Sparse Sampling of Velocity MRI

Sparse Sampling of Velocity MRI

By
Yogesh Chinta Venkateswarao B.E.
A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Applied Science

McMaster University

Master of Applied Science(2011) McMaster University
Computational Engineering and Science Hamilton, Ontario

TITLE: Sparse Sampling of Velocity MRI

AUTHOR: Yogesh Chinta Venkateswarao B.E.

SUPERVISOR: Dr. Christopher Anand

NUMBER OF PAGES: xi, 61

LEGAL DISCLAIMER: This is an academic research report. I, my supervisor, defence committee, and university, make no claim as to the fitness for any purpose, and accept no direct or indirect liability for the use of algorithms, findings, or recommendations in this thesis.

**Abstract**

Standard MRI is used to image objects at rest. In addition to standard MRI images, which measure tissues at rest, Phase Contrast MRI can be used to quantify the motion of blood and tissue in the human body. The current method used in Phase Contrast MRI is time consuming. The development of new trajectories has minimized imaging time, but creates sub-sampling errors. The proposed method uses regularization of velocities and proton densities to eliminate errors arising from k-space under-sampling.

**Acknowledgments**

I express my sincere gratitude to my supervisor, Dr. Christopher Anand, for his neverending helpfulness, encouragement and guidance during the course of my Master's degree.

I would like to thank my late grandparents, Shri G. Sadagopan and Smt S. Suseela, my mother, Dr.C. Sridevi and my aunt, Ms S. Vijayalakshmi and both my brothers (Lak and Deepu) for their support and constant encouragement. I would also like to thank my friends, especially my fellow students at ITB 229 for their help and encouragement.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction and Background Information

## 1.1 Document Structure

We begin with an introduction to the study of Magnetic Resonance Imaging by giving a basic summary of its governing equations and related principles of phase-contrast angiography. We then introduce the optimization model, which is the focus of this thesis in order to describe the use of under-sampling in k-space and give a basic idea of what can be done to improve its results. The functions used in the optimization, including the penalties, are described and numerical examples of this method are provided. Conclusions are then drawn and required future work is discussed. Matlab codes used in this thesis are described in the appendix.

## 1.2 Principles of MRI

Protons, electrons and neutrons in a magnetic field possess a nuclear spin angular momentum which is a fundamental property of nature; they act like tiny toy tops that wobble as they spin. The rate of the wobbling or precession is what is known as magnetic resonance or Larmor frequency [HBTV99], [Hah60]. The magnetic resonance signal is acquired from precessing magnetic moments of protons. We refer to this magnetic resonance as spin. The principle of Magnetic Resonance Imaging (MRI) is based on the interaction of these spins with three types of magnetic fields:

1. the static magnetic field, around which the protons precess.

2. the radio frequency field, used for tipping the protons to generate a signal that can be read.

3. the gradient fields, used for imaging the position of the object being measured, and to quantify velocity and other properties of tissues.

In the absence of an external magnetic field, the spins are oriented randomly and the net magnetic moment is zero. However, in the presence of an external magnetic field, the spins will align with or against the external magnetic field, but the net magnetic moment vector is in alignment with the external field. Also the nuclear spins exhibit resonance at the Larmor frequency. The exposure of an object or a person to radio-frequency (RF) pulse at the Larmor frequency causes the net magnetization to spiral away from the main magnetic field. After a certain length of time, the net magnetization vector rotates 90 degrees and lies in the transverse or $x$-$y$ plane. It is in this position that the transverse component is measured. The pulse sequences between the initial RF pulse and the measurement determine the meaning of the signal. This is called the preparation phase, and it is possible to generate signals weighted by the amount of free water, fat, white matter, grey matter etc. If each of the regions of spin was to experience a unique magnetic field we would be able to image their positions. A gradient in the magnetic field is what will allow us to accomplish this. A magnetic field gradient is a variation in the magnetic field with respect to position. In this thesis we will be concerned with the quantification of velocity using images in which the relative angle of the transverse component is proportional to the velocity of the tissue in each voxel (discrete tissue unit volume).

### 1.2.1 Phase Contrast Angiography

Phase-Contrast Angiography is one of the important techniques used in Magnetic Resonance Angiography (MRA) that can be used to directly image flow in arteries, veins and cerebrospinal fluid [NMP86]. Phase-contrast imaging has excellent background suppression, allows variable velocity encoding, and provides directional flow information [IL93], [NFL86]. Two-dimensional phase-contrast imaging is useful in the assessment of major vascular structures. Three-dimensional phase-contrast imaging is also useful in depicting small and medium-sized aneurysms. Other applications of phase-contrast angiography also concern cerebral venous imaging (thrombosis of cerebral veins, fistula)

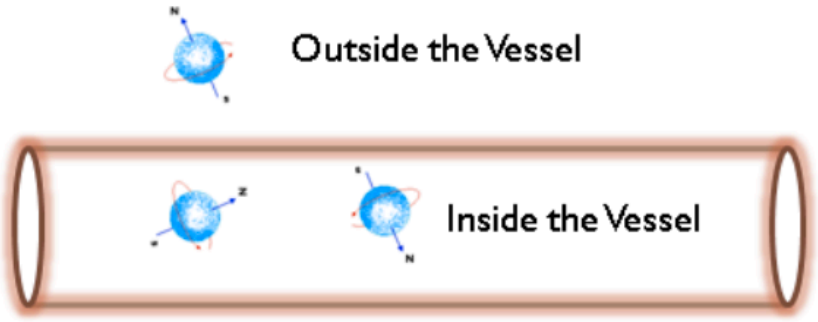


Figure 1.1: Protons at rest in a static magnetic field are randomly distributed with a small net moment parallel to the main field.

and may also be used after injection of a contrast agent. The unique aspect of phase-contrast angiography is that it can detect both high velocity flow and extremely slow flow. The application of a stronger flow-encoding gradient makes it possible to detect slow flow, whereas the application of a weaker flow-encoding gradient leads to the detection of high-velocity flow [IL93]. During initial magnetization, the net magnetic moment of the protons are tipped by the initial radio-frequency (RF) pulse as shown in Fig. 1.2.

Phase-contrast angiography detects the phase changes of the proton spins, which the moving spins acquire proportional to their velocities. Here phase means the transverse component (induced by the radio frequency pulse) that is being measured. At rest protons point in different directions, but the net moment is parallel to the field as shown in Fig. 1.1. Fig. 1.3 shows that during the gradient pulse the axis of rotation rotates in the transverse plane with an angle which is a function of the applied field variation and the flow/velocity of the matter they are in. After the gradient pulse the proton's magnetization, precessing in the transverse plane, is measured: recorded as a complex number, its phase is a function of the applied field variation (G) and the flow/velocity (V).

The differential equation for the evolution of magnetization (M) over short time periods in the presence of an external magnetic field B is defined

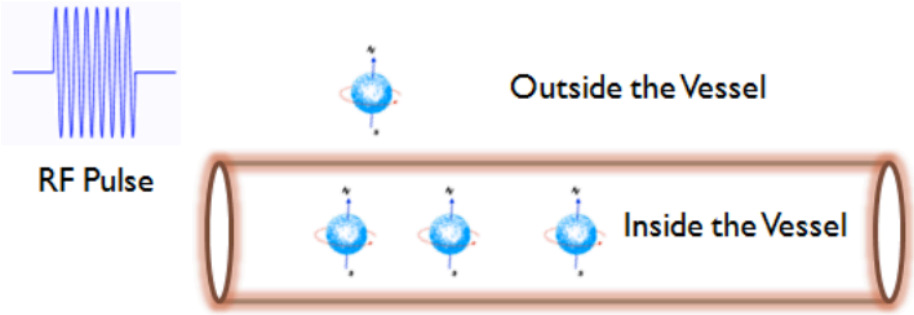


Figure 1.2: The net magnetic moments are tipped into the transverse plain by the RF pulse.

