a field map.

In subsequent chapters, the field map will depict the difference between the field strength at any point in the field of view (FoV), and the center field strength. That is:

$$\begin{aligned}\Delta B_0(\mathbf{r}) &= B_0(\mathbf{r}) - B_{center} \\ \Delta\omega_0(\mathbf{r}) &= \gamma \Delta B_0(\mathbf{r})\end{aligned} \tag{3.1}$$

3.1.1. Phase Difference

Acquiring the field map can be achieved in two ways, mathematically or empirically. Mathematically, the field inhomogeneities result in a shift in the readout direction. Recall the signal model:

$$S_\tau(t) = \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau}\, dr \tag{3.2}$$

One possible method to obtain the field map, $\Delta B(\mathbf{r})$, is by performing two scans on the same volume [22]. In the second scan, a time-shift, t_{shift}, is introduced in the readout gradient. Alternatively, a dual-echo gradient echo scan can be employed to decrease the scanning time [23].

The signal of the second scan is described as:

$$S_{2,\tau}(t) = \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})(t+t_{shift})} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau}\, dr \tag{3.3}$$

Once the signals have been transformed to the spatial domain, the field map can be found using the phase difference in the two images.

$$\begin{aligned} S_1(\tau,t) &\xrightarrow{\mathcal{F}^{-1}} I_1(\mathbf{r}) \\ S_2(\tau,t) &\xrightarrow{\mathcal{F}^{-1}} I_2(\mathbf{r}) \end{aligned} \tag{3.4}$$

$$\Delta B_0(\mathbf{r}) = \frac{\angle I_2 - \angle I_1}{t_{shift}} \tag{3.5}$$

Where $S(\tau,t)$ takes the form:

$$S(\tau,t) = \begin{bmatrix} \mathbf{S}_{\tau_0}^T \\ \mathbf{S}_{\tau_1}^T \\ \vdots \\ \mathbf{S}_{\tau_{M-1}}^T \end{bmatrix} \tag{3.6}$$

Though this method can quickly obtain the field map once both scans are complete, its accuracy is susceptible to noise. In typical high-field scanners, the SNR is sufficient to generate field maps using this method. Low-field scanners, on the other hand, have reduced SNR, leading to inaccurate field maps using this approach.

There are several methods that attempt to alleviate field map inaccuracies. One such method, developed by Funai et al. [24], uses multiple scans, each with a unique time-shift in the readout gradient, in conjunction with regularization. Alternatively, a basis of spherical harmonics can be employed to obtain a noiseless field map of the initial, noisy field map estimate, as shown by Koolstra et al. [16]. This mitigates the need for more than two scans, thus not adversely affecting the total scan time.

3.1.2. Empirical Measurements

While the performance of the aforementioned analytical method depends on the quality of the obtained images, the empirical method does not. Empirically, the field map can be acquired using a gaussmeter connected to a 3D positioning robot. This results in a field map that, depending on the gaussmeter, can be an accurate representation of the actual field inside the bore. Though accurate, the measurements only provide a snapshot of the current field map. Due to thermal effects, the field map varies over time, eventually resulting in obsolete measurements. To reacquire the measurements during scanning with a gaussmeter is infeasible, in part due to significant amount of time needed for each set of measurements, as well as the presence of the imaging volume in the bore. Therefore, obtaining the field map through the aforementioned mathematical method is preferential.

3.2. Conjugate Phase Reconstruction

After the field map has been acquired, the image distortion can be compensated in the reconstruction process. There are several methods that attempt to correct the image distortion. One such method is conjugate phase reconstruction (CPR). In the CPR approach, the inverse signal model is approximated as:

$$\rho_{cp}(\mathbf{r}) \approx \int_0^T S(t) e^{j\gamma \Delta B_0(\mathbf{r})t} e^{j2\pi \mathbf{k}(t)\cdot \mathbf{r}}\, dt \tag{3.7}$$

With T the length of the readout interval.

This method reconstructs each pixel individually, leading to long computation times. To alleviate this, Man et al. developed a multifrequency interpolation (MFI) technique that requires significantly less computations [25]. In the proposed algorithm, only $L+1$ inverse FFTs are required, where L is:

$$L > 8\Delta B_{max} T \tag{3.8}$$

The recovered image is then a linear combination of the $L+1$ inverse FFTs:

$$\rho(\mathbf{r}) \approx \sum_{i=0}^{L} c_i(\Delta B(\mathbf{r})) \int_0^T S(t) e^{j\gamma \Delta B_{0_i} t} e^{j2\pi \mathbf{k}(t)\cdot \mathbf{r}}\, dt \tag{3.9}$$

In the case that the center frequency, $\Delta B_{0_{L/2}}$ is zero, the coefficients, $c_i(\Delta B(\mathbf{r}))$, can be taken to be the $L+1$ lowest discrete Fourier transform (DFT) coefficients of $e^{j\gamma \Delta B_0 t_k}$.

3.3. Model-Based Reconstruction

While CPR provides a fast approximation for the reconstruction, MB reconstruction delivers more precise distortion correction. Unlike CPR, MB reconstruction does not rely on an approximation of the inverse signal model. Instead, MB reconstructs the image using iterative algorithms.

In MB reconstruction, the linear system $\mathbf{S} = E\boldsymbol{\rho}$ is considered. Here, the system matrix, E, is of the form: $E \in \mathbb{C}^{N^2 \times M^2}$, while the vectorized signal and image matrices take the form: $\mathbf{S} \in \mathbb{C}^{N^2 \times 1}$ and $\boldsymbol{\rho} \in \mathbb{C}^{M^2 \times 1}$.

To obtain the unknown image $\boldsymbol{\rho}$, minimization is employed. The minimization problem is of the form:

$$\hat{\boldsymbol{\rho}} = \underset{\boldsymbol{\rho}}{\operatorname{argmin}} \|E\boldsymbol{\rho} - \mathbf{S}\|_2^2 \tag{3.10}$$

Due to the nature of the field inhomogeneities and presence of noise, the minimization problem is often ill-posed. To counteract this, regularization in the form of a penalty term can be employed in the cost function. One possible minimization formulation that includes regularization is:

$$\hat{\boldsymbol{\rho}} = \underset{\boldsymbol{\rho}}{\operatorname{argmin}} \left\{ \frac{\mu}{2} \|E\boldsymbol{\rho} - \mathbf{S}\|_2^2 + \frac{\lambda}{2} \left(\|\nabla_x \boldsymbol{\rho}\|_1 + \|\nabla_y \boldsymbol{\rho}\|_1 \right) \right\} \tag{3.11}$$

This can be solved using, for example, the Split Bregman scheme.

Alternatively, Tikhonov regularization, which does not require an L1 norm minimization, can be employed. The minimization problem with Tikhonov regularization takes the form:

$$\begin{aligned} \hat{\boldsymbol{\rho}} &= \underset{\boldsymbol{\rho}}{\operatorname{argmin}} \left\{ \|E\boldsymbol{\rho} - \mathbf{S}\|_2^2 + \|\Gamma\boldsymbol{\rho}\|_2^2 \right\} \\ \Gamma &= \alpha I \end{aligned} \tag{3.12}$$

where I is the identity matrix and α the regularization parameter.

There are a myriad of possible regularization schemes and iterative solvers. De Leeuw den Bouter et al. [17] have investigated several of those regularizers and solvers.

3.3.1. With Compressed Sensing

Compressed sensing (CS) can be employed to reduce imaging time. In CS, a diagonal sampling matrix, $R \in \mathbb{R}^{N^2 \times N^2}$, is introduced in the model. The minimization problem then becomes:

$$\hat{\boldsymbol{\rho}} = \underset{m}{\operatorname{argmin}} \left\{ \frac{\mu}{2} \|RE\boldsymbol{\rho} - \mathbf{S}\|_2^2 + \frac{\lambda}{2} \left(\|\nabla_x \boldsymbol{\rho}\|_1 + \|\nabla_y \boldsymbol{\rho}\|_1 \right) \right\} \tag{3.13}$$

3.4. Effectiveness of CPR and MB

Koolstra et al. [16] have investigated the effectiveness of the CPR and MB reconstruction methods on both simulated and in vivo data. Figure 3.2 gives an overview of the reconstruction results, together with the outcome of the FFT method.

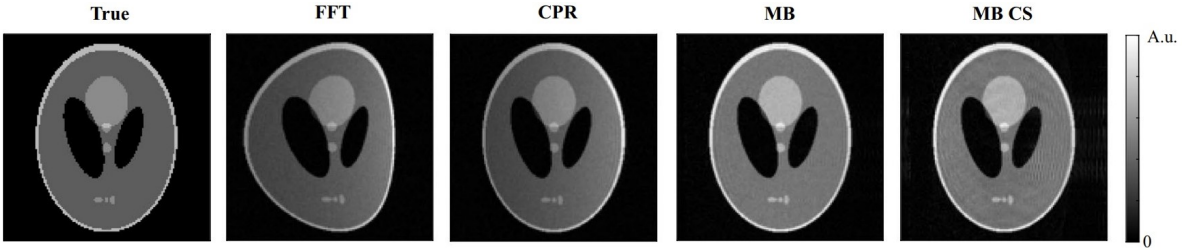

Figure 3.2: The reconstruction results of the FFT, CPR, and MB techniques on a distorted Shepp-Logan phantom. Image adapted from Koolstra et al. [16].

From Figure 3.2 it is apparent that both the CPR and MB methods are able to recover the original shape of the phantom. Furthermore, the MB method is able to provides a more accurate reconstruction of the pixel intensities compared to the CPR method.

3.5. Density Correction

Sekihara et al. [22] propose to correct for distortion by applying density correction to the acquired image. The two-dimensional distorted image, $S(x', y')$, is of the form:

$$\begin{aligned} S(x', y') &= \rho(x, y)/W(x, y) \\ x' &= x \\ y' &= y + \frac{\Delta B_0(x, y)}{G_y} \\ W(x, y) &= 1 + \frac{\partial \Delta B_0(x, y)}{\partial y} \frac{1}{G_y} \end{aligned} \tag{3.14}$$

The original image, $S_r(I,J)$, is then resolved using discrete pixel units I and J as follows:

$$\begin{aligned} S_r(I,J) &= S_g(I,J)W(I,J) \\ W(I,J) &= 1 + \frac{\Delta B_0(I,J+1) - \Delta B_0(I,J)}{G_y p_y} \\ S_g(I,J) &= (1-\eta)S(I,J') + \eta S(I,J'+1) \\ \eta &= h - J' \\ J' &= \lfloor h \rfloor \\ h &= J + \frac{\Delta B_0(I,J)}{G_y p_y} \end{aligned} \tag{3.15}$$

Where p_y is the pixel width in the y-direction.

Although this technique offers accurate image reconstruction, there is degradation in the SNR. This is undesirable in the case of low-field imaging, where the SNR is already low.

3.6. Voxel Shift Correction

Both the CPR and MB methods attempt to correct the image distortion due to B_0 field inhomogeneities; their current formulation does not take into account other sources of field disturbances. Rather than correcting for each individual component of field inhomogeneity in the signal model, the voxel shift method directly translates the image voxels from their original location to their estimated location.

This method requires an accurate distortion map of the scanner. To that end, Doran et al. [26] designed a custom three-dimensional phantom with orthogonal grids of fluid-filled rods. Once the distortion map is known, the original image is resolved by shifting the voxels by the amount prescribed by the map. Figure 3.3 illustrates the degree of correction achieved using this method.

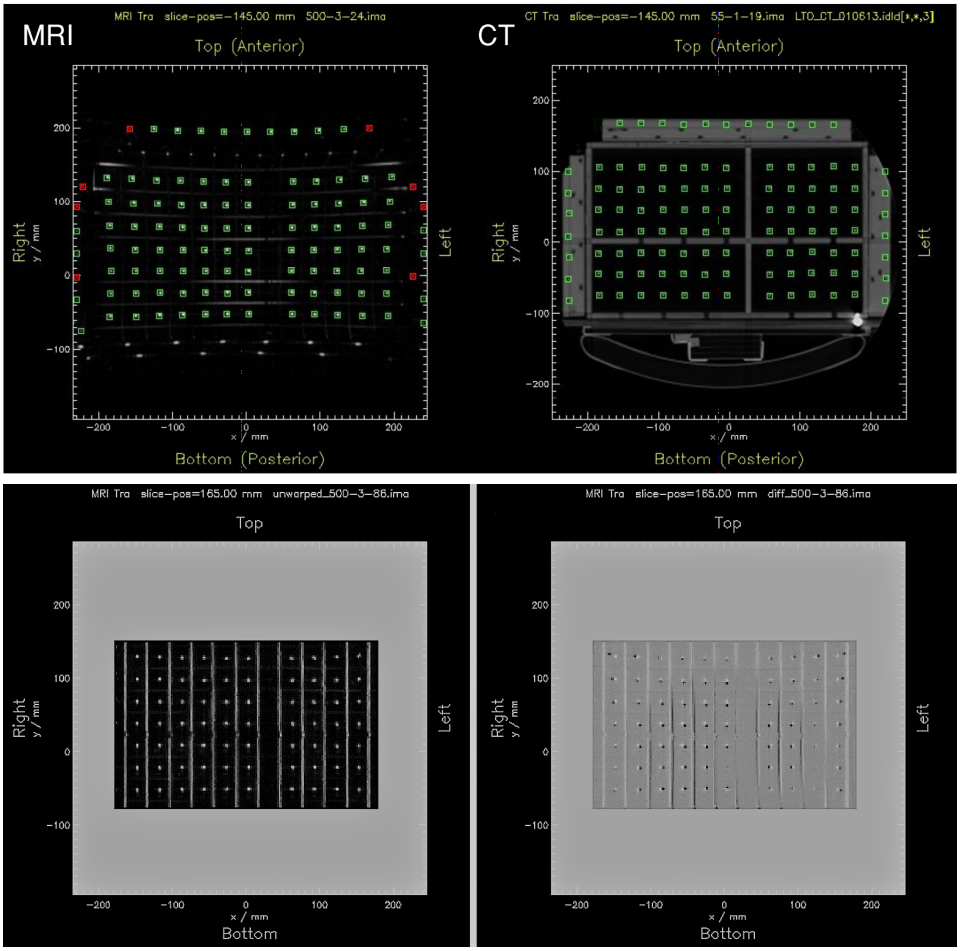

Figure 3.3: A tube phantom scanned with a MRI scanner (top left) and a computed tomography scanner (top right) for comparison. The corrected image (bottom left) and the difference between the reconstructed and original image (bottom right), using the voxel shift correction technique are shown. Images adapted from Doran et al. [26].

The method proposed by Doran et al. is able to efficiently correct for large distortions (> 25mm) [26]. There is, however, one drawback. When the field changes, due to either the use of a different scanner or the usage of a time-varying field, a new distortion map is required.

3.7. Optimal Correction Method

With the low-field scanner currently in development, the field map of the scanner might change due to updated components. Furthermore, thermal effects lead to a field map that gradually varies over time. This makes the voxel shift method, which relies on an accurate distortion map, impractical, as it would require a new distortion map for every scan.

Density correction, on the other hand, does not require a pre-computed distortion map. It does, however, decrease the SNR.

CPR and MB methods do not suffer from the aforementioned drawbacks. Both are able to recover the image, with MB reconstruction taking preference over CPR due to increased pixel intensity accuracy.

Thus, the work presented in the forthcoming chapters builds forth on techniques used in MB reconstruction.

4

Two-Dimensional Reconstruction

This chapter investigates how current reconstruction methods perform on simulated and in vivo single-slice MRI data. These methods are subsequently build into an iterative and a multi-slice framework. Both the processing time and degree of distortion correction will be taken into account. Initially, the simulated data will be noiseless, after which the effects of additive noise will be examined. Additionally, the difference in utilizing a measured and estimated field map will be investigated.

4.1. Simulation Model

In this chapter, all reconstruction simulations are applied on a distorted Shepp-Logan phantom. The distortion arises from B_0 field inhomogeneities. The Shepp-Logan phantom, illustrated in Figure 4.1, designed by L. A. Shepp and B. F. Logan [27], forms an ideal basis on which to perform MRI reconstruction techniques. The phantom models the contrast present in in vivo images, utilizes symmetrical shapes, and is free of noise. Hence, simulations with the parameters noted in Table 4.1, are done on this phantom unless mentioned otherwise.

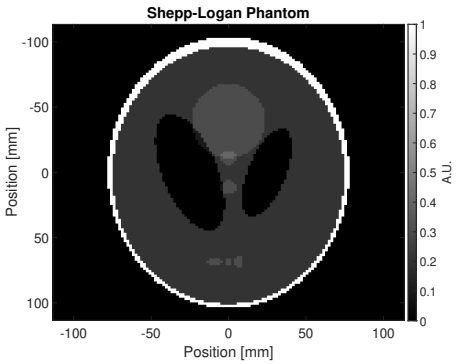

Figure 4.1: The undistorted Shepp-Logan phantom.

Table 4.1: Simulation parameters for three-dimensional data. In reconstruction of two-dimensional data, the third dimension was omitted.

Parameter	**Value**
Readout Bandwidth	20 kHz
Field of View	$225 \times 225 \times 225$ mm^3
Data Size	$128 \times 128 \times 30$
Resolution	$1.75 \times 1.75 \times 7.50$ mm^3

To simulate the incoming signal, the previously introduced signal model (Equation 4.1) is applied on the phantom ($\rho(\mathbf{r})$).

$$S_\tau(t) = \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau}\, d\mathbf{r} \tag{4.1}$$

The vectorized matrices are of the form mentioned in Table 4.2, where N is the number of samples per phase encoding step.

Table 4.2: Sizes of data matrices.

Data	Form
$\mathbf{r}$	$\mathbb{R}^{1\times N^2}$
ρ	$\mathbb{R}^{N^2\times 1}$
ΔB_0	$\mathbb{R}^{N^2\times 1}$
G_x	$\mathbb{R}^{N^2\times 1}$
G_y	$\mathbb{R}^{N^2\times 1}$

For simulations, the gradient fields are linear functions which model ideal MRI gradients. Figure 4.2 illustrates the gradient fields using the previously introduced parameters. In this work, the frequency and phase encoding are in the x- and y-direction respectively.

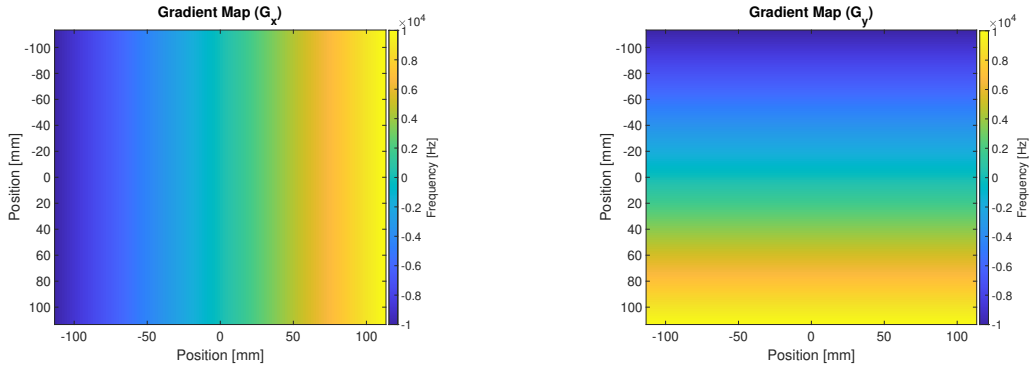

Figure 4.2: The simulated gradient fields in the x-direction (left) and y-direction (right).

To distort the phantom, the field map in Figure 4.3 is applied to the signal. This field map represents a realistic scenario for an off-center slice. The center field strength is subtracted from the field map, after which it is converted to hertz.

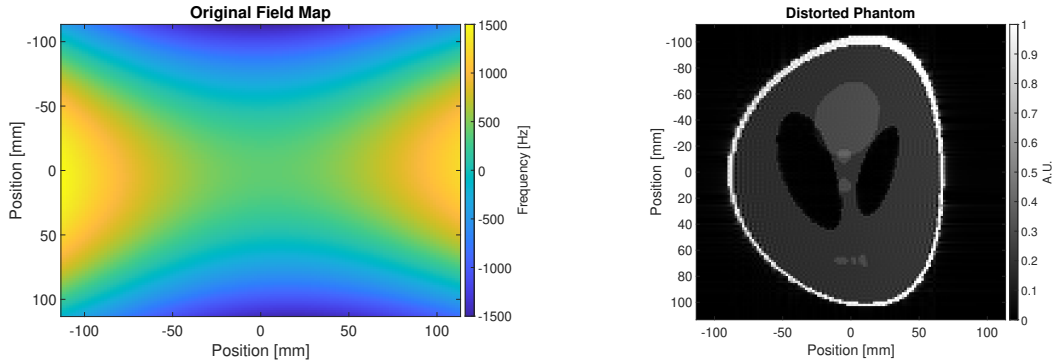

Figure 4.3: The field map used in the simulations (left) and the resulting distorted Shepp-Logan phantom (right).

The resulting distorted phantom (Figure 4.3) is then obtained by applying a 2D FFT on the simulated signal.

4.1.1. Sine-Bell Squared Filter

Currently, the data obtained with the low-field scanner is plagued with noise, which, due to the low signal intensity, results in a less than ideal SNR.

To reduced the impact of low SNR on the reconstruction, a pre-processing step is introduced, in which the signal is filtered to remove high-frequency components. One possible filter is the 2D sine-bell squared filter illustrated in Figure 4.4. Mathematically, the filter is:

$$H(t,\tau) = sin^2\left(\frac{\pi t}{T_f}\right) sin^2\left(\frac{\pi \tau}{T_p}\right) \tag{4.2}$$

with t and τ the time steps of the frequency and phase encoding directions, and T_f and T_p the times to complete a full set of frequency and phase encoding.

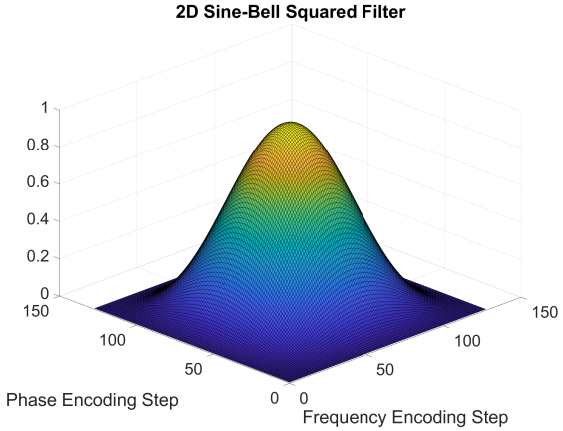

Figure 4.4: 2D Sine-Bell squared filter.

As part of the simulation process, the filter is also applied to the phantom data. This results in the image in Figure 4.5.

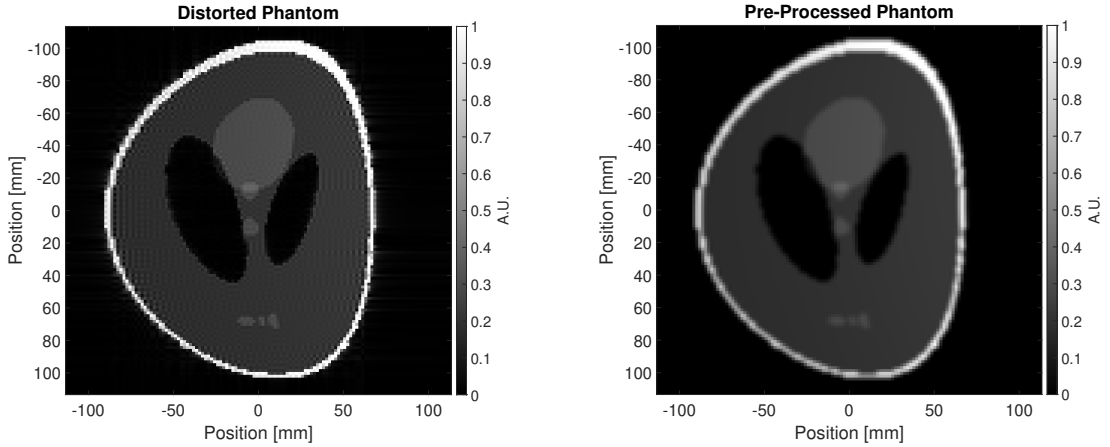

Figure 4.5: The simulated distorted phantom before (left) and after (right) pre-processing.

4.2. Reconstruction

From the previous section it is apparent that a 2D FFT is not capable of reproducing the undistorted phantom, hence the methods in Chapter 3 have been developed.

4.2.1. CPR

Recall that the CPR method attempts to reconstruct the image by multiplying the conjugate phase with the time domain signal. When discretized and written as a linear system, this becomes a Hermitian transpose.

$$\begin{aligned} \mathbf{S} &= E\boldsymbol{\rho} \\ \boldsymbol{\rho} &\approx E^H\mathbf{S} \end{aligned} \tag{4.3}$$

Reconstructing the image with CPR in the absence of noise and with the original field map results in a near perfect rendition of the original phantom. The reconstructed image is displayed in Figure 4.6. The residual two-norm error, compared to the original phantom, is 0.4734. In the absence of noise, CPR is able to recover a visually identical image compared to the original.

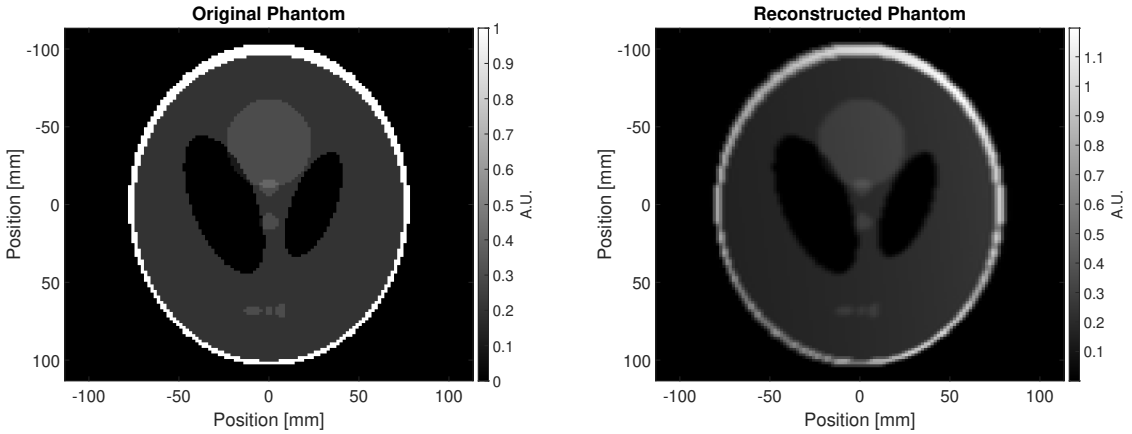

Figure 4.6: The original phantom (left) and CP reconstructed phantom (right).

4.2.2. Model-Based

As mentioned in the previous chapter, the goal of MB reconstruction is to solve the inverse problem $E\mathbf{x} = \mathbf{y}$, where $\mathbf{x}$ is the original image and $\mathbf{y}$ the acquired MRI signal.

A myriad of solvers use the conjugate gradient (CG) method, developed by M.R. Hestenes and E. Stiefel [28], as a basis. One such solver is the conjugate gradient least squares (CGLS) algorithm, introduced by P. Sonneveld, which offers improved stability over CG [17], [29]. The CGLS method is depicted in Algorithm 1.

In order for CG-based methods to converge, the system matrix, A, needs to be symmetric positive-definite. To ensure this condition, the normal equations are introduced, which take the form:

$$\begin{aligned} E^HE\mathbf{x} &= E^H\mathbf{y} \\ A\mathbf{x} &= \mathbf{b} \end{aligned} \tag{4.4}$$

Algorithm 1: Conjugate Gradient Least Squares (CGLS)

Input: $A \in \mathbb{C}^{M^2 x N^2}$, $\mathbf{b} \in \mathbb{C}^{M^2 x 1}$,$\mathbf{x}_0 \in \mathbb{C}^{N^2 x 1}$,$TOL \in \mathbb{R}_+$, $k_{max} \in \mathbb{N}_1$

$\mathbf{x}_0 = \mathbf{0}$
$\tau_0 = \|A\mathbf{x}_0 - \mathbf{b}\|_2^2$
$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$
$\mathbf{s}_0 = A\mathbf{r}_0$
$\mathbf{p}_0 = \mathbf{s}_0$
$k = 0$
while $\tau_k > TOL$ ***or*** $k < k_{max}$ **do**
  $\alpha_k = \frac{\langle \mathbf{s}_k, \mathbf{s}_k \rangle}{\langle A\mathbf{p}_k, A\mathbf{p}_k \rangle}$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$
  $\mathbf{s}_{k+1} = A^H \mathbf{r}_{k+1}$
  $\beta_k = \frac{\langle \mathbf{s}_{k+1}, \mathbf{s}_{k+1} \rangle}{\langle \mathbf{s}_k, \mathbf{s}_k \rangle}$
  $\mathbf{p}_{k+1} = \mathbf{s}_{k+1} + \beta_k \mathbf{p}_k$
  $\tau_{k+1} = \|A\mathbf{x}_{k+1} - \mathbf{b}\|_2^2$
  $k = k + 1$
end

The ensuing reconstructions utilize a tolerance, TOL, and maximum number of iterations, k_{max}, of $TOL = 10^{-2}$ and $k_{max} = 100$. A tolerance of 10^{-2} leads to adequate image recovery, while a lower or higher tolerance might lead to an increase in computation time or a decrease in reconstruction accuracy respectively.

Employing the CGLS algorithm in the reconstruction of the distorted, noiseless phantom, results in the recovered image shown in Figure 4.7. The resulting image is similar to the image recovered using the CPR method, and is visually identical to the original phantom. The residual error, compared to the original phantom, is 0.1641.

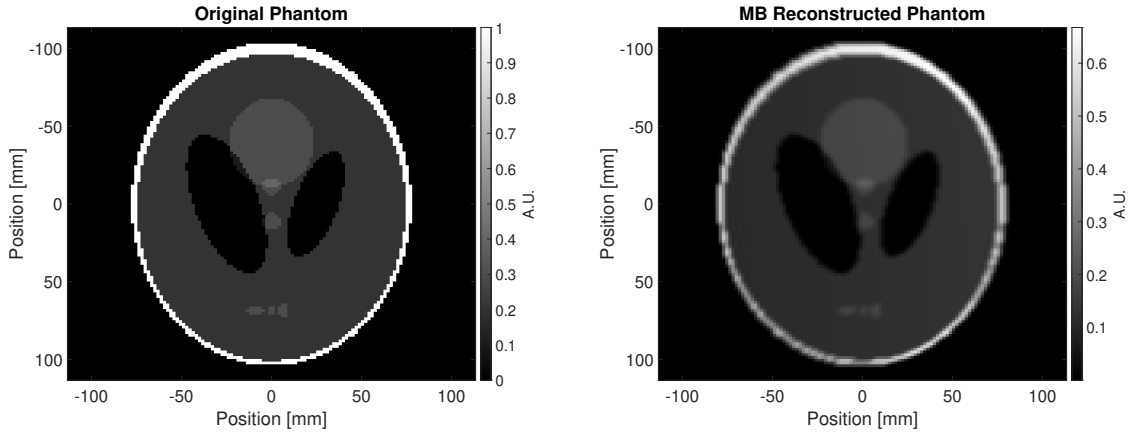

Figure 4.7: The original phantom (left) and MB reconstructed phantom (right).

4.3. Estimated Field Map

Thus far, the field map utilized in the reconstruction is the field map used to distort the phantom. In practice, the original field map is unknown. However, as expressed in section 3.1.1, an estimate of the field map can be made by exploiting the phase difference between two sets of images.

4.3.1. Spherical Harmonics

Computing the field map through the use of phase shift leads to an estimation of the field strength over the domain of the volume. MB algorithms, however, rely on a field map that extends over the complete FoV. Moreover, a method that estimates the field map outside the imaging volume is required.

One approach to this obstacle is the use of spherical harmonics. In this method, the initial field map (the estimate over the object domain), is fitted on a set of spherical harmonics, resulting in an estimate of the field map over the complete FoV.

The exact mathematics behind spherical harmonics fall outside the scope of this thesis. Thus, an in-house spherical harmonics Matlab code developed at the Leiden University Medical Center by Tom O'Reilly is used in subsequent reconstructions. For more information on spherical harmonics in the context of low-field MRI and image reconstruction, refer to the paper by Koolstra et al. [16].

4.4. Additive White Gaussian Noise

Real-world signals suffer from noisy measurements. Due to low signal power in low-field MRI systems, noise is significantly more prevalent than in high-field scanners.

The noise in the real and imaginary components of the signal of the low-field scanner can be modelled as zero-mean additive white Gaussian noise (AWGN). This can be inferred by considering the measured signal outside the image domain. In the in vivo data set used in this thesis, the first few slices are void of the imaged object, thus making them ideal for noise measurements. Appendix A explains this in more detail. The measured noisy signal, S, thus takes the form:

$$\begin{aligned} S &= X_R + N_R + j(X_I + N_I) \\ \{N_R, N_I\} &\sim \mathcal{N}(0, 0.0014) \end{aligned} \tag{4.5}$$

where X is the noiseless signal and N the AWGN.

With the addition of noise on the distorted phantom, the simulated reconstruction process represents a more accurate picture of reconstruction of low-field MRI data.

The resulting simulation pipeline is depicted in Figure 4.8. With this pipeline, the recovered image shown in Figure 4.9 is obtained.

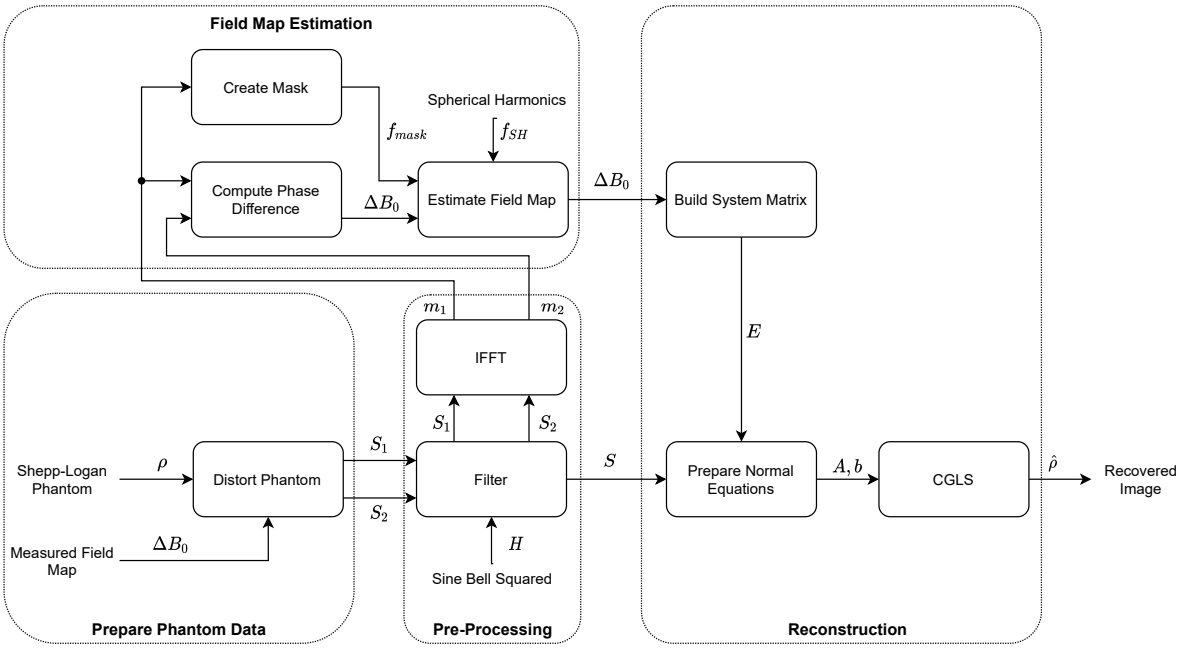

Figure 4.8: The simulation pipeline.