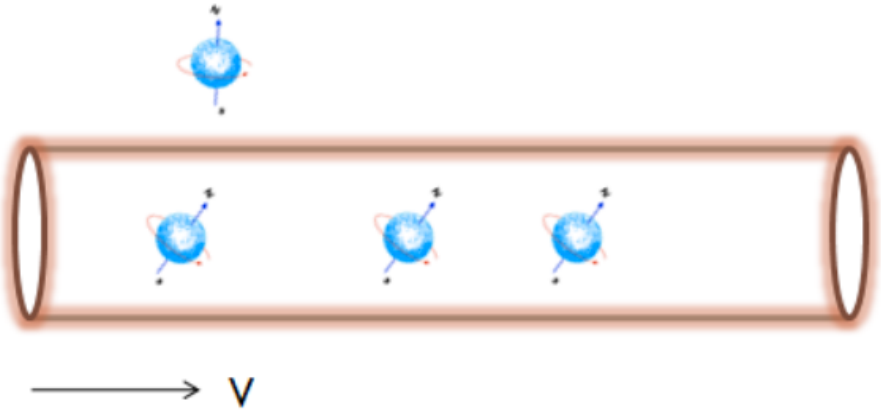


Figure 1.3: Net moments after a velocity-encoding bipolar gradient pulse.

by the empirical Eq. (1.1) referred to as the Bloch equation.

$$\frac{dM}{dt} = \gamma M \times B \tag{1.1}$$

When there is no radio-frequency pulse the external magnetic field (B) is represented by $B_Z = B_0 + G_x x + G_y y + G_z z$, where $B_Z$ is the Z component of the the external magnetic field B. The equation of translational motion of an isochromat of spins during sampling can be described by the expression $x(t) = x_0 + v_x t$. According to magnetic resonance (MR) theory [HBTV99], the fact that the principal action of the proton spin in a constant magnetic field is a rotation in the two-dimensional transverse plane suggests that a complex number will be useful. The differential equation described in Eq. (1.1) could further be simplified by employing the complex representation as shown in Eq. (1.2), We can categorize Eq. (1.2) as the general solution to the Bloch equation where the Z component of the external magnetic field ($B_Z$) and the gradient is constant when there is no radio-frequency pulse. The scaling of the Z component of the the external magnetic field ($B_Z$) is described by the Eq. (1.3).

$$M_x(t) + iM_y(t) = (M_x(0) + iM_y(0)).e^{iB_Z.t} \tag{1.2}$$

$$M_x(t) + iM_y(t) = (M_x(0) + iM_y(0)).e^{i\int_0^t B_Z.t^{'}\,dt^{'}} \tag{1.3}$$

From Eq. (1.3), the expressions $(M_x(0) + iM_y(0)).e^{i\int_0^1 G_x x(t^{'})dt^{'}}$ and $(M_x(0) + iM_y(0)).e^{i\int_1^2 G_x x(t^{'})dt^{'}}$ is derived when $t < 1$ and $t > 1$ respectively. The positive and negative gradients derived from the bipolar gradient shown in Fig. 1.4 is represented by the expressions $(M_x(0) + iM_y(0).e^{iG_x(x_0+\frac{1}{2}.V)}$ and $(M_x(0) + iM_y(0)).e^{-iG_x(x_0+\frac{3}{2}.V)}$. The velocity dependent phase at the echo of a simple bipolar gradient pulse shows minor variation across a single voxel and, after the data are Fourier transformed, the image for laminar flow in a pixel centred at position $(i, j)$ is given by Eq. (1.4), where $\rho$ is the proton density, $V$ is the velocity of blood and the operator $(\odot)$ signifies point-wise multiplication.

$$\rho_{i,j} \odot e^{-iG_x x \cdot V_{i,j}} \tag{1.4}$$

In the next section we talk about k-space and Fourier transform essential to our understanding of this thesis.

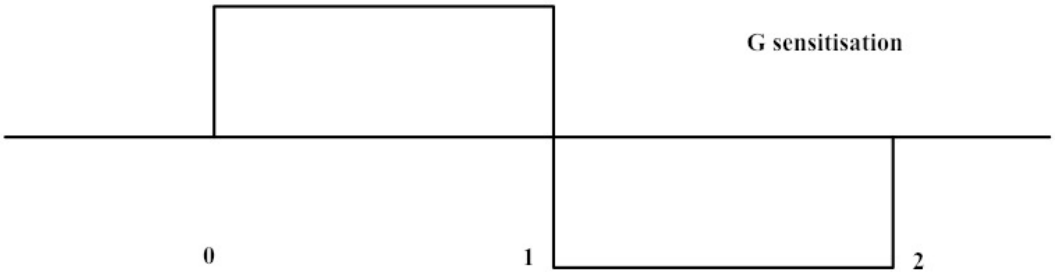


Figure 1.4: Bipolar gradient pulse.

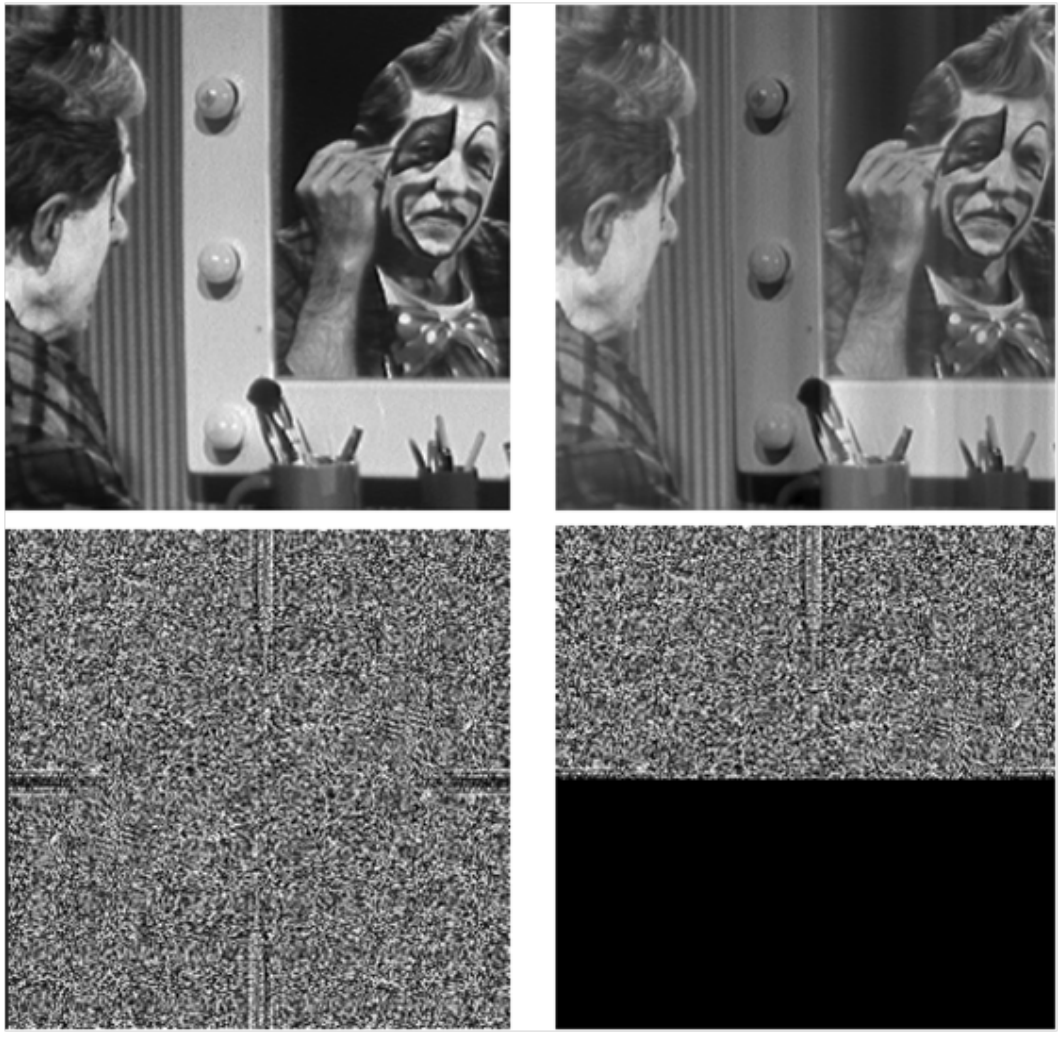

Figure 1.5: Full versus half k-space. [Moz08]

### 1.2.2 K-Space

For our understanding of k-space we need to explain the significance of the Fourier transform to Magnetic Resonance Imaging (MRI). The signals that we measure in MRI are a combination of signals from all over the object being imaged. Most signals are composed of a series of sine waves, each with an individual frequency and amplitude. The Fourier transform allows us to work out what these frequencies and amplitudes are. The Fourier transform converts the signal from the time domain into the frequency domain. For any image, use of the Fourier transform allows us to manipulate the data in the frequency domain also known as k-space [Ber69].