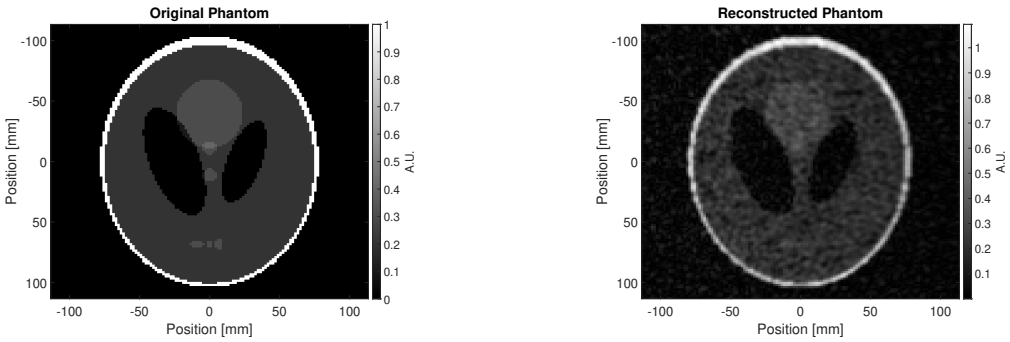

Figure 4.9: The original phantom (left) and the MB recovered image (right), with distorted noisy phantom as input.

4.5. Reconstruction of Low-Field MRI Data

To validate the reconstruction algorithm, two sets of data obtained using the low-field scanner are reconstructed. The first set comprises of 45 equally spaced tubes of sunflower oil, which, when recovered, will indicate the degree of distortion correction. A second set of data will illustrate the effect of the algorithm on an in vivo scan.

4.5.1. Tube Phantom

The tube phantom, depicted in Figure 4.10, presents distortion congruent with the simulated phantom when recovered using the FFT method. Reconstruction with the MB algorithm, on the other hand, returns the expected square array of tubes. This indicates that the MB method is able to accurately correct for field inhomogeneities in a single-slice data set.

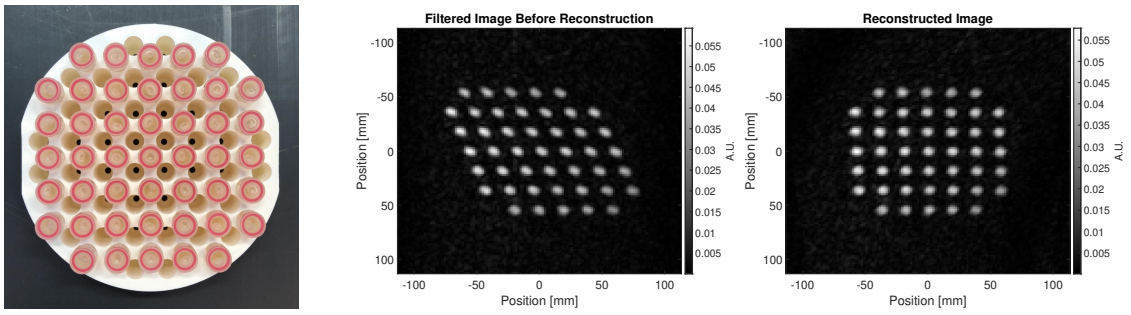

Figure 4.10: A tube phantom (left) reconstructed using the FFT (middle) and MB (right) methods.

Before reconstruction, the field map is estimated. The field map based on the phase shift in the two data sets of the tube phantom, together with the fit on a basis of spherical harmonics, are displayed in Figure 4.11.

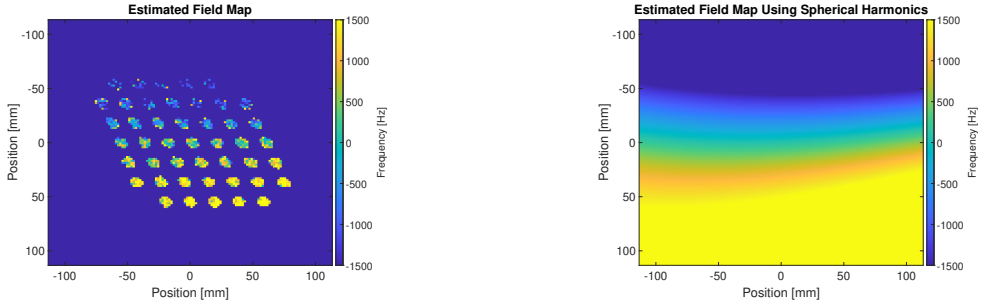

Figure 4.11: The estimated field map over the object domain (left) and imaging domain (right) using spherical harmonics.

4.5.2. In Vivo

Reconstruction will ultimately be applied on in vivo data. To that end, the reconstruction algorithm is tested on a slice of a brain scan. Figure 4.12 illustrates the distortion correction of the MB this method on the data in comparison to the classical FFT method. Similar to reconstruction of the tube data set and simulated phantom, MB recovery is able to correct the image distortion.

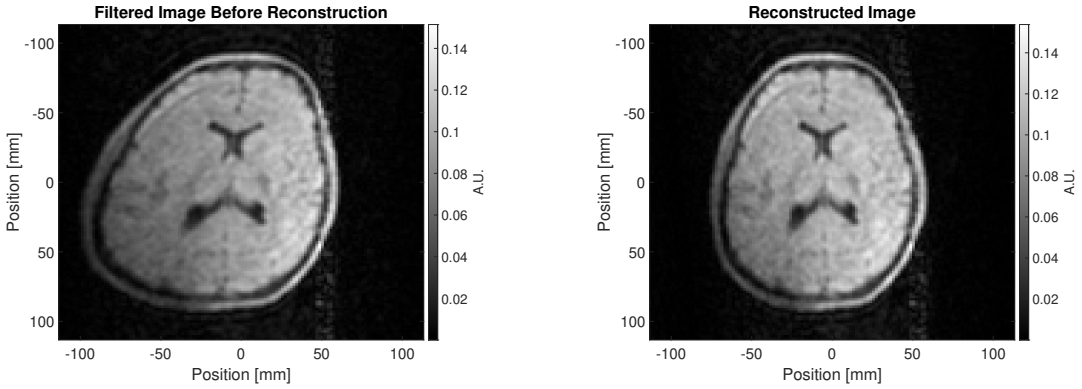

Figure 4.12: The in vivo image recovered using FFT (left) and MB reconstruction (right).

4.6. Iterative Reconstruction

Post reconstruction of the two sets of in vivo data (one with, and one without a readout time-shift), a new estimate of the field map can be realized. This new field map possesses enhanced accuracy compared to the previous estimate, according to Koolstra et al. [16]. Through passing this field map to the start of the pipeline and repeating reconstruction, an iterative MB method is achieved.

To verify this, the simulated phantom is reconstructed with this iterative framework. After every iteration, the residual error is calculated. The resulting residual error plot is displayed in Figure 4.13. From the figure it is apparent that a second iteration of reconstruction is beneficial to the results, while three or more iterations do not contribute to an improved image.

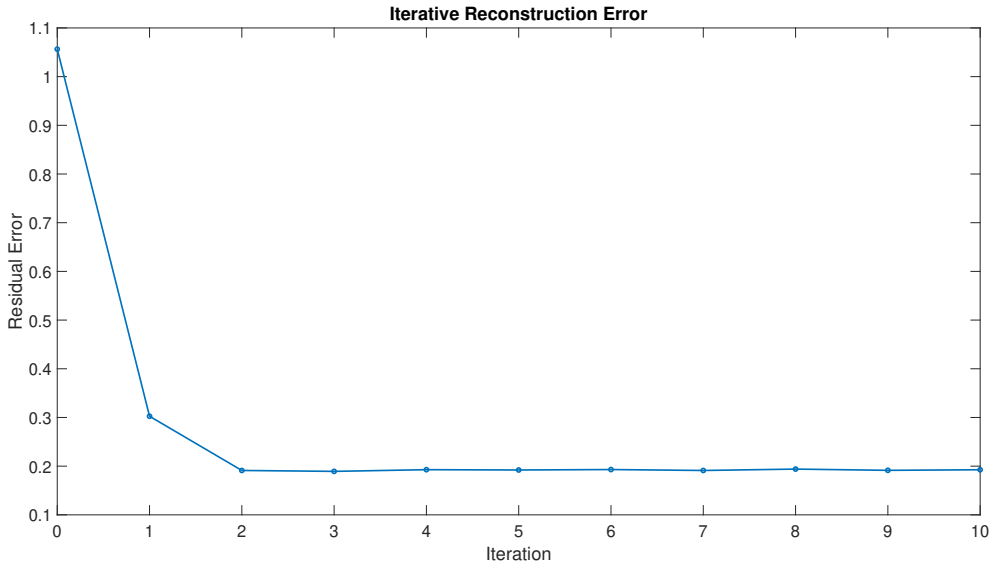

Figure 4.13: The residual error, calculated using the relative two-norm error between the reconstructed and the original image, after each iteration of reconstruction. The zeroth iteration indicates the error between the FFT reconstructed and original image.

4.7. Multi-Slice Reconstruction

Analogous to iterative reconstruction, the full three-dimensional data set can be recovered by sequentially passing the individual slices of data through the system. Figure 4.14 displays a subset of slices from the three-dimensional data, both pre- and post-reconstruction.

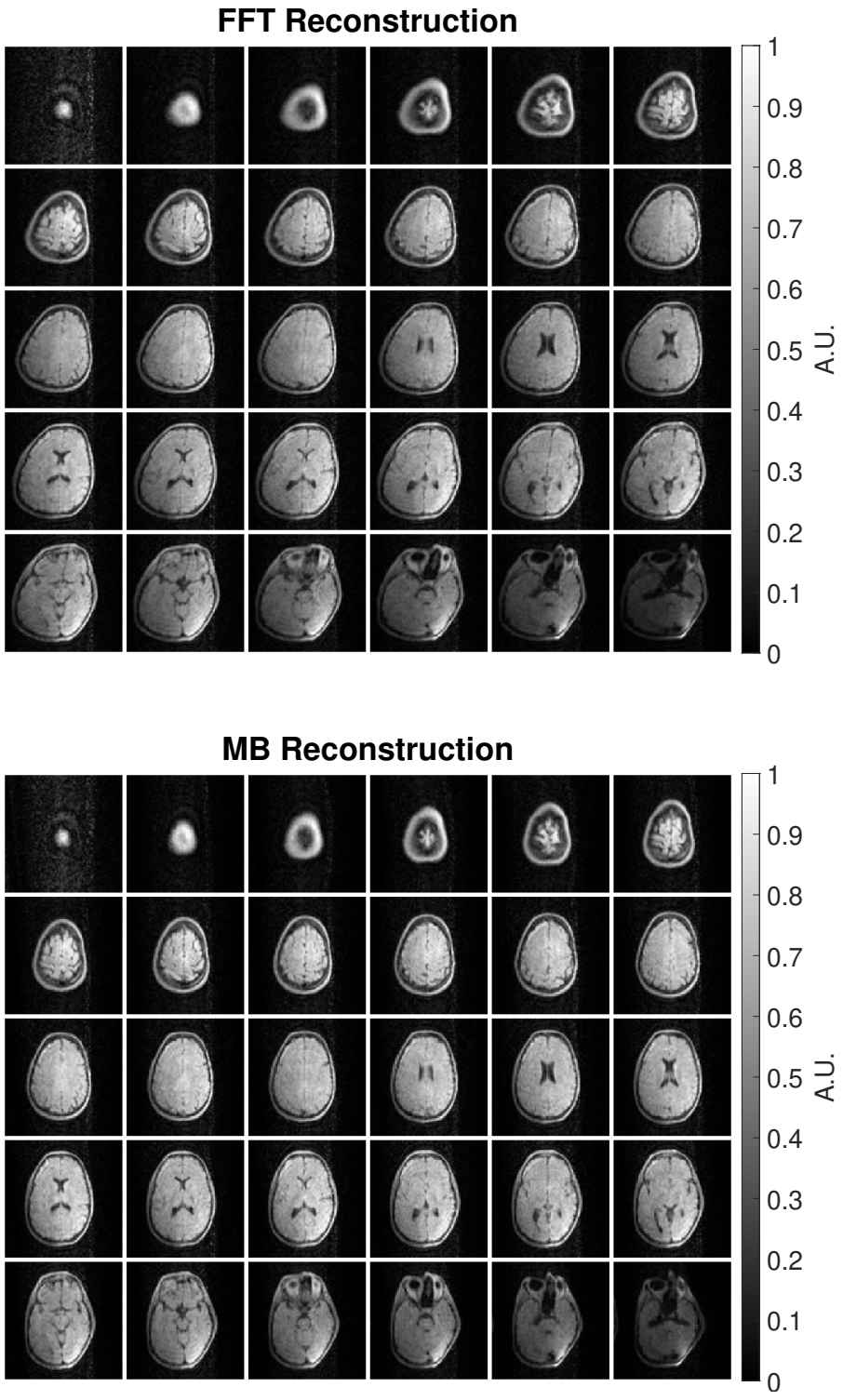

Figure 4.14: Two-dimensional FFT (top) and MB (bottom) multi-slice reconstruction of in vivo data.

To obtain the in vivo data set used in reconstruction, the original signal was first transformed to the spatial domain using a three-dimensional FFT. From this spatial data, leading and trailing slices absent of in vivo data were omitted. A subset of equally spaced slices was then selected. Each slice was transformed back to the time domain with a two-dimensional FFT, leading to a $128 \times 128 \times 30$ multi-slice data set.

From visual inspection, it is apparent that the reconstruction system does indeed correct for the image distortion. In the FFT reconstructed images, slice 25 and onward exhibit a visual artifact in the form of a bright spot, which MB reconstruction is unable to correct. This artifact is likely caused by non-linearities in the gradient fields, which are more pronounced near the edges of the imaging domain. These non-linearities furthermore cause the MB method to inadequately correct the image distortion in the last six slices.

Gradient correction might be able to relieve the problem, depending on whether or not the encoding remains unique. Furthermore, transforming the two-dimensional multi-slice reconstruction algorithm to a three-dimensional reconstruction system might improve the reconstruction.

5

Three-Dimensional Reconstruction

Two-dimensional reconstruction offers the ability to recover images on a slice-by-slice basis. While effective, the techniques that reconstruct the images are limited to data sets comprised of individual slices of data. In certain cases, MRI systems perform volumetric scans, with concomitant three-dimensional spatial encoding.

This chapter builds forth on the two-dimensional multi-slice reconstruction system. To that end, a second phase encoding step is introduced, accompanied by a third gradient vector. Subsequently, the effectiveness of three-dimensional reconstruction will be explored and compared to its two-dimensional counterpart.

5.1. Slice Encoding

Thus far, the simulated MRI data has been encoded using a frequency and a phase encoding step. With the addition of a third dimension, a second phase encoding step becomes necessary. This encodes the data along the slice direction.

The additional phase encoding step calls for a third gradient, G_z. Expanding the previously introduced gradients, G_x and G_y, over the three-dimensional FoV leads to gradient fields exhibited in Figure 5.1.

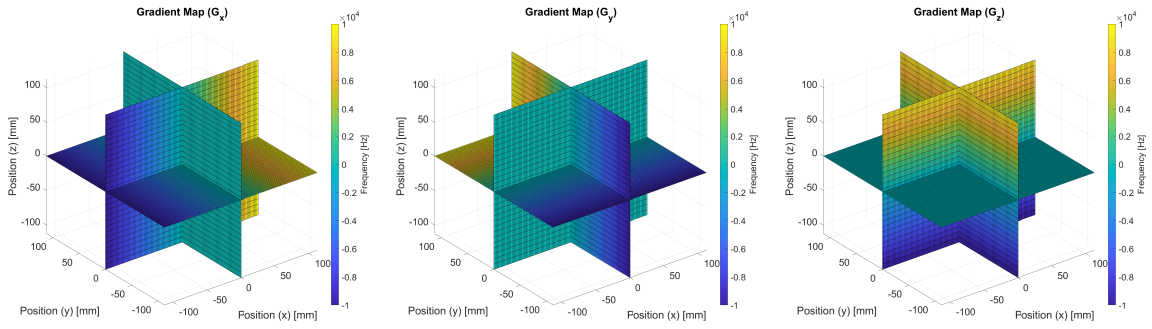

Figure 5.1: The gradient fields extended over the three-dimensional FoV.

With the extra phase encoding step, the three-dimensional signal model becomes:

$$S_{\tau_{pe},\tau_{se}}(t) = \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau_{pe}} e^{-j\gamma G_z(\mathbf{r})\tau_{se}}\, d\mathbf{r} \tag{5.1}$$

where τ_{se} represents the time step in the slice encoding direction.