K-space can be defined as an array of numbers whose inverse Fourier transform is the magnetic resonance image. It is in k-space that the magnetic resonance signals are stored and the images are reconstructed from this data. According to MR theory [Nis96], [Bri74], [Ber69], we can derive the equation for magnetic resonance signals as

$$S(t) = \int_x \int_y m(x,y) e^{-2\pi[k_x(t)x + k_y(t)y]} dxdy \tag{1.5}$$

where

$$k_x(t) = \frac{\gamma}{2\pi} \int_0^t G_x(\tau) d\tau, \;\; k_y(t) = \frac{\gamma}{2\pi} \int_0^t G_y(\tau) d\tau \tag{1.6}$$

$m(x,y)$ is the transverse nuclear magnetization and $G_x$, $G_y$ are the gradient fields in the $x$ and $y$ directions respectively. Comparing the signal equation Eq. (1.5) with the Fourier transform of $m(x,y)$.

$$M(k_x, k_y) = \int_x \int_y m(x,y) e^{-i2\pi[k_x(t)x + k_y(t)y]} dxdy \tag{1.7}$$

we can see that

$$S(t) = M(k_x, k_y) \tag{1.8}$$

or

$$S(t) = M(\gamma/2\pi \int_0^t G_x(\tau) d\tau, \gamma/2\pi \int_0^t G_y(\tau) d\tau) \tag{1.9}$$

Thus, $k_x$ and $k_y$ are in units of spatial frequency, typically cycles/cm. This is the most important relationship in MRI. At any given time $t$, $S(t)$ equals the value of the two-dimensional Fourier transform of $m(x,y)$ at ($k_x$, $k_y$). The total recorded signal function $s(t)$ therefore maps directly to a trajectory through the spatial-frequency (Fourier transform) space as determined by the time integrals of the applied gradient waveforms $G_x$ and $G_y$.

Fig. 1.5 shows the difference between an image obtained from a full k-space and a half k-space data. K-space has this particular property whereby we can reconstruct an image using only partial k-space data. It is this property which we exploit in this thesis. Acquiring complete data (full k-space) is a long and computationally expensive process.As a result, it is not possible for critical patients or patients on life support to stay still for a long time in a magnetic resonance scanner. To get a very detailed and clear image the magnetic resonance machine tries to fully collect k-space data which is time consuming. Partial Fourier reconstruction is a technique that reduces the scanning time by skipping over some data and sampling certain rows of k-space [Moz08]. By reducing the amount of data to be collected, the scanning time can be drastically reduced. Hence, understanding efficient under-sampling and reconstruction of k-space data is very important in the context of this thesis.

## 1.3 Process Goal

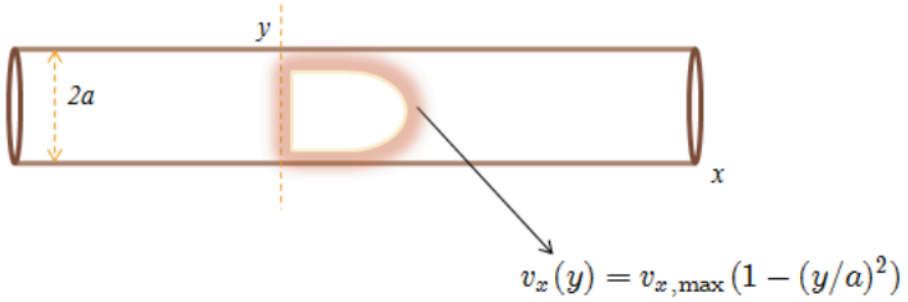


Figure 1.6: Laminar flow velocity phantom used in this thesis.

We present a model-based image reconstruction method that can be applied in phase-contrast angiography to detect anomalies in blood vessels. Since most images are distorted or contain significant amounts of noise, we try to eliminate errors using regularization. Regularization usually involves adding some sort of penalty to improve the smoothness and thereby the sharpness of the distorted images.

The main disadvantages of using phase-contrast angiography is the relatively long time that is required to produce images, and that phase-contrast images are degraded by aliasing artifacts [IL93]. Additionally this method

is plagued by long acquisition times and relatively long echo times. Hence shortening the time critically ill patients spend in the MRI machine is a major motivating factor to develop efficient image reconstruction techniques that considerably shortens scan times. Although advances like the development of new trajectories, parallel imaging and faster image acquisitions have reduced total imaging time for phase-contrast angiographic procedures, it is still relatively slow. However by using under-sampling and regularization of k-space, the scan time can be greatly reduced.

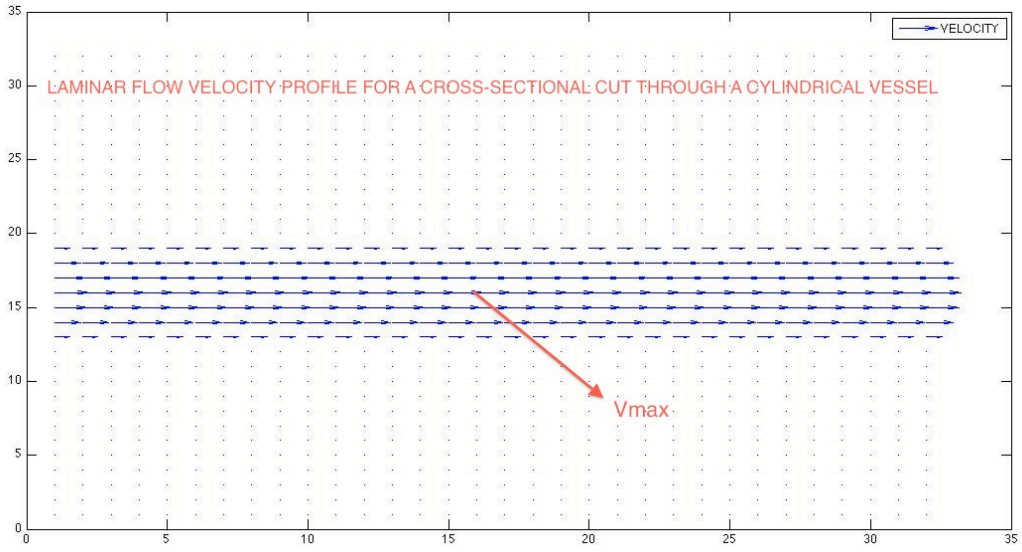


Figure 1.7: Depiction of flow as a continuous velocity field, with the maximum flow in the centre of the region of interest.

The numerical phantom which we use to test our model is a simulated laminar flow velocity profile of a blood flow in an artery as shown in Fig. 1.7. It is composed of complex data with maximum flow encountered at the centre of the vessel, while the velocity tapers to the minimum at the walls of the vessel as shown in Fig. 1.6. In order to show different phases and frequencies of magnetic resonance signals, all the acquired data are complex.

In this thesis we propose a two stage solution. First we reconstruct the problem in image space to test our model and then we exploit the problem in k-space to considerably reduce the amount of data we use. We use the $L_2$-difference regularization method to eliminate noise in our model, thereby increasing accuracy of the solution. Our goals in this thesis are outlined below:

1. Faster image reconstruction for shorter MRI scan times.
2. Larger images for better clarity.
3. Accurate images for better diagnosis.

In the next chapter we explain the necessary theory required to understand our optimization model and then present our model both in the image space and k-space domain.

# Chapter 2

# Optimization Model

In this chapter, we will present a method that solves the image reconstruction problem in image space. The motivation to solve the problem in image space is to use it as a prototype to solve the problem in k-space later on. We discuss how to model our problem as an inverse problem and how regularization is used to reduce artifacts and noise. The model we solve is an unconstrained non-linear optimization problem.