Discretizing the signal model leads to a set of system matrices, where each system matrix describes one phase encoding step. Stacking these matrices leads to a two-dimensional matrix which contains the complete three-dimensional encoding information.

$$\begin{aligned}
S_{\tau_{pe},\tau_{se}}(t) &= \int_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B_0(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau_{pe}} e^{-j\gamma G_z(\mathbf{r})\tau_{se}}\, d\mathbf{r} \\
S_{\tau_{pe},\tau_{se}}(t) &= \lim_{\Delta\mathbf{r}\to 0} \sum_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{(-j\gamma\Delta B(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau_{pe}} e^{-j\gamma G_z(\mathbf{r})\tau_{se}}\, \Delta\mathbf{r} \\
S_{\tau_{pe},\tau_{se}}(t) &\propto \sum_{\mathbf{r}\in\mathbb{D}} \rho(\mathbf{r}) e^{-j\gamma\Delta B(\mathbf{r})t} e^{-j\gamma G_x(\mathbf{r})t} e^{-j\gamma G_y(\mathbf{r})\tau_{pe}} e^{-j\gamma G_z(\mathbf{r})\tau_{se}} \\
\mathbf{S}_{\tau_{pe},\tau_{se}} &= E_{\tau_{pe},\tau_{se}}\boldsymbol{\rho}
\end{aligned} \tag{5.2}$$

For one step in both the phase encoding directions, the system matrix has the following form:

$$E_{\tau_{pe},\tau_{se}} = \begin{bmatrix}
e^{-j\gamma((\Delta B(r_0)+G_x(r_0))t_0+G_y(r_0)\tau_{pe}+G_z(r_0)\tau_{se})} & \dots & e^{-j\gamma((\Delta B(r_{p-1})+G_x(r_{p-1}))t_0+G_y(r_{p-1})\tau_{pe}+G_z(r_{p-1})\tau_{se})} \\
e^{-j\gamma((\Delta B(r_0)+G_x(r_0))t_1+G_y(r_0)\tau_{pe}+G_z(r_0)\tau_{se})} & \dots & e^{-j\gamma((\Delta B(r_{p-1})+G_x(r_{p-1}))t_1+G_y(r_{p-1})\tau_{pe}+G_z(r_{p-1})\tau_{se})} \\
\vdots & \ddots & \vdots \\
e^{-j\gamma((\Delta B(r_0)+G_x(r_0))t_{N-1}+G_y(r_0)\tau_{pe}+G_z(r_0)\tau_{se})} & \dots & e^{-j\gamma((\Delta B(r_{p-1})+G_x(r_{p-1}))t_{N-1}+G_y(r_{p-1})\tau_{pe}+G_z(r_{p-1})\tau_{se})}
\end{bmatrix} \tag{5.3}$$

where p is the total number of encoding points ($M \cdot N \cdot L$).

With M phase encoding steps, N frequency encoding steps, and L slice encoding steps, the total encoding matrix takes the form: $E \in \mathbb{C}^{M\cdot N\cdot L\times M\cdot N\cdot L}$. Meanwhile, the encoded signal and phantom take the form $\mathbf{S} \in \mathbb{C}^{M\cdot N\cdot L\times 1}$ and $\boldsymbol{\rho} \in \mathbb{R}^{M\cdot N\cdot L\times 1}$ respectively.

5.2. Simulation Parameters

To analyze the performance of three-dimensional reconstruction, the system is initially tested on a simulated phantom. Similar to the two-dimensional situation, the Shepp-Logan phantom provides an ideal basis for simulation. The three-dimensional phantom used henceforth, based on the Shepp-Logan phantom, was created by Matthias Christian Schabel [30]. This phantom is illustrated in Figure 5.2. The accompanying simulation parameters are depicted in Table 4.1.

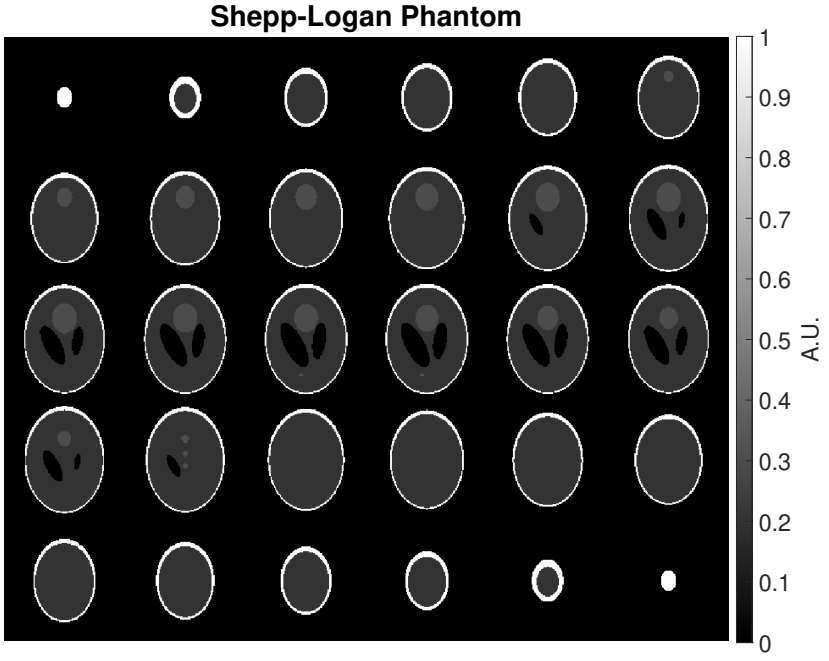

Figure 5.2: The three-dimensional Shepp-Logan phantom, developed by Matthias Schabel [30].

5.3. Memory Constraint

In order to build the complete system matrix for an imaging volume of $128 \times 128 \times 30$ data points, a matrix of size 491520×491520 is required. At double precision, the matrix will occupy roughly $491520 \cdot 491520 \cdot 64 = 1.54 \cdot 10^{13}$ bits $= 1.93$ TB. Working with a matrix of this size is infeasible for most desktop computers, and thus poses a memory constraint.

The forward problem, which encompasses encoding of the phantom using the channel information (the system matrix) to achieve a simulate MRI signal, can be solved by multiplying the system matrix with the phantom data. Alternatively, the simulated signal can be obtained by sequentially multiplying the individual rows of the system matrix with the phantom data. This nullifies the requirement of forming the complete system matrix, and thus eliminates the memory constraint.

Antipodal to the forward problem is the inverse problem, where the scanned image is recovered using estimated channel information and the time-domain signal. Analogous to the two-dimensional situation, the inverse problem can be solved using methods such as CPR and MB reconstruction. While CPR is able to recover the image employing individual rows of the system matrix, MB reconstruction relies on the complete system matrix.

To ensure that MB reconstruction remains feasible, the time-domain signal is downsampled to a workable size. On a computer with 16GB of RAM, the largest possible system matrix is approximately 44700×44700. Thus, unless otherwise mentioned, the downsampled dimensions of the time-domain signal are $64 \times 64 \times 6$, which satisfies the constraint.

With this memory constraint, the pipeline for three-dimensional simulation with MB reconstruction changes slightly with respect to the two-dimensional pipeline with the addition of a downsampling step.

5.4. Filter

When processing in vivo MRI data, a filter is employed to combat the noise. For completeness, this step is also included in the simulated reconstruction. The same filter is chosen as in the two-dimensional scenario, and expanded to the three-dimensional FoV (Equation 5.4). This filter is illustrated in Figure 5.3.

$$H(t, \tau_{pe}, \tau_{se}) = sin^2\left(\frac{\pi t}{T_{fe}}\right) sin^2\left(\frac{\pi \tau_{pe}}{T_{pe}}\right) sin^2\left(\frac{\pi \tau_{se}}{T_{se}}\right) \tag{5.4}$$

with t, τ_{pe}, and τ_{se} the time steps of the frequency, phase, and slice encoding directions, and T_{fe}, T_{pe}, and T_{se} the times to complete a full set of frequency, phase, and slice encoding.

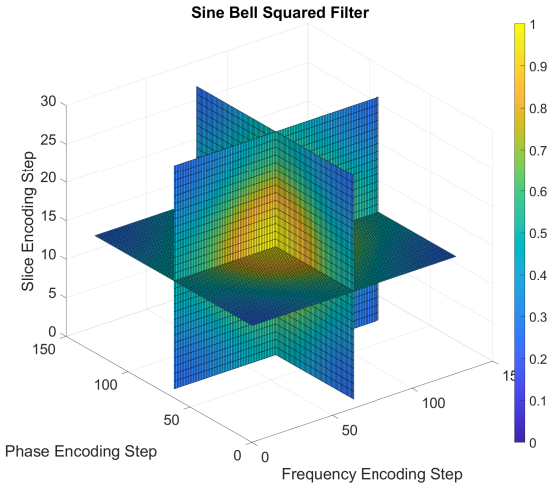

Figure 5.3: Three-dimensional sine bell squared filter.

In addition to the noise reduction, this filter ensures that the edges of the k-space data are brought to zero. This prevents discontinuity in the data when performing Fourier transforms.

5.5. Reconstruction

Following the steps from the pipeline in Figure 4.8 with the addition of a downsampling step, results in the intermediate images displayed in Figure 5.4, which depicts the simulated distorted phantom and the downsampled image, and Figure 5.5, which displays the FFT reconstructed image and the recovered image, using MB reconstruction.

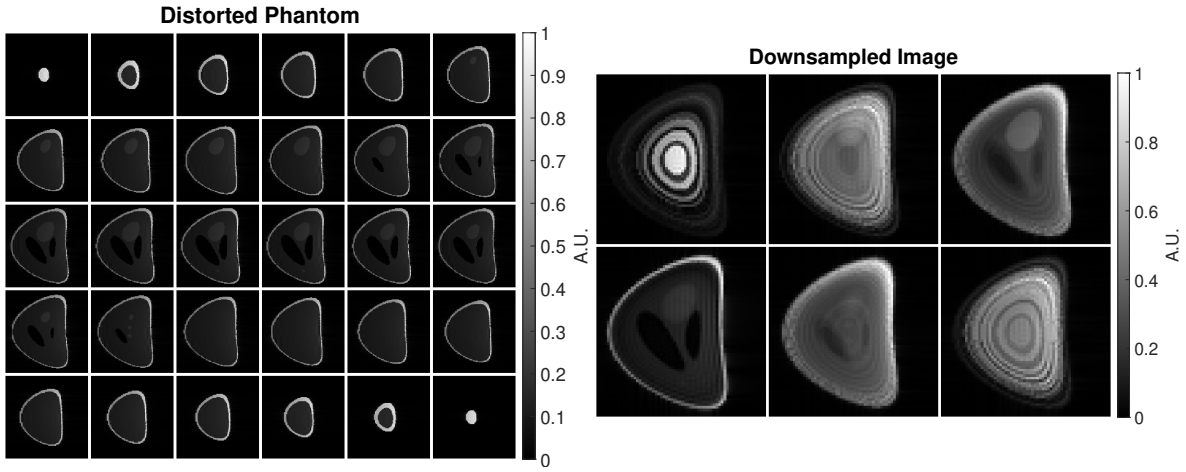

Figure 5.4: The phantom after simulated distortion (left) and after downsampling (right).

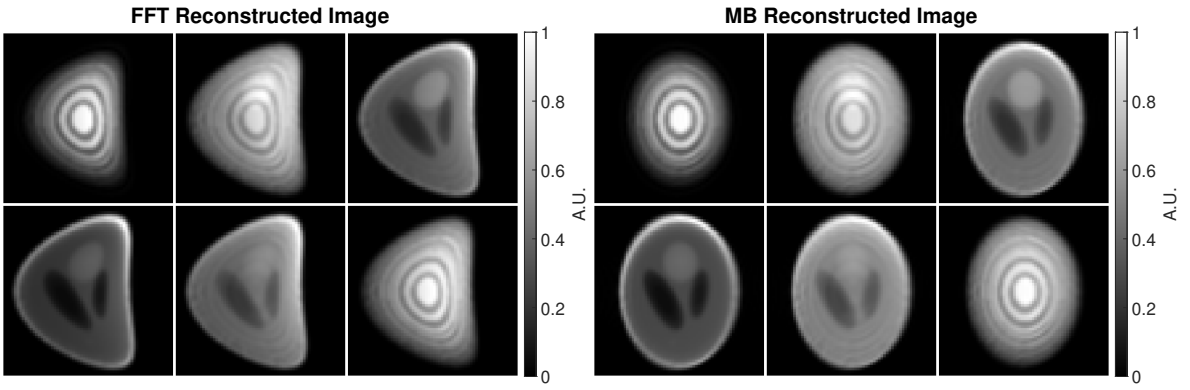

Figure 5.5: The recovered phantom using FFT (left) and one iteration of MB reconstruction (right).

While MB reconstruction is able to correct the image distortion, the ringing artifacts induced by downsampling of the k-space data remain unchanged.

5.6. Additive White Gaussian Noise

For increased simulation accuracy, zero-mean Gaussian noise is introduced in the signal according to the model presented in section 4.4 of the previous chapter. Reconstruction with the noisy signal leads to the recovered volumetric image depicted in Figure 5.6.

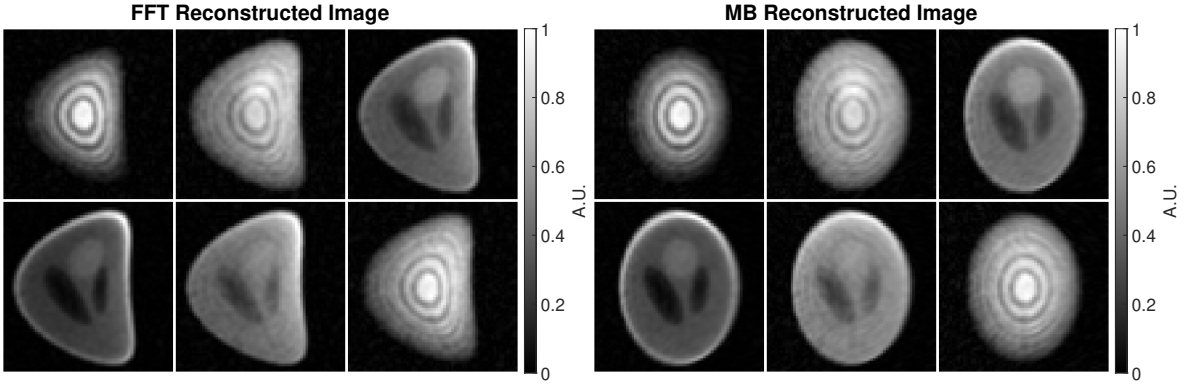

Figure 5.6: The recovered noisy phantom using FFT (left) and one iteration of MB reconstruction (right).

With the exception of noise in the recovered image, MB reconstruction leads to a visually identical image compared to the noiseless scenario.

5.7. Alternative Reconstruction Method

Due to significant downsampling of the original image, ringing artifacts appear. Neither the filter nor the reconstruction are able to alleviate this. To that end, a new reconstruction method that does not employ downsampling, nor exceed the memory constraint, needs to be designed.

By dividing the complete system matrix into subsets, the maximum memory required per set can be controlled. There are several methods to approach this subdivision, two of which are dismembering the system into sets of rows or into sets of columns. The former leads to a set of underdetermined systems, while the latter leads to a set of overdetermined systems. An underdetermined system has either an infinite amount of solutions or no solution, while an overdetermined system might display inconsistencies. Nevertheless, if a solution exists, CGLS is able to determine the optimal solution by minimizing the least squares residual.

In the case of column subdivision, the system takes the form shown in Equation 5.5, where $E_{i,j}$ is the i,j-th element of the total system matrix and P the total number of time samples. The column width is then determined based on the maximum amount of available virtual memory.

$$E = \left[\begin{bmatrix} E_{0,0} & E_{0,1} & \cdots \\ E_{1,0} & E_{1,1} & \cdots \\ E_{2,0} & E_{2,1} & \cdots \\ \vdots & \vdots & \vdots \\ E_{P-1,0} & E_{P-1,1} & \cdots \end{bmatrix} \cdots \begin{bmatrix} \cdots & E_{0,j} & \cdots \\ \cdots & E_{1,j} & \cdots \\ \cdots & E_{2,j} & \cdots \\ \vdots & \vdots & \vdots \\ \cdots & E_{P-1,j} & \cdots \end{bmatrix} \cdots \begin{bmatrix} \cdots & E_{0,P-2} & E_{0,P-1} \\ \cdots & E_{1,P-2} & E_{1,P-1} \\ \cdots & E_{2,P-2} & E_{2,P-1} \\ \vdots & \vdots & \vdots \\ \cdots & E_{P-1,P-2} & E_{P-1,P-1} \end{bmatrix}\right] \tag{5.5}$$

The new three-dimensional reconstruction method now consists of sequentially generating subsets, for which the normal equations are subsequently calculated, after which CGLS finds the least squares solution for each subset. The recovered image is then, in the case of sets of columns, the concatenation of the subset solutions.

Figure 5.7 displays the noisy phantom after reconstruction with the new method.

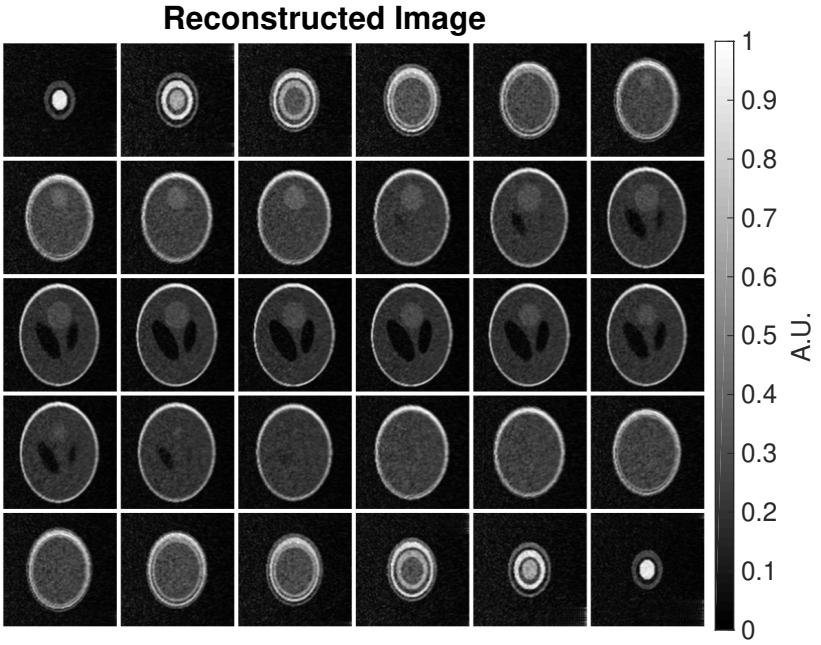

Figure 5.7: A $128 \times 128 \times 30$ noisy phantom recovered with the updated reconstruction method.

Visually, the reconstructed image closely resembles the original image. The ringing artifacts present in the recovered $64 \times 64 \times 6$ data set are now limited to the outer slices.

To investigate the extend of subdivision of the system matrix on the complete solution, the relative two-norm error for two reconstructed data sets has been calculated. In both sets, a $64 \times 64 \times 6$ noisy phantom is used as input. The first set is reconstructed without subdivision of the system matrix, while the system matrix in the second set is divided into four subsets. Their respective errors are 1.1464 and 1.1512, which, together with visual inspection (see Figure 5.8), indicates that the new method is sufficient at recovering the original images.