## 2.1 Inverse Problem

The inverse problem uses the actual observations of the system to infer the values of the parameters characterizing the system under investigation. To demonstrate how this is related to phase-contrast angiography, we take the example where the exact properties of the flow of blood are known. Then on imaging those blood vessels with contrast agents, the resultant flow map would be known. That would be the "forward problem". However, it is nearly always the properties of the flow of blood that we are trying to find, ideally without using invasive surgery or contrast agents. Thus we need to solve an inverse problem. Conceptually an inverse problem is described by the diagram below [Moz08].

$$\text{Actual observations} \rightarrow \text{ Parameters of the model} \tag{2.1}$$

The most common concern of the inverse problem is ill-posedness, which is opposite to being well-posed. A problem is said to be well-posed if its solution is unique and exits for the arbitrary and continuous data set, whereas for the ill-posed problem the solution is not unique. The unique solution computed is unacceptable in a physical sense because it is an approximate

solution [BB98], [Moz08]. The unique solution does not reproduce the exact solution, but has experimental errors and the original data is not completely reproduced. We can categorize a problem as a linear inverse problem, if the relationship between the actual observations and the model parameters are linear as shown in Eq. (2.2)

$$Ax = b \tag{2.2}$$

where $A$ is a linear operator, $b$ and $x$ represent data and model parameters respectively. The objective of the problem is to find $x$. We can use methods from optimization to solve such inverse problems. To do so, we define a goal, also known as an objective function, for the inverse problem. The goal is a function that measures how close the predicted data from the recovered model fits the observed data. In cases where we have perfect data (i.e. no noise) and perfect physical understanding (i.e. the physics) then the recovered model should fit the observed data perfectly. The standard objective function term is usually of the form as shown in Eq. (2.3)

$$\|Ax - b\|^2 \tag{2.3}$$

In relation to Magnetic Resonance Imaging (MRI), we can define an optimization problem in which the objective function measures the likelihood of observing image pixel values given the underlying tissue concentrations. In other words, we minimize the distance between the measured Magnetic Resonance (MR) experimental images and the images predicted by the forward problem.

## 2.2 Regularization

Regularization and optimization of an inverse problem has been an active area of research with a long history. An example of such problem might be the formation of an original image captured using an imaging device (original data) [Aga03]. Regularization can be used to make an ill-posed problem well-posed by introducing additional information about the solution, such as an assumption about the smoothness or a bound on the norm. The problem is then changed to a new problem with improved conditioning. Inverse problems are not usually well-posed, and in order to be solvable a variety of regularization techniques can be applied. Most often the solution for ill-posed problems can be expected to be useless, therefore a method to compute the approximate solutions that are less sensitive to perturbations is needed. We call this method regularization because it enforces regularity (smoothness) on

the computed solutions. We suppress unwanted noise components by enforcing this regularity, or smoothness which leads to better approximations of the solution. The Eq. (2.4) shows Tikhonov's method [TA77] where the regularity requirement is explicitly incorporated in the formulation of the problem.

$$\min\{\|Ax - b\|^2 + \lambda \|x\|^2\} \tag{2.4}$$

The first term $\|Ax - b\|^2$ measures the fit-to-data and the second term $\|x\|^2$ measures the regularity of the solution. The regularization parameter ($\lambda$) is a positive parameter which controls the weighting between the two terms. It indicates that high frequencies in the solution are penalized and converge to zero, but low frequencies are not regularized. Thus the regularization parameter ($\lambda$) behaves as a low-pass filter. In order to obtain a balance or minimize the trade-off, the optimal selection of the regularization parameter ($\lambda$) becomes important. Various methods [Idi99], [BPT88] have been developed for the optimal selection of these regularization parameters.

- Discrepancy principle
- L-curve
- Generalized cross validation method

In recent years, the L-curve despite its limitations has gained attention for computing the selection of regularization parameters. Its a log-log plot of the regularized solution against the squared norm of the regularized residual for a range of values of regularization parameters. The numerical computation and limitation of the L-curve is explained in [Han10] and [Vog96]. Regularization is implemented on both the rows and columns of the two-dimensional image we use in this thesis. The steps carried out for regularization are outlined below, where A is one of the variables ($\rho$ and $V$) and $n$ is the size of the image.

**for** $i = 1 \rightarrow n$ **do**
  **for** $j = 1 \rightarrow n - 1$ **do**
    rowdifference $\leftarrow$ rowdifference $+(A_{i,j+1} - A_{i,j})^2$
  **end for**
**end for**
**for** $i = 1 \rightarrow n - 1$ **do**
  **for** $j = 1 \rightarrow n$ **do**
    columndifference $\leftarrow$ columndifference $+(A_{i+1,j} - A_{i,j})^2$
  **end for**
**end for**

### 2.2.1 Model in Image Space

In analogy with the definition of a linear inverse problem in Eq. (2.2) we describe the (nonlinear) fit-to-data term in image space, where $\widetilde{m}$ is the experimental data acquired from the Magnetic Resonance Image (MRI) scanners.

$$\sum_{G} \left\| I_{G_k} \circ (\rho_{i,j} \odot e^{-iG_k \cdot V_{i,j}}) - \widetilde{m}_{k,i,j} \right\|^2 \tag{2.5}$$

The model in image space is used to test our theory before we eventually solve the problem in k-space. The fit-to-data term for the model in image space is described in Eq. (2.5). As mentioned before in Eq. (1.4), the operator ($\odot$) in Eq. (2.5) represents point-wise multiplication. Since the model in k-space requires projections to test the effects of minimal data (partial k-space), we introduce the identity matrix $I_G$ as a place-holder for the future application of projections. Here $G$ determines the gradient vectors used to sensitize the blood flow components along multiple directions. We use two parameters ($\rho$ and $V$) to determine the computed solution of our optimization problem. $\rho$ is used to determine the proton density, which is a measure of the amount of hydrogen nuclei present in water. We solve for both the real and imaginary parts of $\rho$. Here, $V$ is the encoded velocity of the blood flow in the artery or a blood vessel in the $x$ and $y$ directions.

$$\begin{aligned} f_{obj} = & \left\| (\rho_{i,j} \odot e^{-iG_1 \cdot V_{i,j}}) - \widetilde{m}_{1,i,j}) \right\|^2 \\ & + \left\| (\rho_{i,j} \odot e^{-iG_2 \cdot V_{i,j}}) - \widetilde{m}_{2,i,j}) \right\|^2 \\ & + \left\| (\rho_{i,j} \odot e^{-iG_3 \cdot V_{i,j}}) - \widetilde{m}_{3,i,j}) \right\|^2 \\ & + \left\| (\rho_{i,j} \odot e^{-iG_4 \cdot V_{i,j}}) - \widetilde{m}_{4,i,j}) \right\|^2 \end{aligned} \tag{2.6}$$

We are solving a problem of size $4n^2$ as shown in Eq. (2.6) where $G_1, G_2, G_3$ and $G_4$ represent the directions along (0,1), (1,1), (1,-1),(1,0) respectively. Since we are solving both real and imaginary parts of $\rho$, we simplify Eq. (2.5), which is described below as Eq. (2.7).

$$\begin{aligned} & \{[(\mathrm{Re}(\rho_{i,j}) \cdot \cos(G_k \cdot V_{x,i,j} + G_k \cdot V_{y,i,j})) \\ & + (\mathrm{Im}(\rho_{i,j}) \cdot \sin(G_k \cdot V_{x,i,j} + G_k \cdot V_{y,i,j})) - \mathrm{Re}(\widetilde{m}_{k,i,j})]\}^2 \\ & + \{[(\mathrm{Im}(\rho_{i,j}) \cdot \cos(G_k \cdot V_{x,i,j} + G_k \cdot V_{y,i,j})) \\ & - (\mathrm{Re}(\rho_{(i,,j)}) \cdot \sin(G_k \cdot V_{x,i,j} + G_k \cdot V_{y,i,j})) - \mathrm{Im}(\widetilde{m}_{k,i,j})]\}^2 \end{aligned} \tag{2.7}$$

We introduce the regularization parameters ($\lambda_1$ and $\lambda_2$) to $\rho$ and $V$ to Eq. (2.5) to define our objective function for the model in image space which is described in Eq. (2.8)

$$\begin{aligned}\min \sum_{G} \left\| I_{G_k} \circ (\rho_{i,j} \odot e^{-iG_k \cdot V_{i,j}}) - \widetilde{m}_{k,i,j} \right\|^2 \\ +\lambda_1 \sum_{i,j} \left\| \delta\rho_{i,j} \right\|^2 \\ +\lambda_2 \sum_{i,j} \left\| \delta V_{i,j} \right\|^2\end{aligned} \tag{2.8}$$

### 2.2.2 Model in K-space

According to MRI theory [Pau07], most Magnetic Resonance (MR) images depict the spin density as a function of position, and hence should be real-valued. If this were true, then by the symmetry of the Fourier transform, only half of the spatial frequency data would need to be collected. Since real functions have conjugate symmetry in spatial frequency space, the uncollected data could be synthesized by reflecting conjugate data across the origin. Unfortunately, there are many sources of phase errors that cause the real-valued assumption to be violated. These include variations in the resonance frequency, flow, and motion. One of the general applications of two-dimensional Fourier transform imaging is reducing scan time. Conventionally more than half of the complete k-space data is collected, allowing the scan time to be reduced by almost a factor of two [Pau07]. In this thesis, however, we collect only five-percent of k-space data.

#### Projection

The phantom that we are going to use for our model in k-space is completely sampled in k-space. We can simulate the partial k-space data by projecting the full k-space data onto a subspace.

$$\text{projection( full k-space)} = \text{partial k-space} \tag{2.9}$$

The simplest way to reconstruct a partial k-space data set is to simply fill the uncollected data (phase-encodes or readout samples) with zeroes. Then, we perform the two-dimensional Fourier transform and display the magnitude. Thus, in our model the projection is described below in Eq. (2.10)

$$\pi_G \circ \mathcal{F}^{-1}(\rho_{i,j} \odot e^{-iG_k \cdot V_{i,,j}}) \tag{2.10}$$

The projection in k-space will be different for each velocity sensitization. This approach enables us to choose a variety of projection matrices for each gradient

sensitization vectors. In our case we take the horizontal, leading diagonal, secondary diagonal and vertical samples to effectively sensitize the velocity/flow of the blood. Let

$$F(\rho, V) = \left\| \pi_{G_k} \circ \mathcal{F}^{-1}(\rho_{i,j} \odot e^{-i \cdot G_k \cdot V_{i,,j}}) - m_{k,i,j} \right\|^2 .$$

In order to use the chain rule, we define it as a composition of $f$ and $g$ as follows,

$$\begin{aligned}
u_{i,j,k} &= g_k(\rho, V) = \rho_{i,j} \odot e^{-iG_k \cdot V_{i,,j}} \\
&= [(\mathrm{Re}(\rho_{i,j}) \cdot \cos(G_{\mathrm{x},k} \cdot V_{\mathrm{x},i,j} + G_{\mathrm{y},k} \cdot V_{\mathrm{y},i,j)})) \\
&\quad +(\mathrm{Im}(\rho_{i,j}) \cdot \sin(G_{\mathrm{x},k} \cdot V_{\mathrm{x},i,j} + G_{\mathrm{y},k} \cdot V_{\mathrm{y},i,j)})) \\
&\quad - \mathrm{Re}(\tilde{m_{k,i,j}})] \\
&\quad +i[(\mathrm{Im}(\rho_{i,j}) \cdot \cos(G_{\mathrm{x},k} \cdot V_{\mathrm{x},i,j} + G_{\mathrm{y},k} \cdot V_{\mathrm{y},i,j)})) \\
&\quad -(\mathrm{Re}(\rho_{i,j}) \cdot \sin(G_{\mathrm{x},k} \cdot V_{\mathrm{x},i,j} + G_{\mathrm{y},k} \cdot V_{\mathrm{y},i,j)})) - \mathrm{Im}(\tilde{m_{k,i,j}})] \\
f(u_{i,j,k}) &= \left\| \pi_G \circ \mathcal{F}^{-1}(u) - m_{k,i,j} \right\|^2 \\
F(\rho, v) &= f(g(\rho, v)) \\
\nabla F(\rho, v) &= \nabla f(u_{_{i,j,k}}) \nabla g(\rho, v)
\end{aligned}$$

The gradient of the model in k-space is described in the Eq. (2.11) below:

$$\nabla F(\rho, V) = \quad (2\mathcal{F}(\pi^*(\pi(\mathcal{F}^{-1}(u)) - m)))(\nabla g(\rho, v)) \tag{2.11}$$

Minimizing the term shown in Eq. (2.10) defines an approximate solution, $\rho$ and $V$ to the expression $\pi_G \circ \mathcal{F}^{-1}(\rho \odot e^{-iG \cdot V}) = m$. However, if $\pi_G \circ \mathcal{F}^{-1}(\rho \odot e^{-iG \cdot V})$ is singular or ill-conditioned, and $m$ is the result of measurement contaminated by noise, the estimate may be inaccurate and there may be a large number of solutions. We introduce the two-dimensional Fourier transform and the regularization parameters to construct our objective function for the model in k-space as shown in Eq. (2.12)

$$\begin{aligned}
\min \sum_G \left\| \pi_{G_k} \circ \mathcal{F}^{-1}(\rho_{i,j} \odot e^{-iG_k \cdot V_{i,j}}) - m_{i,j}) \right\|^2 \\
+\lambda_1 \sum_{i,j} \left\| \delta \rho_{i,j} \right\|^2 \\
+\lambda_2 \sum_{i,j} \left\| \delta V_{i,j} \right\|^2
\end{aligned} \tag{2.12}$$

**Justification for sparse sampling**

Having defined the projections we will use, it is clear that (1) we are sampling a very small subset of the full data set, and (2) there is a pattern to the under-sampling. Under-sampling is always justified if a priori knowledge, such as the expectation of smoothness, is available. In this case, we do expect smooth images, but we also know that the noise-free images have a great deal of redundancy. All static tissue, which will generally be most of the tissue, will be identical in images with all sensitizations. Moving tissue will have velocity information encoded as phase, but the magnitude will always be the same. So depending on the relative blood volume, we can expect upwards of three quarters of the data to be redundant. Furthermore, MRI images have large constant regions, which causes the information in k-space to be concentrated near k=0.

In our projection or subsampling scheme, we collect the low k-space data which contains most of the energy in each sensitization, which can be interpreted as a low-pass filtering of the data in k-space. We do not collect all of the high-frequency data, but when we consider all of the sensitizations, we collect bands in eight principle directions. This results in some coverage of all parts of k-space, with complete coverage of the lowest frequencies. In some sense, this is analogous to the situation in PROPELLER [Pip99], whose sampling pattern is similar to the sampling pattern we would get if we replaced all of the gradient sensitization vectors with zero.

**Sparse structure of the gradient in K-space**

The gradient $\nabla F(\rho, V)$ represented in Eq. (2.11) consists of two components which are $\nabla f(u_{_{i,j,k}})$ and $\nabla g(\rho, v)$. $\nabla g(\rho, V)$ represents the derivative of the image in image space and consists of a total of $8n^4k$ elements where $n$ is the size of the image and $k$ is the total number of gradient sensitization vectors. The sparseness of the gradient can be attributed to the derivative of $g(\rho, V)$ with respect to a particular variable which will produce $2n^2k$ *non-zero* elements taking into account both the real and imaginary parts of $g(\rho, V)$. The structure of sparsity resembles a block diagonal structure and the values of the diagonal representing the *non-zero* elements. This is because each pixel in $\nabla g(\rho, V)$ is composed of all the four variables $(\mathrm{Re}(\rho), \mathrm{Im}(\rho), V_{\mathrm{x}}, V_{\mathrm{y}})$ given by the equation. The other component $\nabla f(u_{_{i,j,k}})$ is a matrix containing the intermediate sensitization variables in image space. The output of the gradient $\nabla F(\rho, V)$ can be visualized as the multiplication of two eight block components, the output of which will produce $4n^2$ elements.

## 2.3 Implementation

We implement our model in MATLAB using various algorithms available in the optimization toolbox. We use the function *fminunc* to find the minimum of an unconstrained multivariable function. The function *fminunc* attempts to find a minimum of a scalar function of several variables, starting at an initial estimate. The general description of the function signature of *fminunc* is shown to be f[x fval]=fminunc('objFun',A,options), where *fval* is the value of the objective function *objFun* at the solution $x$. Here the objFun is a MATLAB function of the form function [f g] = objFunK(A), where $f$ is the function value at $x$. The gradient of *objFun* can also be computed by setting the *GradObj* option as "on", by setting options = optimset('GradObj','on'). The gradient is the partial derivatives $\partial f/\partial x_i$ of $f$ at the point $x$. That is, the $i$th component of $g$ is the partial derivative of f with respect to the $i$th component of $x$.