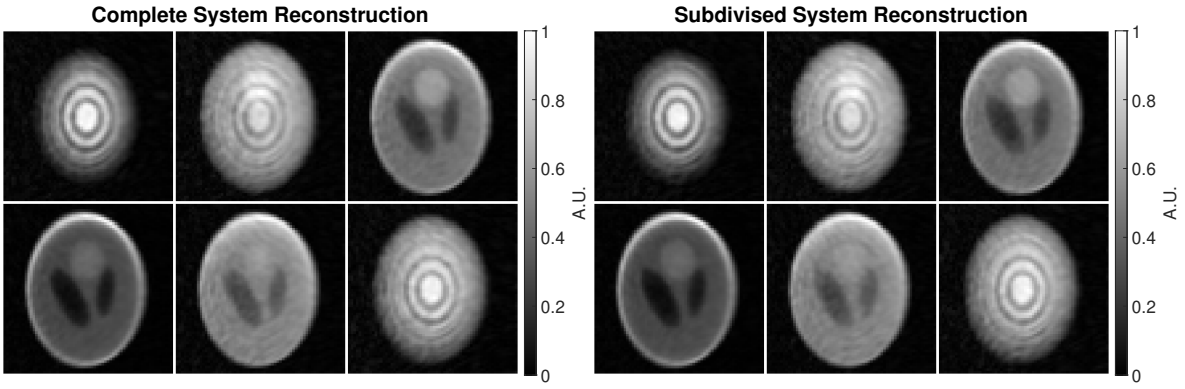

Figure 5.8: A $64 \times 64 \times 6$ phantom recovered using MB reconstruction with the full system matrix (left), and the subdivided system matrix (right).

5.8. In Vivo Reconstruction

With the revised reconstruction system in hand, $128 \times 128 \times 30$ in vivo data can now be recovered. Figure 5.9 displays the reconstructed image in conjunction with FFT recovery.

Reconstruction of the in vivo data leads to a volumetric image in which the initial distortion is corrected. Comparable to two-dimensional multi-slice reconstruction, the trailing slices are not accurately corrected due to gradient non-linearities.

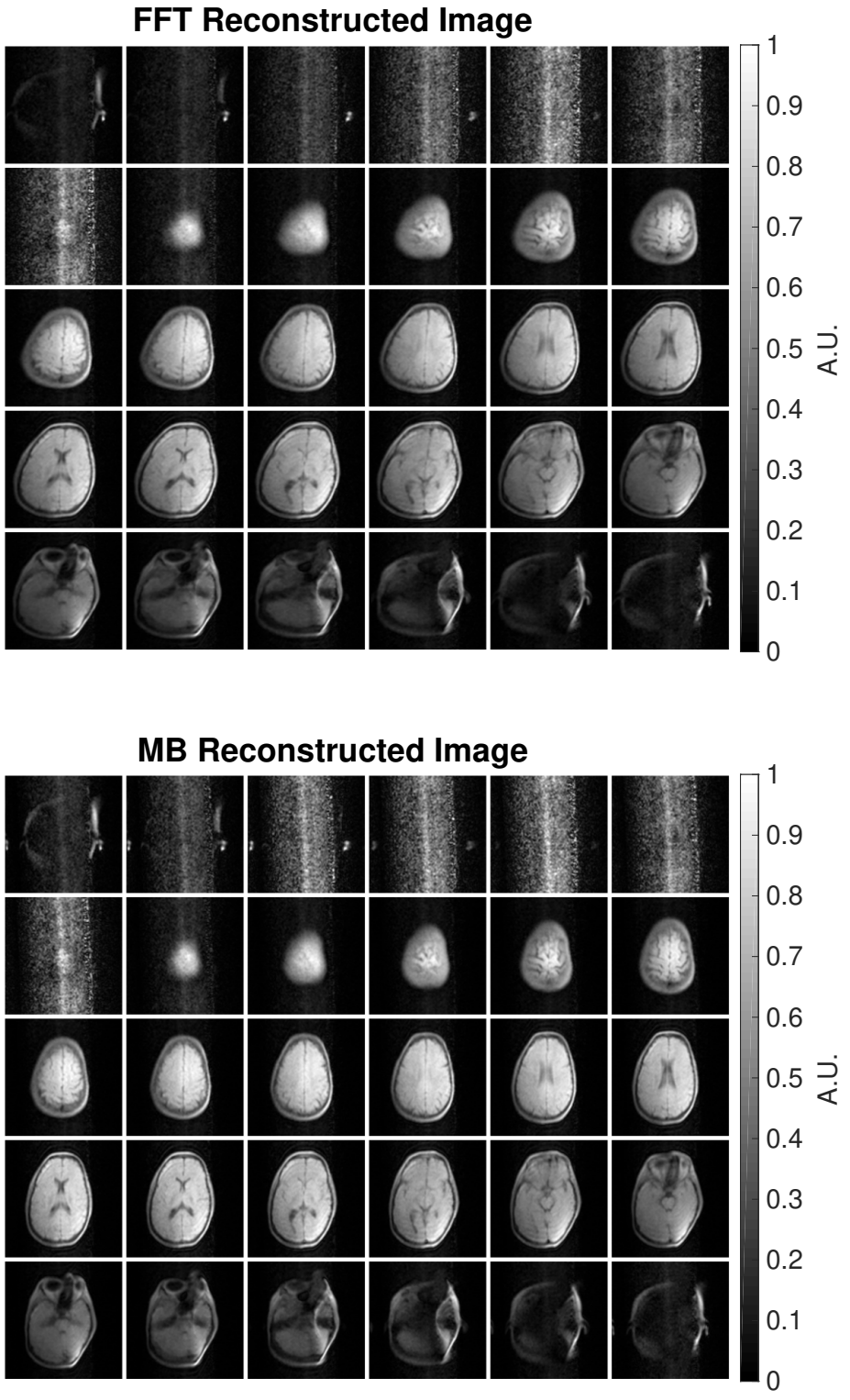

Figure 5.9: Recovered in vivo images from a three-dimensional data set utilizing FFT reconstruction (top) and the proposed algorithm (bottom).

6

Regularization

MB reconstruction relies on solving an inverse problem, which, in the presence of noise, is often ill-posed. Slight variations in the input signal from, for example, a Gaussian noise source, can lead to an unstable solution procedure.

To counteract instability, a regularizer is introduced in the inverse problem. This chapter focuses on what form this regularizer takes and how it affects the results. Furthermore, a method for finding the optimal regularization parameters is proposed.

6.1. Reconstruction Without Regularization

In the present two-dimensional reconstruction framework, CGLS is able to recover the image when the tolerance is set to 10^{-2}. When the tolerance is set to 10^{-4} for increased accuracy, the recovered image displays reconstruction artifacts, as depicted in Figure 6.1. This is caused by noise, which causes CGLS to converge to a non-optimal solution when the tolerance is too strict.

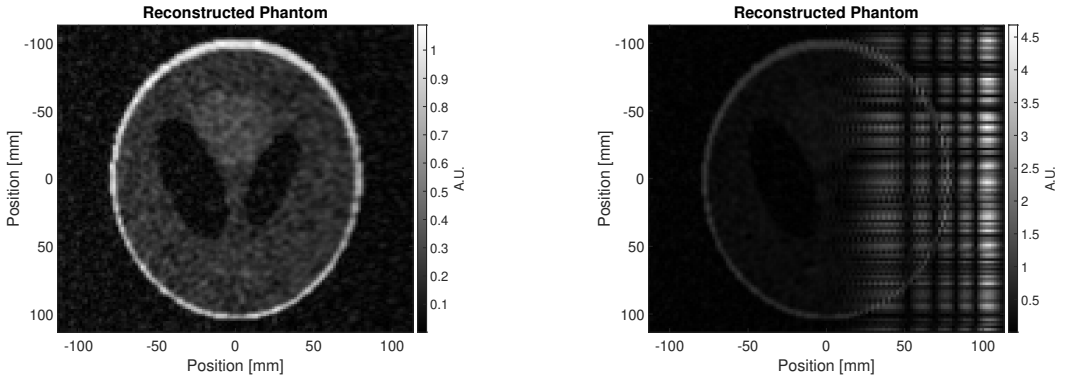

Figure 6.1: Reconstructed noisy phantom with CGLS tolerance of 10^{-2} (left), and 10^{-4} (right).

Similarly, when the filter is omitted from the reconstruction process, CGLS fails to accurately recover the original image. Hence, a regularizer is introduced.

6.2. Tikhonov Regularization

The current inverse problem constitutes solving the least squares minimization:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|A\boldsymbol{x} - \mathbf{b}\|_2^2 \tag{6.1}$$

To reinforce stability in the output solution, a regularizer can be employed. One possible regularization technique is Tikhonov regularization. Tikhonov regularization introduces a diagonal matrix with

constant value to the system, which decreases the sensitivity of the output due to the input. The general minimization problem then becomes:

$$\begin{aligned} \hat{\boldsymbol{x}} &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \|A\boldsymbol{x} - \mathbf{b}\|_2^2 + \|\Gamma\boldsymbol{x}\|_2^2 \\ \Gamma &= \alpha I \\ \alpha &\in \mathbb{R}_{\geq 0} \end{aligned} \tag{6.2}$$

where I is the identity matrix and α the regularization constant.

Determining the optimum value of the regularization parameter, α, to use is integral to the solver. A higher-than-optimal value for α might annul the results, while a value close to or equal to zero reverts the system to the unregularized scenario.

The regularization parameter can be found empirically by minimizing the 2-norm error between the recovered image and the original image. Alternatively, when the original image is not available, methods such as the L-curve can help find the optimal value for the regularization parameter [31]. The L-curve compares the residual norm, $\|A\boldsymbol{x} - \mathbf{b}\|_2$, with the solution norm, $\|\boldsymbol{x}\|_2$, where the solution is found using Tikhonov regularization. The optimum value of α is then located directly after the corner of the L-curve. Here, both the residual error and solution norm are at a minimum. Figure 6.2 illustrates the L-curve for the two-dimensional phantom data, recovered using MB reconstruction with Tikhonov regularization.

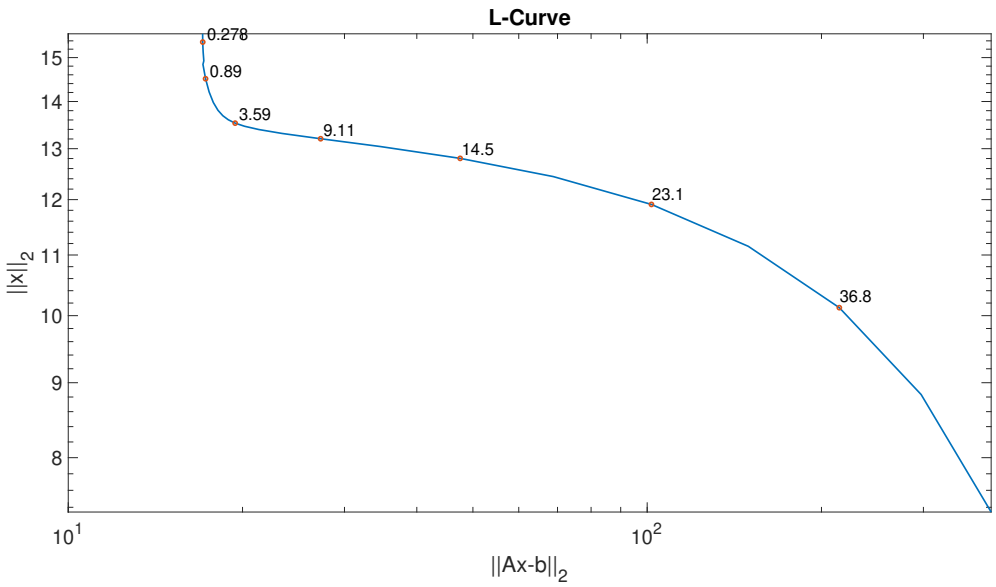

Figure 6.2: The L-curve for phantom data with identical SNR to in vivo data. Several values of α have been labeled. The optimum regularization parameter is located directly after the bend of the L-curve, at $\alpha = 3.59$. The CGLS tolerance is set to 10^{-4}.

From Figure 6.2, the optimal regularization parameter for this scenario is determined to be $\alpha = 3.59$. To verify this, the relative two-norm error between the reconstructed and original phantom has been plotted in Figure 6.3. Indeed, the aforementioned optimal regularization parameter falls within the acceptable range.

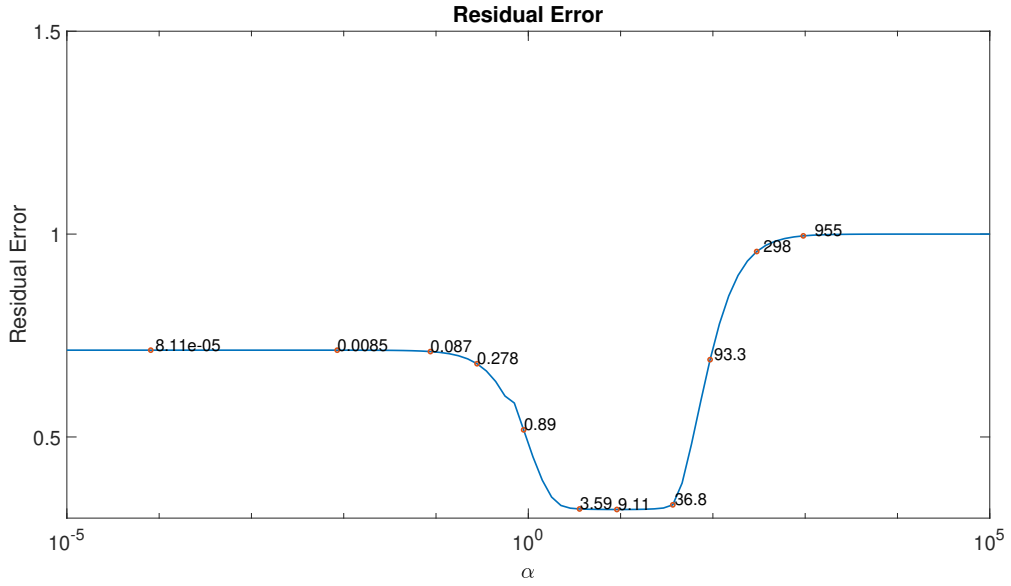

Figure 6.3: The relative two-norm error between the reconstructed and original phantom. Several values of the regularization parameter have been labeled.

Employing this optimal value, the reconstructed two-dimensional phantom is displayed in Figure 6.4. This leads to a visually identical image compared to unregularized reconstruction with tolerance of 10^{-2}.

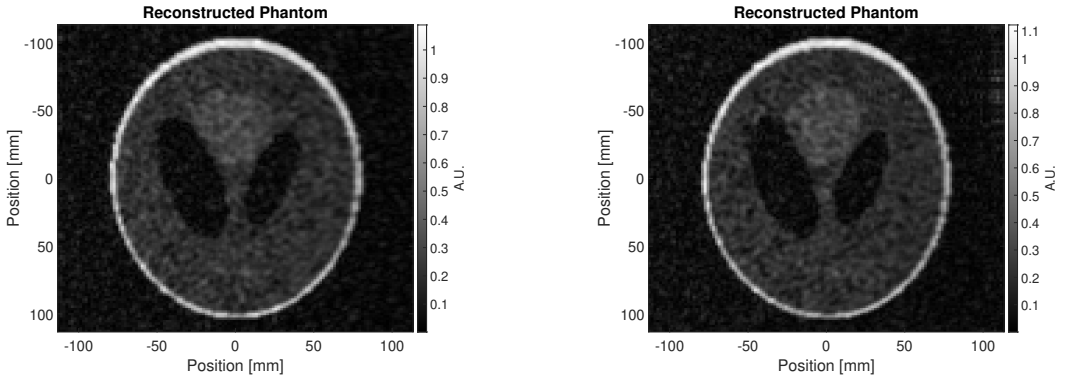

Figure 6.4: Reconstructed noisy phantom with CGLS tolerance of 10^{-2} without regularization (left), and 10^{-4} with optimal Tikhonov regularization (right).

6.3. Optimal Regularization Parameter

In the former section, a regularization parameter was found that is optimal for the reconstruction of a noisy two-dimensional Shepp-Logan phantom. This optimal parameter might not hold when performing reconstruction on different phantoms or in vivo data. Finding the optimal parameter for each individual scenario via, for example, the L-curve method, is a time-consuming task. An alternative time-efficient method that automates this procedure is thus required.

As previously mentioned, regularization is necessary in part due to the noise present in the images. This leads to the assumption that the optimal regularization parameter is correlated to the relative noise power present in the image. If the regularization parameter is indeed inversely proportional to the SNR, a function to determine the optimal parameter can be derived. To that end, the L-curves for varying SNRs have been plotted in Figure 6.5.

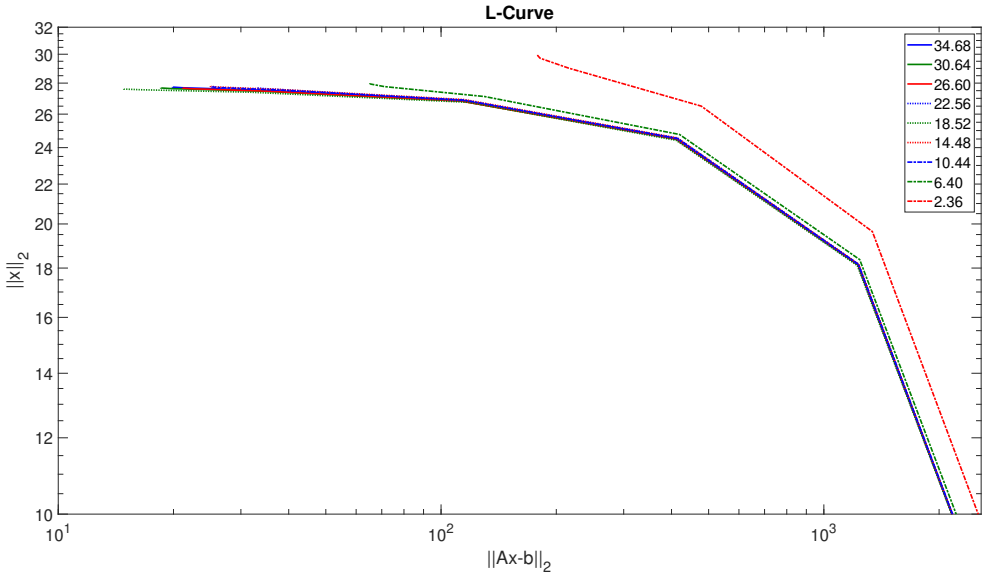

Figure 6.5: The L-curves for several different values of SNR. CGLS tolerance is set to 10^{-2}.

The absence of a bend in the L-curves indicates that regularization has minimal impact on the reconstruction when CGLS tolerance is set to 10^{-2}, regardless of SNR. This can be verified by considering the residual error, illustrated in Figure 6.6.

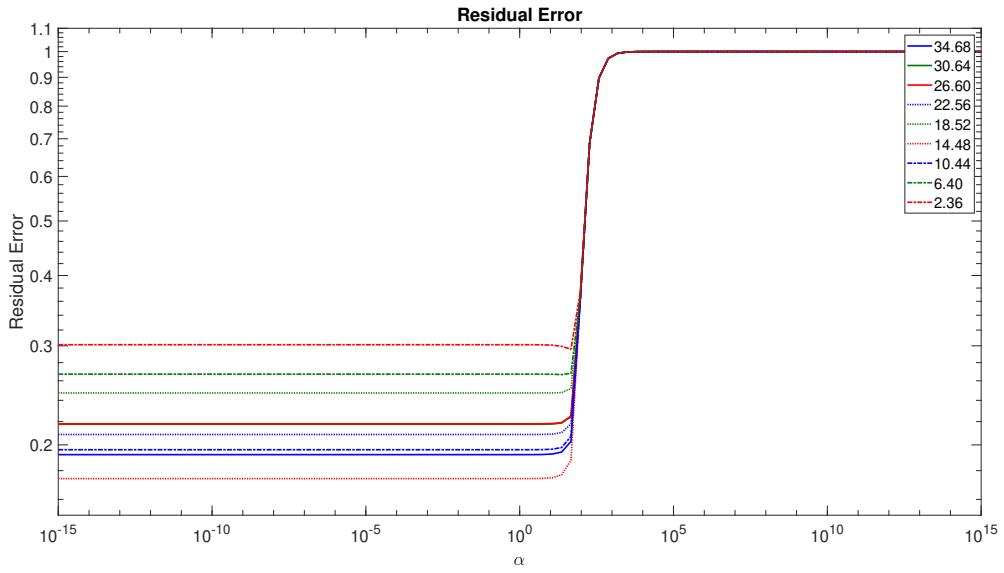

Figure 6.6: The residual error for varying SNR.

Figure 6.7 confirms this result. In this figure, a single-slice of in vivo data has been reconstructed with increasing regularization. The regularization has no beneficial effect on the result, and from iteration 14 and onward suppresses the output towards zero.

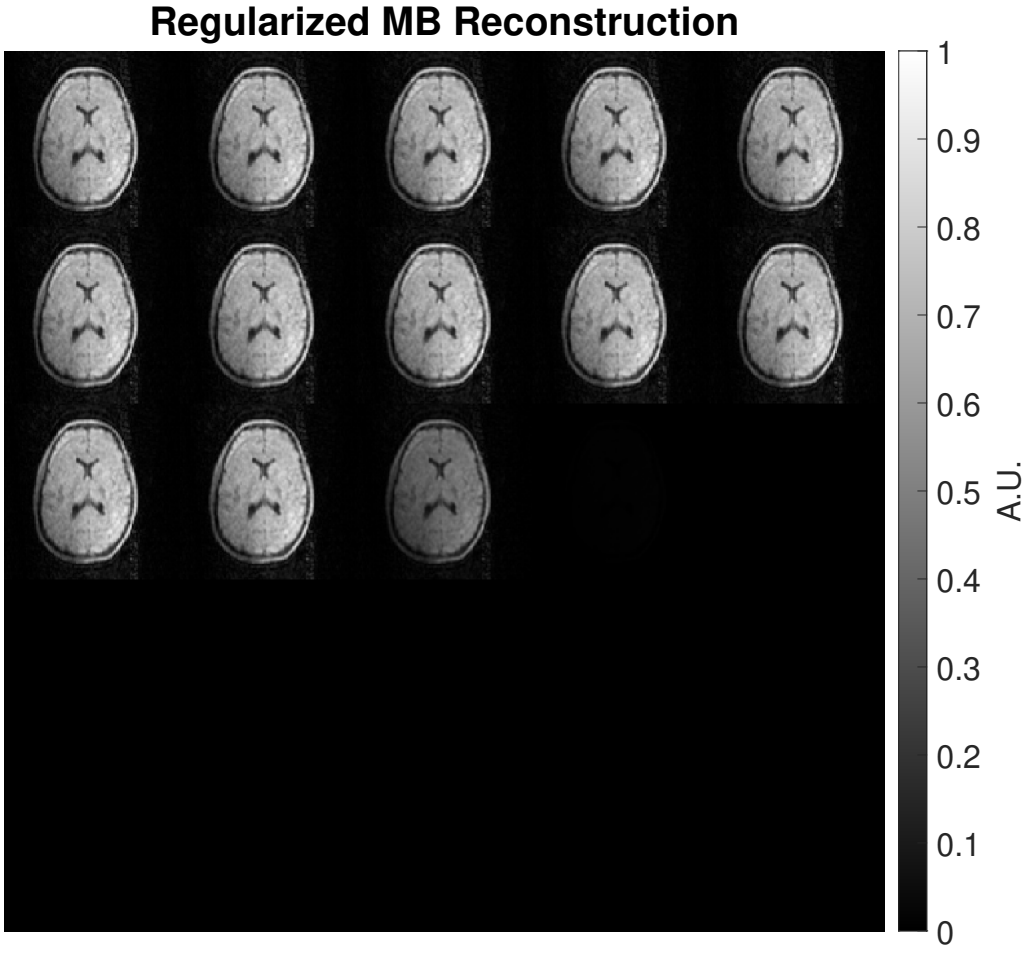

Figure 6.7: Single-slice of in vivo data reconstructed with increasing regularization. The regularization parameter ranges logarithmically from 10^{-10} to 10^{14}.

7

Performance

To assess the performance of the reconstruction algorithms, this chapter delves into the analytical results. Two key factors to consider are the residual error and the processing time. Furthermore, the effect of Tikhonov Regularization on the error will be investigated.

7.1. Residual Error

Thus far, the primary method for determining the effectiveness of the reconstruction algorithms was via visual inspection. An empirical approach to analyzing the recovered images is essential in ascertaining the performance.

In the case of simulated data, the original, undistorted image is available. This allows for error calculations based on the residual error between the recovered and original image. One method to determine the residual error is the relative two-norm error, which is determined as followed:

$$\epsilon = \frac{\|\hat{\boldsymbol{\rho}} - \boldsymbol{\rho}\|_2}{\|\boldsymbol{\rho}\|_2} \tag{7.1}$$

Here, ϵ is the error, $\boldsymbol{\rho}$ the original image, and $\hat{\boldsymbol{\rho}}$ the recovered image.

7.2. Two-Dimensional Reconstruction

Employing the aforementioned error function, the performance of two-dimensional reconstruction on a 128×128 phantom can be analyzed. The error and processing times for the scenarios examined in Chapter 4 are tabulated in Table 7.1.

Table 7.1: Two-norm reconstruction error and processing time for four scenarios of two-dimensional reconstruction after one iteration, averaged over three repetitions. The processing time includes the time required for field map estimation, system matrix construction, and image recovery, and excludes the time required for initialization and pre-processing. The phantom is of size 128×128.

Case	Field Map	Noise	Regularization	Two-Norm Error	Processing Time [s]
1	Original	None	No	0.21	17.74
2	Estimated	None	No	0.43	16.56
3	Estimated	AWGN	No	0.43	17.18
4	Estimated	AWGN	Yes	0.43	17.16

In the two-dimensional reconstruction, the type of field map used has the most significant impact on the results. Utilizing the estimated field map, as opposed to the measured field map, leads to roughly twice the two-norm error compared to the noiseless scenario. A second iteration of field map estimation leads to a comparable error, though at the cost of additional processing time.

7.3. Two-Dimensional Multi-Slice Reconstruction

Likewise, the empirical results for two-dimensional multi-slice reconstruction on a $128 \times 128 \times 30$ phantom are tabulated in Table 7.2.

Table 7.2: Two-norm reconstruction error and processing time of four scenarios of two-dimensional multi-slice reconstruction after one iteration, averaged over three repetitions. The processing time includes the time required for field map estimation, system matrix construction, and image recovery, and excludes the time required for initialization and pre-processing. The phantom has a size of $128 \times 128 \times 30$.

Case	Field Map	Noise	Regularization	Two-Norm Error	Processing Time [s]	Processing Time per Slice [s]
1	Original	None	No	1.05	256.26	8.54
2	Estimated	None	No	1.04	253.87	8.46
3	Estimated	AWGN	No	1.24	277.22	9.24
4	Estimated	AWGN	Yes	1.24	279.02	9.30

Dissimilar to single-slice reconstruction, noise has the largest impact on the error and processing time. Regularization, on the other hand, has no noticeable effect.

In the multi-slice reconstruction, the system matrix only needs to be constructed once, after which it is updated with the field map of the slice being reconstructed. This almost halves the reconstruction time per slice compared to single-slice reconstruction.

7.4. Three-Dimensional Reconstruction

The numerical results for three-dimensional reconstruction of a $128 \times 128 \times 30$ phantom are tabulated in Table 7.3.

Table 7.3: Two-norm reconstruction error and processing time of four scenarios of three-dimensional reconstruction after one iteration, averaged over three repetitions. The processing time includes the time required for field map estimation, system matrix construction, and image recovery, and excludes the time required for initialization and pre-processing. The phantom is of size $128 \times 128 \times 30$. Recovery of the data set requires almost 7 hours of processing.

Case	Field Map	Noise	Regularization	Two-Norm Error	Processing Time [s]	Processing Time per Slice [s]
1	Original	None	No	0.91	24911.87	830.40
2	Estimated	None	No	0.94	24903.15	830.10
3	Estimated	AWGN	No	0.94	24912.32	830.41
4	Estimated	AWGN	Yes	0.94	24923.57	830.79

Three-dimensional reconstruction requires significantly longer computation time per slice of data compared to two-dimensional single- and multi-slice. This is due to the scaling of the system matrix with the size of the data set. A cubic data set of $N \times N \times N$ data points requires a system matrix with N^6 points for three-dimensional reconstruction, while the same data set only needs a system matrix with N^4 points when reconstructing with the two-dimensional multi-slice technique, due to the absence of a third encoding direction.

Despite the increase in computation time, three-dimensional reconstruction is able to recover the image with less error compared to multi-slice reconstruction.

7.5. In Vivo Reconstruction

Dissimilar to reconstructed simulated phantoms, the residual error of recovered in vivo images cannot be calculated due to the absence of the original image. Hence, Table 7.4 only depicts the processing time for the various types of reconstruction.

Table 7.4: Processing time of in vivo reconstruction after one iteration, averaged over three repetitions. The processing time includes the time required for field map estimation, system matrix construction, and image recovery, and excludes the time required for initialization and pre-processing.

Type	Field Map	Regularization	Processing Time [s]	Processing Time per Slice [s]
2D	Measured	No	18.15	18.15
2D	Estimated	No	18.53	18.53
2D	Estimated	Yes	18.65	18.65
2D Multi-Slice	Measured	No	490.63	16.35
2D Multi-Slice	Estimated	No	299.99	10.00
2D Multi-Slice	Estimated	Yes	297.47	9.92
3D	Measured	No	24933.92	831.13
3D	Estimated	No	24906.74	830.22
3D	Estimated	Yes	24908.76	830.29

The processing times for the in vivo data set are similar to that of the simulate phantom. On exception is the two-dimensional multi-slice reconstruction with measured field map. The reconstruction algorithm required almost twice the time needed to recover the images in this scenario compared to other scenarios of multi-slice reconstruction. The source of this increase is currently unknown and needs to be investigated.

7.5.1. Comparison of 2D Multi-Slice and 3D Reconstruction

There is a significant difference between the two-dimensional multi-slice and three-dimensional reconstructed in vivo data set. This is due to how each initial data set was created. While the three-dimensional method utilized undersampling of the k-space data, the multi-slice technique used a subset of the FFT reconstructed slices. In order to give a fair comparison, both methods are used to reconstruct the same k-space undersampled data set. The resulting reconstructions are depicted in Figure 7.1.

The multi-slice reconstructed image displays higher contrast compared to the three-dimensional counterpart. Additionally, a sharper image is achieved.

To analyze the effect of the reconstruction methods on the SNR, the SNR before and after reconstruction are tabulated in Table 7.5. Here, the SNR is estimated by dividing the mean pixel intensity inside the object domain with the standard deviation of pixel intensity outside the object domain.

Table 7.5: The SNR before reconstruction, after pre-processing, and after reconstruction.

	2D Multi-Slice	3D
Before Reconstruction	2.18	1.98
After Pre-Processing	2.23	2.04
After Reconstruction	6.20	2.06

Contrary to initial assumption, the SNR of the reconstructed image is higher when reconstructed with the multi-slice method compared to three-dimensional algorithm.

The results indicate that two-dimensional multi-slice reconstruction is more effective than three-dimensional reconstruction, both in terms of processing time and SNR improvement.

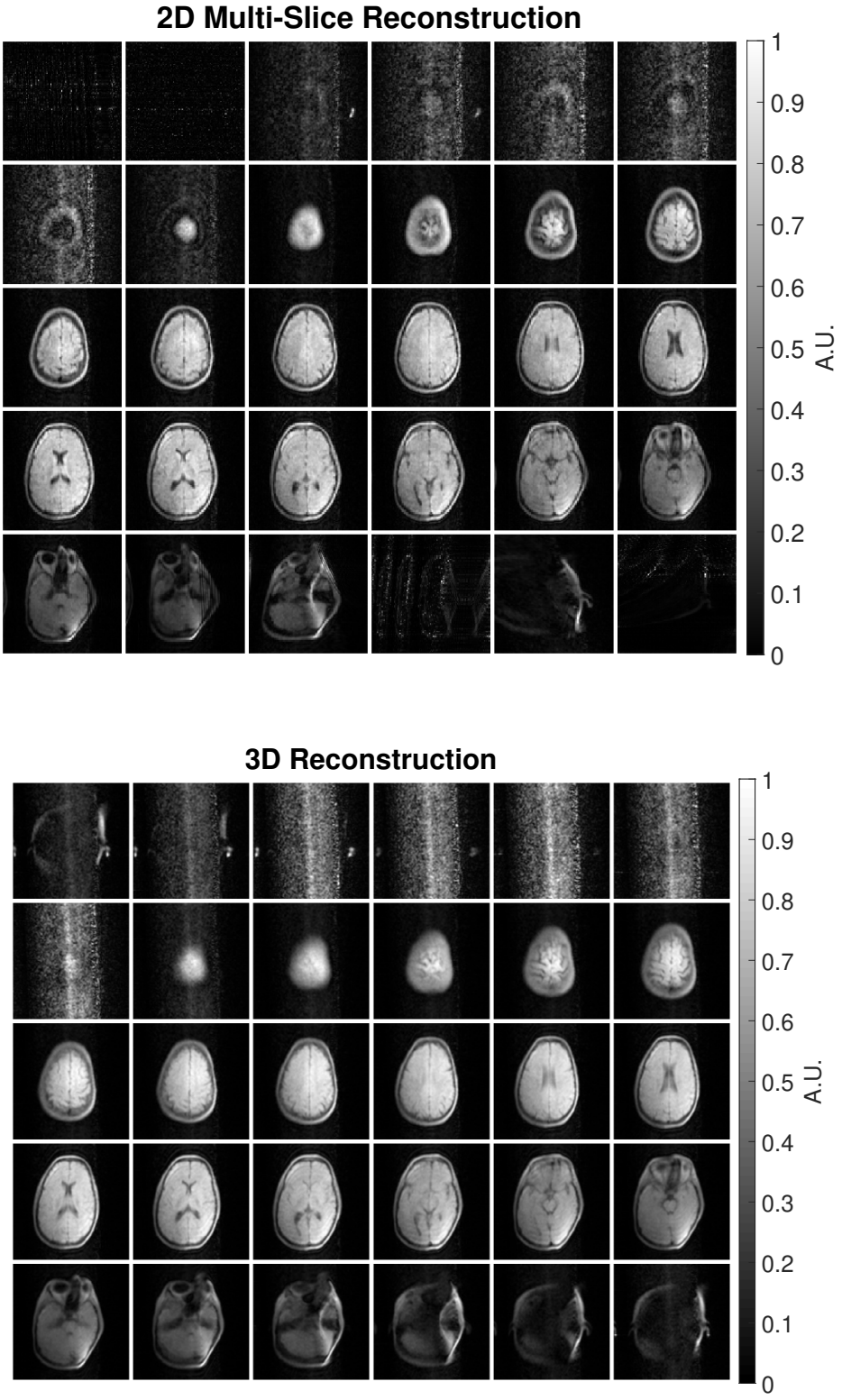

Figure 7.1: Recovered in vivo images utilizing multi-slice (top) and three-dimensional (bottom) reconstruction.

7.6. Implementation on the Low-Field Scanner

The results presented in this chapter are obtained with a computer which has an Intel Core i5-4670K CPU at 3.40 GHz, 16 GB of internal memory, and runs on 64-bit Windows 10 operating system.

Post development, the reconstruction algorithm was implemented on the computer responsible for scanning. This system has an AMD Ryzen 5 3400G CPU at 3.70 Ghz, 32 GB of internal memory, and runs on 64-bit Windows 10 operating system.