We generate the numerical phantom which is represented by $m_{i,j}$ in Eq. (2.12). To generate noise in our phantom we add white Gaussian noise specifying a scalar signal-to-noise ratio (SNR) per sample in decibels. The model was implemented with various sets of signal-to-noise ratio values. In our model, the value of the lowest signal-to-noise ratio used was twenty-five. The Signal-to-noise ratio is a measure to quantify how much a signal has been corrupted by noise.We implement the model in image space first by calculating the gradient and passing it onto the *fminunc* function in Matlab. We implement the same procedure for our model in k-space with changes to the objective function and the corresponding gradient. We then apply a Fast Fourier Transform (FFT) to the complex data and apply the various projections necessary to optimize our model in k-space. We set the penalty parameters as $\lambda_1$ and $\lambda_2$. The penalty parameters were chosen using the L-curve technique, which was covered in previous sections.

# Chapter 3

# Numerical Results

In this chapter we show the results obtained by optimizing the model in both image space and k-space. The Fig. 3.1 shows the noisy measurement and the regularized image of the real part of the proton density and the Fig. 3.2 shows the imaginary part of the proton density. The images shown in Fig. 3.3 is the resultant velocities of the blood flow in $x$ and $y$ directions in image space. We

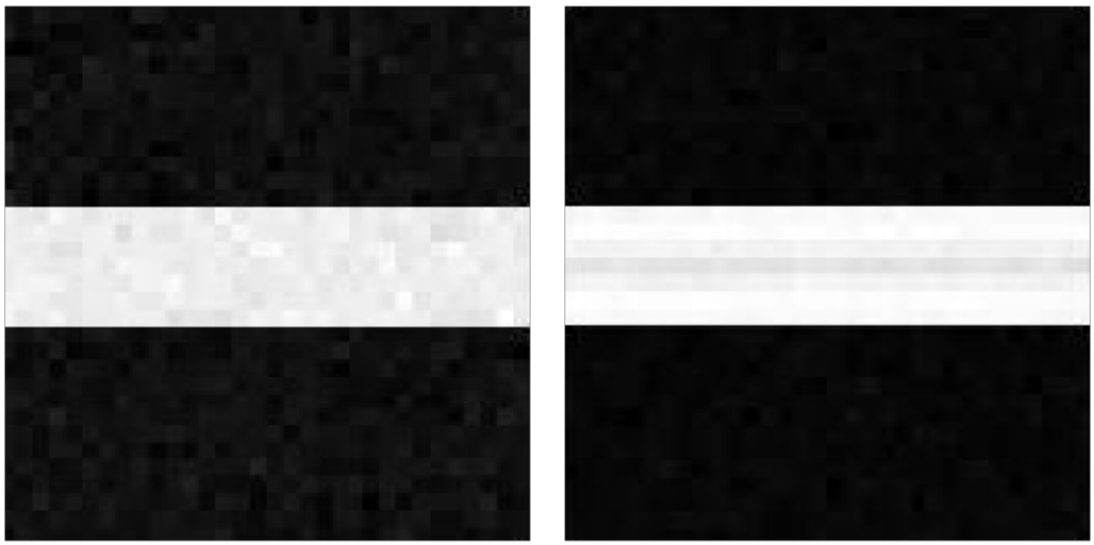

Figure 3.1: Real part of a measured image, showing noise, and the real part of the reconstructed $\rho$, the proton density.

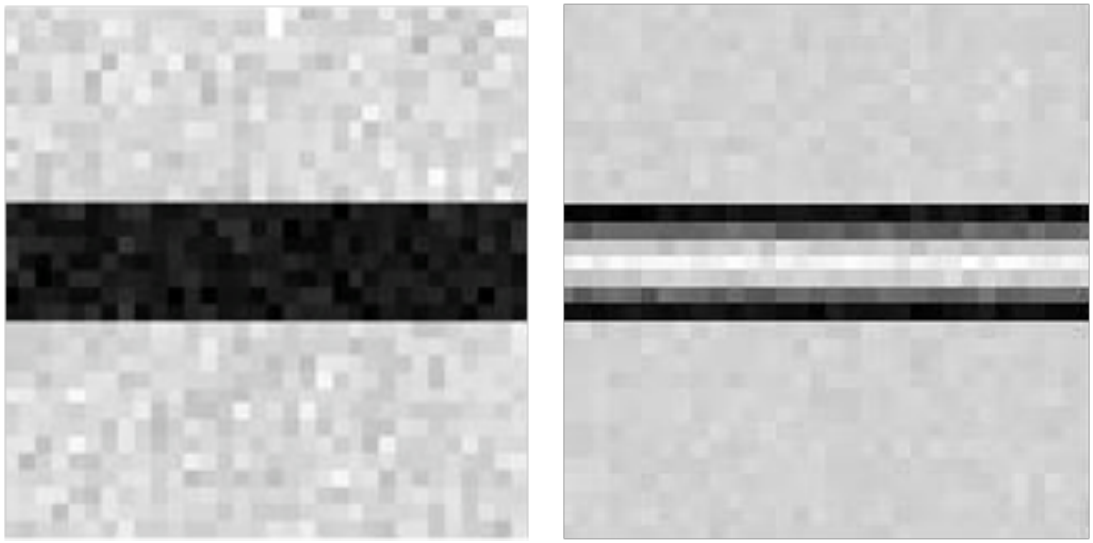

Figure 3.2: Imaginary part of a measured image, showing noise, and the imaginary part of the reconstructed $\rho$, the proton density.

see that that regularization has eliminated noise and the images look much sharper and smoother.

The Figure's 3.4, 3.5 and 3.6 are the results obtained by optimizing in k-space. The images show blurring which is an imaging artifact usually encountered, when k-space is under-sampled. According to MR theory [Pau07], [HNO06], when k-space is under-sampled, the Nyquist criterion is violated, and Fourier reconstructions exhibit aliasing artifacts. The reason for the blurring can be identified by considering the data set to be the product of a full k-space data set multiplied by a projection function which we denote as $\pi_G$ as shown in Eq. (2.10), where G is the velocity sensitization vectors. The inverse Fourier transform of this function is the impulse response that produces the blurring.

Table. 3.1 shows the execution time of the image optimized in image space. The number of variables we solve in this optimization is of the size of

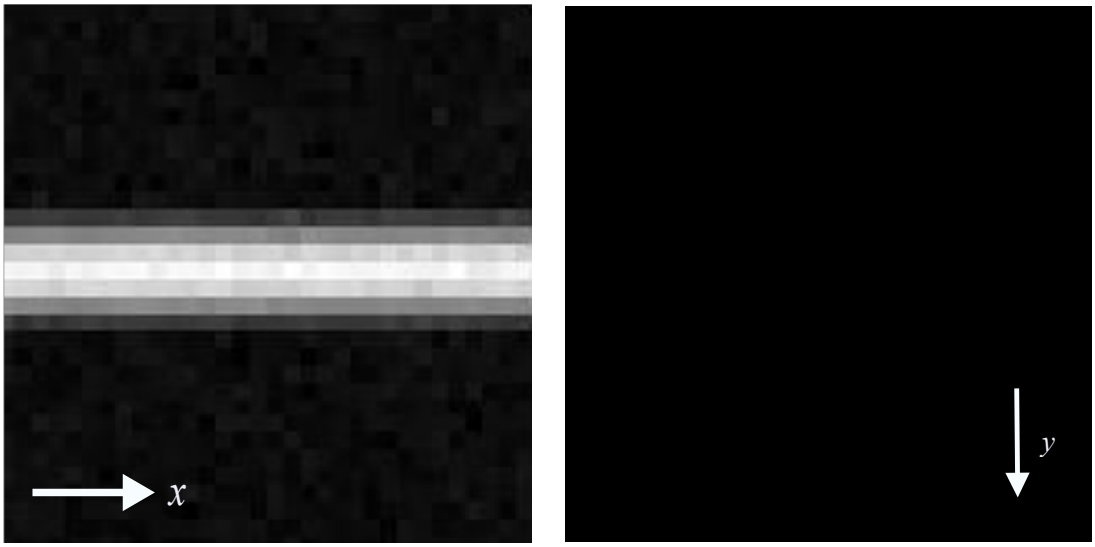


Figure 3.3: Velocity components in the $x$ and $y$ directions.

Table 3.1: Execution time in Image space

| Image size | Time in seconds |
|---|---|
| [8,8] | 0.8966 |
| [16,16] | 13.5156 |
| [32,32] | 142.7051 |
| [64,64] | Out of memory |

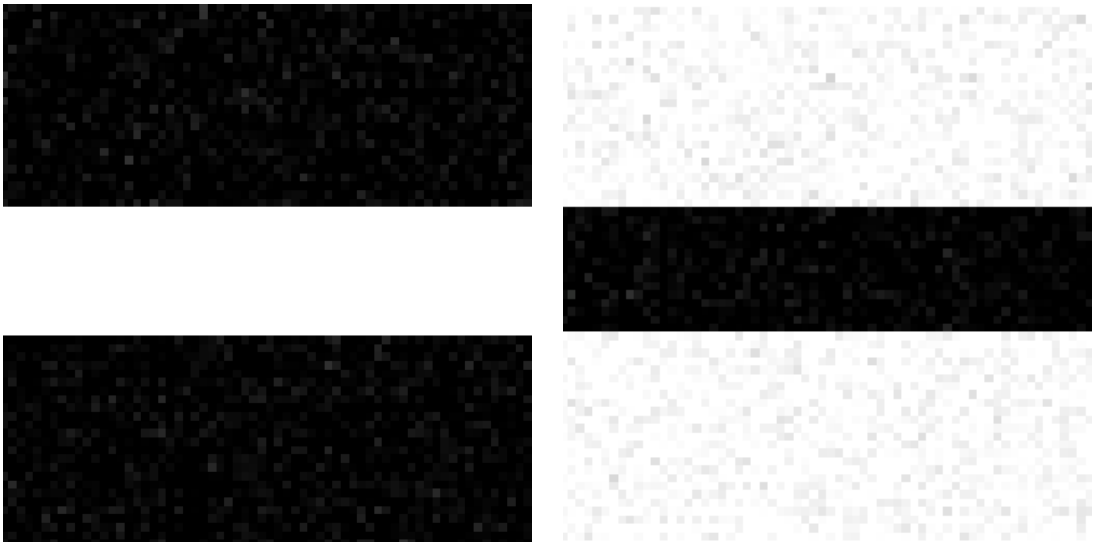

Figure 3.4: Image reconstructed from partial k-space with (1,0) sensitization, real and imaginary parts.

$4n^2$, and as we increase the size of the image, the time taken and the memory constrains on the machine also increases. The results from Table. 3.1 shows that for optimizing an image of size $64 \times 64$, MATLAB runs out of memory. This is because the amount of memory required to store a Hessian ($N^2$) is too big, along with the machine time required to process it. Hence there is a need to try and reduce the data set using partial k-space. Table. 3.2 shows considerable improvement of execution time of the image optimized in k-space. Not only does the partial k-space allow us to optimize larger images ($64 \times 64$) it takes relatively less amount of time as compared to optimizing in image space.

We generate two L-curves, where in Fig. 3.7 $\lambda_2$ is set to 1e-04 and the results are generated for different $\lambda_1$ values. In Fig. 3.9 $\lambda_1$ is set to 1e-04 and the results are generated for different $\lambda_2$ values. As we mentioned before in the previous sections $\lambda_1$ and $\lambda_2$ are the regularization parameters for the variables $\rho$ and $V$ in our model. What we know from regularization is that it improves the results by reducing the artifacts and background noise. In order

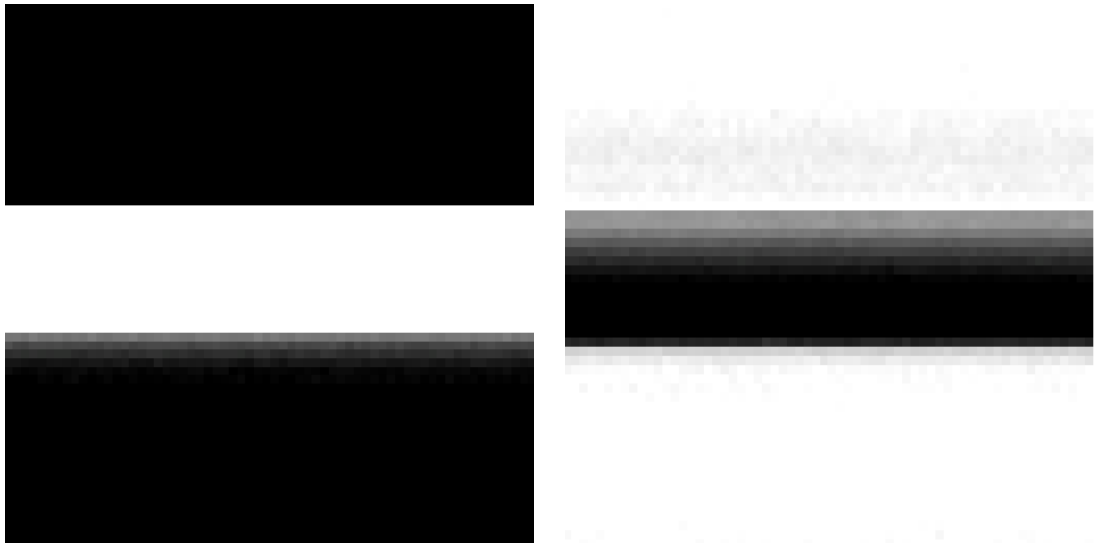

Figure 3.5: Proton density estimate from multiple partial k-space data sets; real and imaginary parts.

Table 3.2: Execution time in K-space

| Image size | Time in seconds |
|---|---|
| [8,8] | 0.1791 |
| [16,16] | 1.4682 |
| [32,32] | 12.7797 |
| [64,64] | 170.1214 |

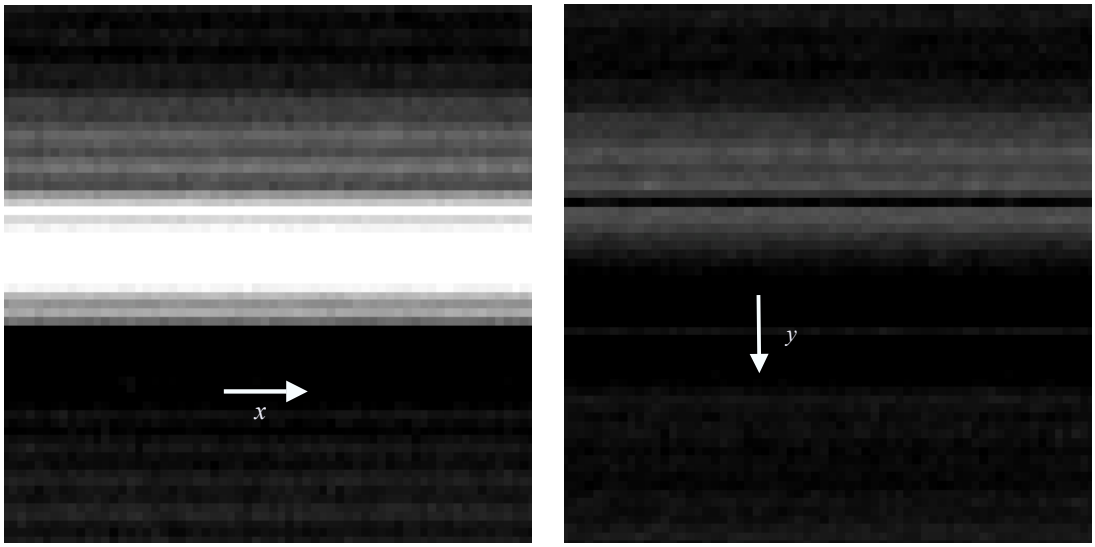


Figure 3.6: Velocity (x- and y-components) estimated, using regularization, from partial k-space data sets.

to select the best regularization parameters $\lambda_1$ and $\lambda_2$, we use the L-curve to find a good estimate. We can apply different weightings to regularization parameters within a certain range of data. If $\lambda_1$ or $\lambda_2$ is very big, we will get over-smoothed images, and if it is very small, the results will look under-smoothed for the corresponding variables. An over-smoothed image reduces the noise, but does not show the details such as edges and contrasts. By using the L-curve technique, we can find a good trade-off for $\lambda_1$ which is located at the very left of the curvature shown in Fig. 3.7, where $\lambda_1 = 1e^{-05}$ and for $\lambda_2$ which is also located at the very left of the curvature shown in Fig. 3.9, where $\lambda_2 = 1e^{-04}$.