7.6.1. Code

Initial development of the algorithm was done in MATLAB due to its versatility. Due to the expensive licence required to operate MATLAB, the algorithm was also written in Python. This ensures future access to the code and is in line with other code written for the low-field scanner.

The main and reconstruction modules of the Python code can be found in Appendix B.

7.6.2. Processing Time

To evaluate the difference in reconstruction time between MATLAB and Python, a $64 \times 64 \times 10$ data set has been processed. The resulting computation times are displayed in Table 7.6 and visualized in Figure 7.2.

Table 7.6: The processing time of a $64 \times 64 \times 10$ in vivo data set in MATLAB and Python

Segment	Processing Time in MATLAB [s]	Processing Time in Python [s]
Initialization	1.77	2.55
Pre-processing	0.01	0.77
Field map estimation	0.25	0.13
System matrix construction	87.53	345.72
Update system matrix	36.28	119.34
CGLS	41.64	198.85
Plotting	4.68	0.21
Total reconstruction time	168.08	676.54
Total program time	174.69	680.35

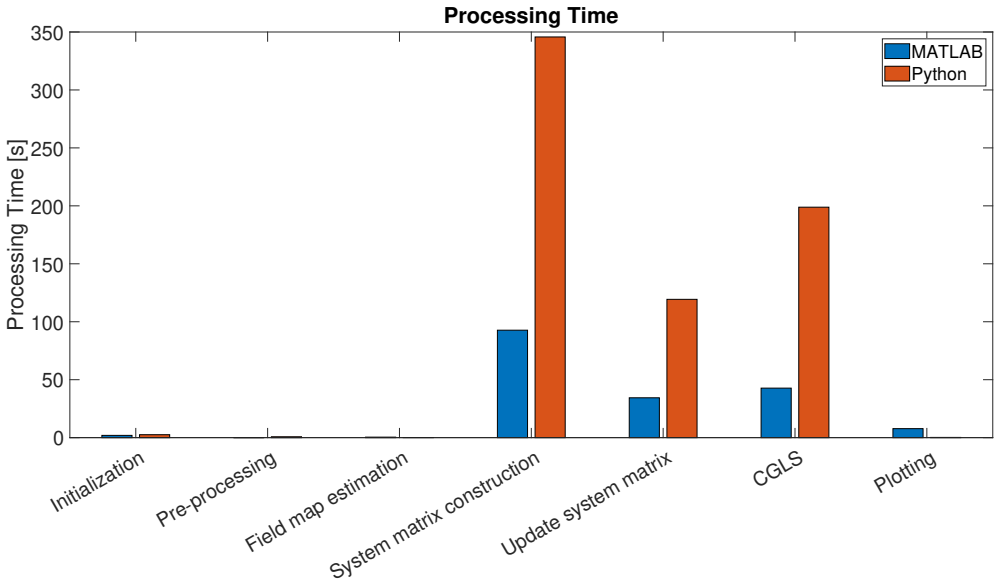

Figure 7.2: The processing time of a $64 \times 64 \times 10$ in vivo data set in MATLAB and Python.

Python shows a significant increase in processing time compared to MATLAB, despite the reconstruction parameters remaining unchanged. This might be due to MATLAB's utilization of built-in functions, which python lacks [32]. Nevertheless, Python will continue to be used in order to remain open-source.

Preliminary analysis of the Python implementation on the scanner computer has shown that a significant decrease in processing time is achieved, due to increased computational power and available internal memory. A rough estimate indicates that a data set of size $128 \times 128 \times 30$ can be reconstructed in one to two hours. This is a 70 to 80 percent reduction in processing time compared to the previous results.

8

Conclusion and Recommendations

In this thesis, the goal was to design and implement a reconstruction algorithm for three-dimensional data sets from a low-field MRI scanner. Data sets from the low-field scanner that are reconstructed using the classical FFT method contain distortion due to field inhomogeneities in the main magnetic field. These inhomogeneities arise from factors such as an imperfect Halbach array. To counteract the inhomogeneities, a map of the spatial variation in main magnetic field is required. With this field map, a model can be build, which allows for reconstruction of the data through the inverse model. To solve the inverse problem, CGLS was utilized.

Reconstruction was initially limited to two-dimensional data sets to analyse the efficacy and compare the results with those obtained in an identical study by Koolstra et al. [16]. Subsequently, an iterative framework was build. After each iteration, the field map is updated using the most recent reconstruction. This new map is then used to obtain a more accurate reconstruction.

A similar framework was applied to a two-dimensional multi-slice data set. In this framework, the individual slices are reconstructed sequentially. The resulting computation time per slice is roughly halved compared to single-slice reconstruction. This is due to the lack of system matrix construction at each iteration. The matrix only needs to be constructed once, after which it is updated with the field map of the slice being reconstructed.

The next step was to introduce a third encoding direction in the system matrix to allow for three-dimensional reconstruction. This resulted in a system matrix that required significantly more memory than available on a conventional computer. To circumvent this problem, the system matrix is split into sets of columns, which leads to a set of overdetermined systems. The concatenated results from the individual sets lead to the reconstructed volumetric image.

From visual inspection, the single- and multi-slice, as well as three-dimensional reconstruction are able to counteract the distortion in the images of several data sets.

The exception to accurate distortion correction is the outer slices of the in vivo data set. Here, gradient non-linearities are a non-negligable source of distortion. The current algorithm is limited to correction of distortion due to field inhomogeneities in the B_0 field, and is thus unable to correct other types of distortion. In future work on this algorithm, alternative sources of distortion need to be considered.

On the computer system used for development and testing, reconstruction of a data set of size $128 \times 128 \times 30$ requires roughly seven hours. This makes the algorithm infeasible for reconstruction of multiple data sets on a daily basis. Preliminary results however, indicate that the computer on which the algorithm is ultimately implemented is able to reconstructed the data sets in roughly one to two hours. Nevertheless, the algorithm needs to be optimized for reconstruction of large data sets before it can efficiently be utilized.

Thus, the proposed algorithm partially answers the original research question: **How can image distortion due to field inhomogeneities effectively be corrected in 3D imaging volumes?** The method is able to correct the image distortion, though requires additional work in order to reconstruct data in a timely manner. This might be achieved by investigating how Python can most effectively construct and utilize the system matrices.

Furthermore, multi-slice reconstruction leads to a larger SNR gain compared to three-dimensional recovery. Thus, two-dimensional multi-slice reconstruction is currently more effective than three-dimensional reconstruction. This might in part be due to subdivision of the system matrix. Hence, in future work, the effect of matrix subdivision of the SNR needs to be examined.

A

Noise Model

Gaining a deeper understanding of the noise present in the in vivo data can lead to an improved simulation model. To that end, this chapter of the appendix will investigate how the noise behaves, and from that, create a noise model.

A.1. In Vivo Data

The complete in vivo data set consists of $128 \times 128 \times 50$ data points. This k-space data set is transformed to the spatial domain by means of an FFT. The tailing slices are then disregarded due to unwanted artifacts (see Figure A.1). Following this, the data is normalized, after which the leading 16 slices are isolated. These isolated slices are absent of in vivo data, and thus solely contain noise.

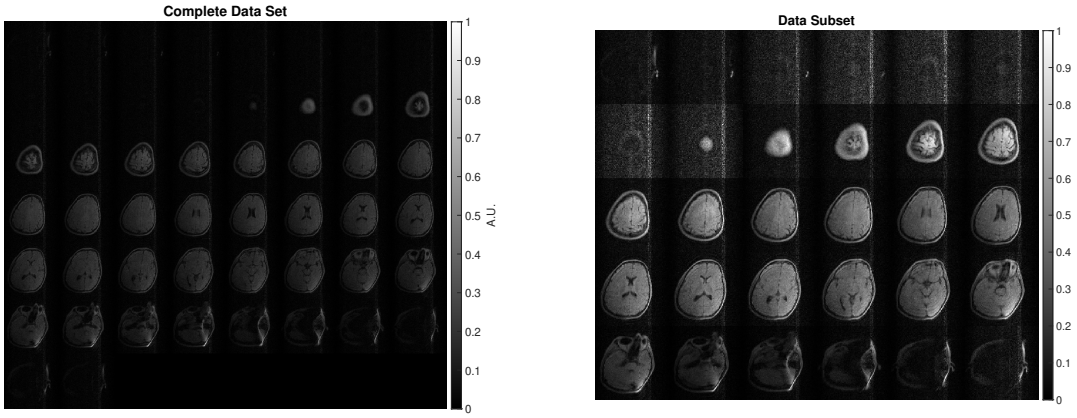

Figure A.1: Complete in vivo data set (left) and normalized susbet of in vivo data (right).

A.2. Probability Distribution

To visualize the distribution of noise, the noise histogram in Figure A.2 has been created. A Gaussian probability distribution has been fitted to the data and plotted alongside the histogram. The mean and standard deviation of the Gaussian probability density function (PDF) respectily, are $\mu = 0.1580$ and $\sigma = 0.0947$.

Though less common, a more accurate fit for the PDF is the Rician distribution. The PDF of the Rician distribution is described as:

$$f(x|s,\sigma) = \frac{x}{\sigma^2} e^{\left(\frac{-(x^2+s^2)}{2\sigma^2}\right)} I_0\left(\frac{xs}{\sigma^2}\right) \tag{A.1}$$

where $I_0(z)$ is the modified Bessel function.

The parameters of the best-fit Rician distribution are $s = 0.0014$ and $\sigma = 0.0380$. Figure A.2 illustrates the noise distribution in conjunction with a best-fit Guassian and Rician distribution.

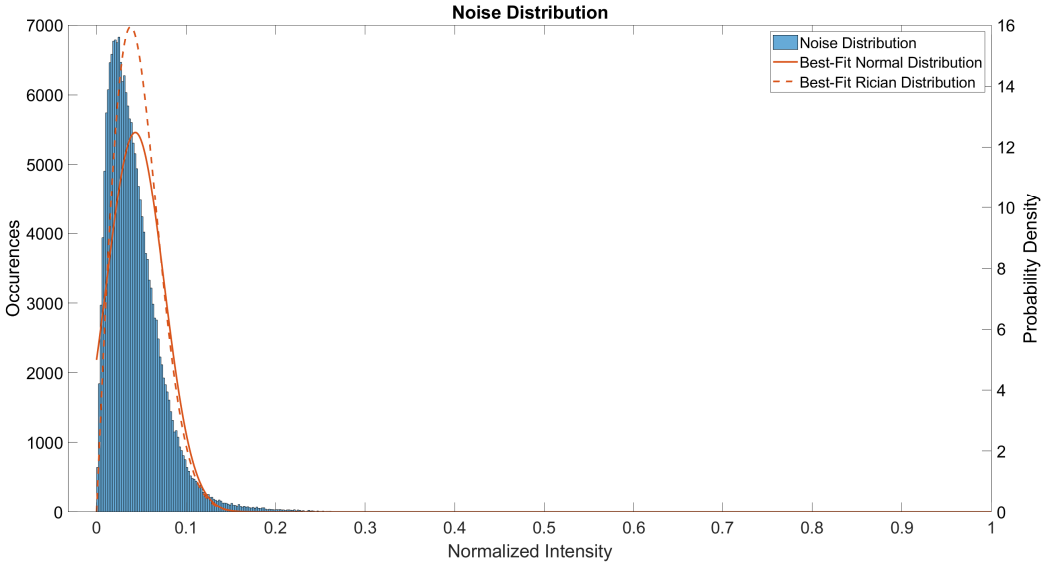

Figure A.2: Probability distribution of the noise. The solid line is a best-fit Gaussian distribution, while the dotted line is a best-fit Rician distribution

Alternatively, complex noise can be simulated, which more accurately represents noise in MRI data. The Rician distribution of noise on the magnitude of the signal suggests that the individual real and imaginary signal components contain AWGN [33]. In the aforementioned noise data, the mean and standard deviation of the real and imaginary data are approximated as: $\mu_R = \mu_I \approx 0$ and $\sigma_R = \sigma_I \approx 0.0379$.

B

Python Code

In this appendix, the main and reconstruction code module can be found. Note that both sets of code require additional functions that are not displayed here.

```
import numpy as np
import matplotlib.pyplot as plt
import time
import scipy.fft as fft
from Code.get_data_package import get_data, save_data
from Code.down_sample_k_space_package import down_sample_k_space
from Code.print_est_recon_time_package import print_est_recon_time
from Code.pre_process_data_package import pre_process_data
from Code.field_map_package import spherical_harmonics
from Code.reconstruction_package import reconstruction
from Code.colormap_package import get_color_map

# Show or hide figures of intermediate steps
# 0: Hide plots
# 1: Show plots
debug_plots = 1

# Time intermediate steps for debugging
# 0: Don't time
# 1: Time
debug_time = 1

# Automatically save data in folder 'Reconstructed'
# 0: Don't save reconstructed data
# 1: Save reconstructed data
save_data_bool = 1

# Location and name of saved data
save_name = 'Reconstructed/recon.csv'

# Location of data sets
location = [None] * 2
location[0] = '../Data/201001_B0_brein_SNR/201001_B0_brein_SNR/103'  # Location of data set 1
location[1] = '../Data/201001_B0_brein_SNR/201001_B0_brein_SNR/104'  # Location of data set 2

num_iter = 1  # Number of outer iterations to perform to find B0
M = 64  # Number of samples in phase   encoding direction (y)
N = 64  # Number of samples in readout encoding direction (x)
L = 10  # Number of samples in slice   encoding direction (z)

alpha = 10**2 # Regularization parameter

parula_map = get_color_map() # MATLAB-like colormap

```

```
45 if __name__ == '__main__':
46     tic_total = time.perf_counter()
47     tic_temp = time.perf_counter()
48 
49     print('Initiating...')
50 
51     print('Reading data')
52 
53     # Read the data
54     data = get_data(location)
55 
56     # Load parameters
57     from parameters import bandwidth as BW
58     from parameters import dwellTime as td
59     from parameters import echoShift as ts
60     from parameters import plane
61 
62     # Adjust scale of parameters
63     BW = BW * 1000
64     td = td * 10**(-6)
65     ts = ts * 10**(-6)
66 
67     # Downsample data
68     S1 = down_sample_k_space(data[2], M, N, L)
69     S2 = down_sample_k_space(data[3], M, N, L)
70 
71     # FFT reconstructed images before filtering
72     m1 = np.flip(fft.fftshift(fft.ifftn(S1)),1)
73     m2 = np.flip(fft.fftshift(fft.ifftn(S2)),1)
74 
75     time_init = time.perf_counter() - tic_temp # Initialization time
76     tic_temp = time.perf_counter()
77 
78     print_est_recon_time(M, N, L, num_iter)
79     print(f'Reconstruction will make {num_iter: d} iteration(s).')
80 
81     print('Pre-processing data')
82 
83     # Filter data
84     S1Filt = pre_process_data(S1)
85     S2Filt = pre_process_data(S2)
86 
87     # FFT reconstructed images after filtering
88     m1Filt = np.flip(fft.fftshift(fft.ifftn(S1Filt)), 1)
89     m2Filt = np.flip(fft.fftshift(fft.ifftn(S2Filt)), 1)
90 
91     time_pre_proc = time.perf_counter() - tic_temp # Pre-processing time
92     tic_temp = time.perf_counter()
93 
94     # Estimate field map
95     print('Applying spherical harmonics')
96     df0SH = spherical_harmonics(m1Filt, m2Filt, ts)
97 
98     time_field_map = time.perf_counter() - tic_temp # Field map estimation time
99     tic_temp = time.perf_counter()
100 
101     # Apply reconstruction
102     print('Starting reconstruction')
103     (pRec1, pRec2, time_pcg, time_sysmat, time_update_sysmat) = reconstruction(m1Filt, m2Filt
        , S1Filt, S2Filt, df0SH, BW, num_iter, ts, plane, debug_plots, alpha)
104 
105     pRec1 = np.flip(pRec1, 1)
106     pRec2 = np.flip(pRec2, 1)
107 
108     time_recon = time.perf_counter() - tic_temp
109     tic_temp = time.perf_counter()
110 
111     # Prepare plots (similare to montage() in MATLAB)
112     num_plots = pRec1.shape[2]
113     rows_plots = int(np.ceil(np.sqrt(num_plots)))
```

```
114     counter = 0
115
116     # Plot MB reconstructed images
117     fig, axs = plt.subplots(rows_plots, rows_plots)
118     for i in range(rows_plots):
119         for j in range(rows_plots):
120             if counter < num_plots:
121                 axs[i, j].imshow(np.abs(pRec1[:, :, counter]), cmap='gray')
122                 counter += 1
123
124     time_plot = time.perf_counter() - tic_temp
125
126     # Save reconstructed images
127     if save_data_bool == 1:
128         save_data(pRec1, save_name)
129
130     # Display processing time
131     time_process = time.perf_counter() - tic_total
132     print(f"Took {time_process:0.4f} seconds")
133     print('Done')
134
135     if debug_time == 1:
136         print('\n---------------------------------')
137         print(f'Processing Times ({M:d}x{N:d}x{L:d}, {num_iter:d} iteration(s) \n')
138         print(f'Initialization: {time_init:.2f}s')
139         print(f'Pre-processing: {time_pre_proc:.2f}s')
140         print(f'Field map estimation: {time_field_map:.2f}s')
141         print(f'System matrix construction: {time_sysmat:.2f}s')
142         print(f'Update system matrix: {time_update_sysmat:.2f}s')
143         print(f'PCG: {time_pcg:.2f}s')
144         print(f'Plotting: {time_plot:.2f}s')
145         print(f'Total reconstruction time: {time_recon:.2f}s')
146         print(f'Total program time: {time_process:.2f}s')
147         print('---------------------------------')
```

Listing B.1: The main Python code module used to execute the reconstruction pipeline.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.sparse as sp
import scipy.sparse.linalg as la
import scipy.optimize as op
import time
import multiprocessing as mp
from sys import exit
from psutil import virtual_memory
from Code.field_map_package import *
from Code.colormap_package import *
from Code.field_map_package import *


def reconstruction(m1, m2, S1, S2, df0, bw, num_iter, ts, plane, debug_plots, alpha):
    time_pcg = 0
    time_sysmat = 0
    time_update_sysmat = 0
    (m, n, l) = S1.shape

    # Determine the number of sets to divide the system matrix into based on available RAM
    mem_avail = virtual_memory().total # Virtual memory available
    mem_avail = mem_avail / 8 # Set margin
    mem_total = ((m * n * l) ** 2) * 16 # Total memory needed for complete system matrix

    if mem_total >= mem_avail:
        mem_per_col = (m * n * l) * 16 # Memory needed per column of the system matrix
        num_cols = int(mem_avail / mem_per_col) # Maximum number of rows to handle per loop
        if num_cols < 1:
            print('System is to large to reconstruct.')
            exit()

        num_sets = int(np.ceil((m * n * l) / num_cols)) # Number of sets to divide system
            matrix into

        # Determine order of sets
        O = order(num_sets)
    else:
        num_sets = 1
        num_cols = m * n * l
        O = np.array([0])

    print(f'The system will be split into {num_sets:d} set(s), with each {num_cols:d} column(
        s).')

    parula_map = get_color_map() # Colormap for plots (based on default Matlab colormap)

    df0SH = df0
    df0 = np.real(to_1D(df0).T)
    S1 = to_1D(S1)
    S2 = to_1D(S2)

    # Err = norm_err(p, m1)
    (Gx, Gy, Gz, Tx, Ty, Tz) = create_gradients(m, n, l, bw)

    global A
    global a
    a = alpha**2

    mask = make_mask(m1)
    df0_mask = estimate_field_map(m1, m2, ts)
    df0_mask[mask == 0] = 0

    # Prepare plots (similare to montage() in MATLAB)
    num_plots = l
    rows_plots = int(np.ceil(np.sqrt(num_plots)))

    # Plot MB reconstructed images
```

```
    fig, axs = plt.subplots(rows_plots, rows_plots)

    # Perform outer iterations that update B0 estimate
    for counter in range(num_iter):
        print(f'Iteration {(counter+1):d} of {(num_iter):d}')

        p_rec_1 = np.zeros((m * n * l, 1), dtype='c8')
        p_rec_2 = np.zeros((m * n * l, 1), dtype='c8')

        tic_time_remaining = time.perf_counter()

        # Sequentially run through subsets of the system matrix
        for set_count in range(num_sets):
            col_count = O[set_count]

            col_count_start = int(num_cols * col_count)
            if set_count == num_sets:
                col_count_stop = int(m * n * l)
            else:
                col_count_stop = int(num_cols * (col_count + 1))

            print(f'Set {(set_count+1):d} of {num_sets}')

            # Update time estimate every set
            if np.mod((set_count + 1), 1) == 0 and set_count > 0:
                time_remaining = ((time.perf_counter() - tic_time_remaining) / set_count) * (
                    num_sets - set_count)
                time_remaining_hour = int(time_remaining / 3600)
                time_remaining -= time_remaining_hour * 3600
                time_remaining_min = int(time_remaining / 60)
                time_remaining -= time_remaining_min * 60
                time_remaining_sec = int(time_remaining)
                print(f'Estimated time remaining: {time_remaining_hour:d}h, {
                    time_remaining_min:d}m, {time_remaining_sec:d}s')

            tic = time.perf_counter()
            E = make_system_matrix(col_count_start, col_count_stop, Gx, Gy, Gz, Tx, Ty, Tz)
            time_sysmat = time_sysmat + (time.perf_counter() - tic)

            tic = time.perf_counter()
            A = update_system_matrix(E, df0, col_count_start, col_count_stop, Tx)
            time_update_sysmat = time_update_sysmat + (time.perf_counter() - tic)

            del E

            tic = time.perf_counter()
            p_rec_1_temp, p_rec_2_temp = recon(S1, S2, A, col_count_start, col_count_stop,
                counter, num_iter)
            time_pcg = time_pcg + (time.perf_counter() - tic)

            del A

            p_rec_1[col_count_start:col_count_stop] = p_rec_1_temp
            p_rec_2[col_count_start:col_count_stop] = p_rec_2_temp

            if debug_plots == 1:
                # Show intermediate plots
                temp = to_3D(p_rec_1, m, n, l)

                fig, axs = plt.subplots(2, 2)
                cs1 = axs[0, 0].imshow(np.abs(m1[:, :, int(l/2)]), cmap='gray')
                fig.colorbar(cs1, ax=axs[0, 0])
                cs2 = axs[0, 1].imshow(np.real(df0_mask[:, :, int(l/2)]), cmap=parula_map,
                    vmin=-1500, vmax=1500)
                fig.colorbar(cs2, ax=axs[0, 1])
                cs3 = axs[1, 0].imshow(np.flip(np.abs(temp[:, :, int(l/2)]), 1), cmap='gray')
                fig.colorbar(cs3, ax=axs[1, 0])
                cs4 = axs[1, 1].imshow(np.flip(np.real(df0SH[:, :, int(l/2)]), 1), cmap=
                    parula_map, vmin=-1500, vmax=1500)
```

```
                fig.colorbar(cs4, ax=axs[1, 1])
                plt.show()


        p_rec_1 = to_3D(p_rec_1, m, n, l)
        p_rec_2 = to_3D(p_rec_2, m, n, l)

        df0 = spherical_harmonics(p_rec_1, p_rec_2, ts)

    return p_rec_1, p_rec_2, time_pcg, time_sysmat, time_update_sysmat


def recon(S1, S2, E, count_start, count_stop, counter, num_iter):
    tol = np.float_power(10, -2)
    max_iter = 100

    x0 = np.zeros((count_stop - count_start, 1), dtype='complex')

    b1 = np.reshape((np.conj(E.T) @ S1), (-1, 1))
    b2 = np.reshape((np.conj(E.T) @ S2), (-1, 1))

    fun = la.LinearOperator((b1.shape[0], b1.shape[0]), f)
    p_rec_1 = la.cgs(fun, b1, x0, tol, max_iter)

    if counter != num_iter:
        p_rec_2 = la.cgs(fun, b2, x0, tol, max_iter)
    else:
        p_rec_2 = np.zeros(p_rec_1[0].shape)

    return np.reshape(p_rec_1[0], (-1, 1)), np.reshape(p_rec_2[0], (-1, 1))


def f(v):
    return (np.conj(A.T) @ np.reshape((A @ v), (-1, 1))) + np.reshape((a * v), (-1, 1))


def update_system_matrix(E, df0, count_start, count_stop, Tx):
    df0 = np.reshape(df0[0, count_start:count_stop], (1, -1))
    Aw = np.exp(-1j * 2 * np.pi * (Tx @ df0))
    A = Aw * E

    return A


def make_system_matrix(count_start, count_stop, Gx, Gy, Gz, Tx, Ty, Tz):
    gx = np.reshape(Gx[0, count_start:count_stop], (1, -1))
    gy = np.reshape(Gy[0, count_start:count_stop], (1, -1))
    gz = np.reshape(Gz[0, count_start:count_stop], (1, -1))

    AGx = np.exp(-1j * 2 * np.pi * (Tx @ gx))
    AGy = np.exp(-1j * 2 * np.pi * (Ty @ gy))
    E = AGx * AGy
    del AGx
    del AGy
    AGz = np.exp(-1j * 2 * np.pi * (Tz @ gz))
    E = E * AGz

    return E


def create_gradients(m, n, l, bw):
    td = 1 / bw  # Time step (x) [s]
    tpeyd = td  # Phase step (y) [s]
    tpezd = td  # Slice step (z) [s]

    # Generate gradients
    Gx = np.linspace(-bw / 2, bw / 2, n + 1)
    Gx = Gx[:-1]  # Remove last element
    Gy = np.linspace(-bw / 2, bw / 2, m + 1)
```

```
    Gy = Gy[:-1]  # Remove last element
    Gz = np.linspace(-bw / 2, bw / 2, l + 1)
    Gz = Gz[:-1]  # Remove last element

    # Expand over 2D surface
    Gx2 = np.ones((m, 1)) * Gx
    Gy2 = np.ones((n, 1)) * Gy
    Gz2 = np.ones((n, 1)) * Gz

    # Expand over 3D volume
    Gx3 = np.moveaxis(np.tile(Gx2, (l, 1, 1)), [0, 1, 2], [2, 1, 0])
    Gy3 = np.moveaxis(np.tile(Gy2, (l, 1, 1)), [0, 1, 2], [2, 0, 1])
    Gz3 = np.tile(Gz2, (m, 1, 1))

    # Convert matrices to vectors
    Gx = np.reshape(Gx3, (1, -1), order='F')
    Gy = np.reshape(Gy3, (1, -1), order='F')
    Gz = np.reshape(Gz3, (1, -1), order='F')

    # Create time-step vectors
    t = np.reshape(np.linspace(-n / 2, (n / 2) - 1, n), (1, -1))
    tpey = np.reshape(np.linspace(-m / 2, (m / 2) - 1, m), (1, -1))
    tpez = np.reshape(np.linspace(-l / 2, (l / 2) - 1, l), (1, -1))

    Tx = np.reshape(np.tile(t * td, (m * l, 1)), (-1, 1))

    for index in range(m):
        tauy = tpey[0, index] * tpeyd
        Tauy = np.tile(tauy, (n, 1))
        if index == 0:
            Ty = Tauy
        else:
            Ty = np.append(Ty, Tauy)

    Ty = np.reshape(Ty, (-1, 1))
    Ty = np.tile(Ty, (l, 1))

    for index in range(l):
        tauz = tpez[0, index] * tpezd
        Tauz = np.tile(tauz, (m * n, 1))
        if index == 0:
            Tz = Tauz
        else:
            Tz = np.append(Tz, Tauz)
    Tz = np.reshape(Tz, (-1, 1))

    return Gx, Gy, Gz, Tx, Ty, Tz


def norm_err(p, m):
    dif = m - p
    ndif = np.linalg.norm(dif, ord=2)
    nm = np.linalg.norm(p, ord=2)
    err = ndif / nm

    return err


def order(n):
    A = np.linspace(0, n - 1, n)
    LA = n
    A1 = A[0:int(np.round(LA/2))]
    A1 = A1[::-1]
    A2 = A[int(np.round(LA/2)):]

    B = np.zeros(A.shape)
    counter = 0

    for i in range(1, LA, 2):
```

```
269             B[i-1] = A1[counter]
270             B[i] = A2[counter]
271             counter += 1
272
273         if np.mod(LA, 2) == 1:
274             O = B[0:-2]
275         else:
276             O = B
277
278         return O
279
280
281 def to_1D(d3):
282         temp = np.moveaxis(d3, [0, 1, 2], [1, 0, 2])
283         d1 = np.reshape(temp, (-1, 1), order='F')
284
285         return d1
286
287
288 def to_3D(d1, m, n, l):
289         temp = np.reshape(d1, (m, n, l), order='F')
290         d3 = np.moveaxis(temp, [0, 1, 2], [1, 0, 2])
291
292         return d3
```

Listing B.2: The Python module used to reconstruct the data set.

Bibliography

[1] P. C. LAUTERBUR, "Image formation by induced local interactions: Examples employing nuclear magnetic resonance," *Nature*, vol. 242, no. 5394, pp. 190–191, Mar. 1973, ISSN: 1476-4687.

[2] A. Alkemade *et al.*, "The amsterdam ultra-high field adult lifespan database (ahead): A freely available multimodal 7 tesla submillimeter magnetic resonance imaging database," *NeuroImage*, vol. 221, p. 117 200, 2020, ISSN: 1053-8119.

[3] *Ingenia elition 3.0t x*, Philips. [Online]. Available: `https://www.philips.nl/healthcare/product/HC781358/ingenia-elition-30t-x#galleryTab=PI`.

[4] N. B. Smith and A. Webb, *Introduction to medical imaging: physics, engineering and clinical applications*. Cambridge university press, 2010.

[5] L. L. Wald, P. C. McDaniel, T. Witzel, J. P. Stockmann, and C. Z. Cooley, "Low-cost and portable mri," *Journal of Magnetic Resonance Imaging*, vol. 52, no. 3, pp. 686–696, 2020.

[6] S. Crozier and D. M. Doddrell, "A design methodology for short, whole-body, shielded gradient coils for mri," *Magnetic Resonance Imaging*, vol. 13, no. 4, pp. 615–620, 1995, ISSN: 0730-725X.

[7] S. Geethanath and J. T. Vaughan Jr., "Accessible magnetic resonance imaging: A review," *Journal of Magnetic Resonance Imaging*, vol. 49, no. 7, e65–e77, 2019.

[8] M. Figini *et al.*, *Image quality transfer enhances contrast and resolution of low-field brain mri in african paediatric epilepsy patients*, 2020.

[9] C. Z. Cooley *et al.*, *A portable brain mri scanner for underserved settings and point-of-care imaging*, 2020.

[10] T. O'Reilly, W. M. Teeuwisse, D. de Gans, K. Koolstra, and A. G. Webb, "In vivo 3d brain and extremity mri at 50 mt using a permanent magnet halbach array," *Magnetic Resonance in Medicine*, Jun. 2020.

[11] T. Miyamoto, H. Sakurai, H. Takabayashi, and M. Aoki, "A development of a permanent magnet assembly for mri devices using nd-fe-b material," *IEEE Transactions on Magnetics*, vol. 25, no. 5, pp. 3907–3909, 1989.

[12] M. Nakagomi *et al.*, "Development of a small car-mounted magnetic resonance imaging system for human elbows using a 0.2t permanent magnet," *Journal of Magnetic Resonance*, vol. 304, pp. 1–6, 2019, ISSN: 1090-7807.

[13] M. R. Del Bigio, "Neuropathological changes caused by hydrocephalus," *Acta Neuropathologica*, vol. 85, no. 6, pp. 573–585, May 1993, ISSN: 1432-0533.

[14] W. E. Dandy, "Experimental hydrocephalus," *Annals of surgery*, vol. 70, no. 2, pp. 129–142, Aug. 1919, ISSN: 0003-4932.

[15] P. Przyborowska, Z. Adamiak, M. Jaskolska, and Y. Zhalniarovich, "Hydrocephalus in dogs: A review.," *Veterinarni Medicina*, vol. 58, no. 6, pp. 73–80, 2013.

[16] K. Koolstra, T. O'Reilly, P. Boörnert, and A. G. Webb, "Image distortion correction for mri in low field permanent magnet systems with strong b_0 inhomogeneity," *Magnetic Resonance Materials in Physics, Biology and Medicine*, Jun. 2020.

[17] M. L. de Leeuw den Bouter, M. B. van Gijzen, and R. F. Remis, "Conjugate gradient variants for ℓ_p-regularized image reconstruction in low-field mri," *SN Applied Sciences*, vol. 1, no. 12, p. 1736, Nov. 2019, ISSN: 2523-3971.

[18] W. S. Hinshaw and A. H. Lent, "An introduction to nmr imaging: From the bloch equation to the imaging equation," *Proceedings of the IEEE*, vol. 71, no. 3, pp. 338–350, 1983.

[19] B. P. Kibble and G. J. Hunt, "A measurement of the gyromagnetic ratio of the proton in a strong magnetic field," *Metrologia*, vol. 15, no. 1, pp. 5–30, Jan. 1979.

[20] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex fourier series,” *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, Dec. 1965, Full publication date: Apr., 1965.

[21] J. A. Fessler, “Model-based image reconstruction for mri,” *IEEE Signal Processing Magazine*, vol. 27, no. 4, pp. 81–89, 2010.

[22] K. Sekihara, S. Matsui, and H. Kohno, “Nmr imaging for magnets with large nonuniformities,” *IEEE Transactions on Medical Imaging*, vol. 4, no. 4, pp. 193–199, 1985.

[23] N.-k. Chen and A. M. Wyrwicz, “Correction for epi distortions using multi-echo gradient-echo imaging,” *Magnetic Resonance in Medicine*, vol. 41, no. 6, pp. 1206–1213, 1999.

[24] A. K. Funai, J. A. Fessler, D. T. B. Yeo, V. T. Olafsson, and D. C. Noll, “Regularized field map estimation in mri,” *IEEE Transactions on Medical Imaging*, vol. 27, no. 10, pp. 1484–1494, 2008.

[25] L.-C. Man, J. M. Pauly, and A. Macovski, “Multifrequency interpolation for fast off-resonance correction,” *Magnetic Resonance in Medicine*, vol. 37, no. 5, pp. 785–792, 1997.

[26] S. J. Doran, L. Charles-Edwards, S. A. Reinsberg, and M. O. Leach, “A complete distortion correction for MR images: I. gradient warp correction,” *Physics in Medicine and Biology*, vol. 50, no. 7, pp. 1343–1361, Mar. 2005.

[27] L. A. Shepp and B. F. Logan, “The fourier reconstruction of a head section,” *IEEE Transactions on Nuclear Science*, vol. 21, no. 3, pp. 21–43, 1974.

[28] M. R. Hestenes, E. Stiefel, *et al.*, “Methods of conjugate gradients for solving linear systems,” *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[29] P. Sonneveld, “Cgs, a fast lanczos-type solver for nonsymmetric linear systems,” *SIAM journal on scientific and statistical computing*, vol. 10, no. 1, pp. 36–52, 1989.

[30] M. C. Schabel, *3d shepp-logan phantom*, MATLAB Central File Exchange, 2005. [Online]. Available: `https://nl.mathworks.com/matlabcentral/fileexchange/9416-3d-shepp-logan-phantom`.

[31] P. C. Hansen, “Analysis of discrete ill-posed problems by means of the l-curve,” *SIAM Review*, vol. 34, no. 4, pp. 561–580, 1992.

[32] P. F. Guedes and E. G. Nepomuceno, “Some remarks on the performance of matlab, python and octave in simulating dynamical systems,” *arXiv preprint arXiv:1910.06117*, 2019.

[33] A. Cárdenas-Blanco, C. Tejos, P. Irarrazaval, and I. Cameron, “Noise in magnitude magnetic resonance images,” *Concepts in Magnetic Resonance Part A: An Educational Journal*, vol. 32, no. 6, pp. 409–416, 2008.