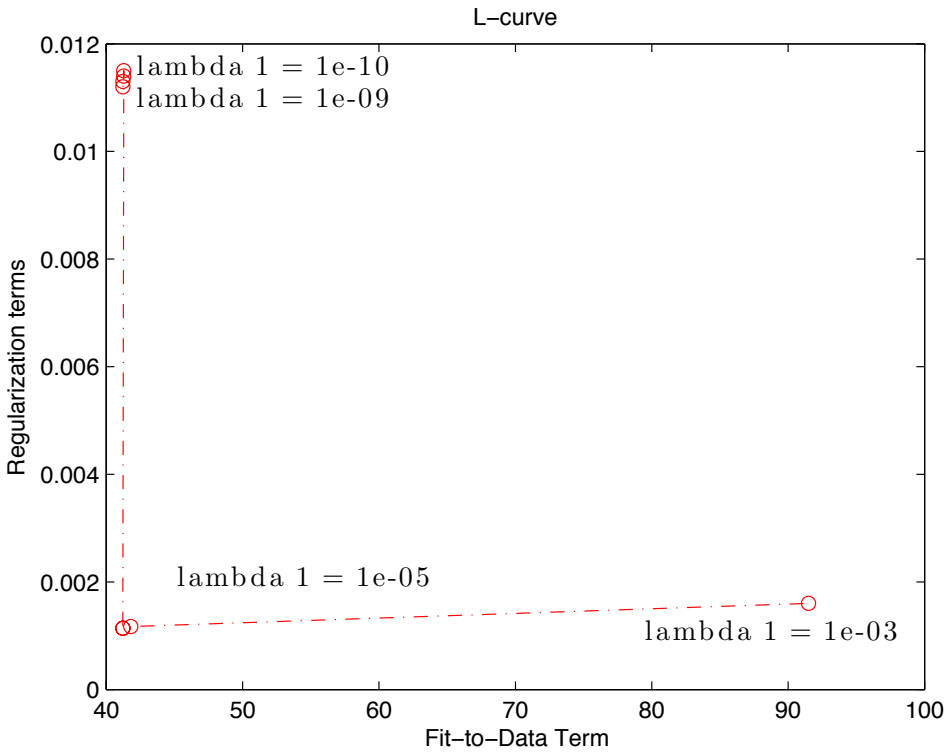


Figure 3.7: L-curve showing the variation in the fit-to-data term versus $\|\delta\rho\|^2$ as $\lambda_1$ varies with $\lambda_2 = 1e^{-4}$ held constant.

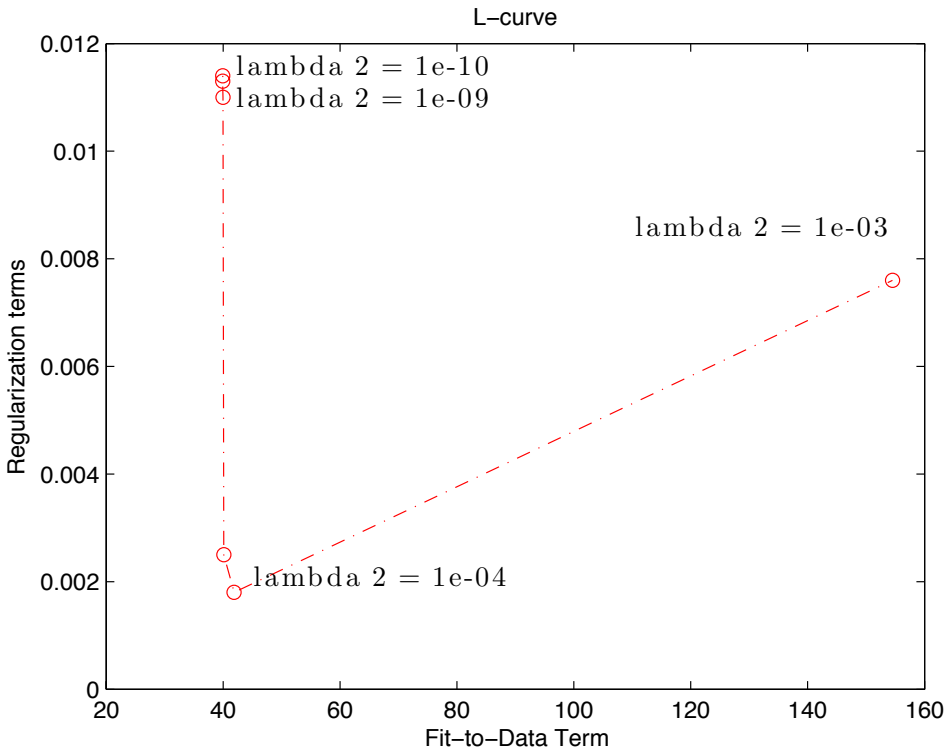


Figure 3.8: L-curve showing the variation in the fit-to-data term versus $\|\delta V\|^2$ as $\lambda_2$ varies with $\lambda_1 = 1e^{-4}$ held constant.

Figure 3.9: L-curve where $\lambda_1 = 1e^{-04}$

# Chapter 4

# Conclusion and Future Work

In conclusion we have demonstrated in this thesis a model-based image reconstruction method that reduces the image reconstruction time in phase-contrast angiography. We demonstrated experimental verification of our model in image space and in k-space. We showed that by under-sampling, we were able to use reduced data to increase the performance of image reconstruction. We demonstrated that using only five-percent of the data we were able to achieve significant benefits in reducing the time taken to optimize large images. We used regularization to reduce noise and increase the accuracy of the image. Due to the reduced data set, we were able to solve the memory problem encountered while optimizing in image space.

This model can be improved in the future by incorporating a three-dimensional model which would provide us with more clinical insights of blood flow. A more complicated phantom can also be used to test our model before we can use it on clinical data. Significant improvement also can be acquired by implementing the model in C and using the L-BFGS algorithm for the optimization. L-BFGS stands for "Limited memory BFGS". In case of big dimensions, the amount of memory required to store a Hessian ($N^2$) is too big, along with the machine time required to process it, hence the need to use L-BFGS . L-BFGS stores only a few vectors that represent the approximation of the dense ($N^2$) matrix implicitly. Due to its moderate memory requirement, L-BFGS method is particularly well suited for optimization problems with a large number of variables. Table. 4.1 shows the preliminary work carried out optimizing in image space using L-BFGS.

Table 4.1: Execution time for image-space model using L-BFGS calling function and gradient written in C.

| Image size | Time in seconds |
|---|---|
| [64,64] | 5.1285 |
| [128,128] | 16.564 |

# Chapter 5

# Appendix A: Matlab Codes

### A1: Objective function in K-space

```
1
2  function [f g] = objFunK(A)
3  % n     - The size of the Image
4  % R     - Radius of the vessel
5  % V_max - Maximum velocity of blood
6  % p     - Proton density matrix
7  % v_x   -  Matrix containing x components of Velocity vector
8  % v_y   -  Matrix containing y components of Velocity vector
9  % (0,1,-1) - Gradient Sensitization vectors
10
11
12 %-------------------------------------------
13 %-----------------  Initialization ----
14 %-------------------------------------------
15
16
17 n=64;
18
19
20
21 lamda_1=1e-6; %------ Penalty for proton density
22
23 lamda_2=1e-2; %------ Penalty for Velocity
24
25 snr=25; % -------- Signal-to-noise ratio
26
27
28 [c,d] = size(A);
29
```

```
30 p_real   = A(1:c/4,:);
31 p_imag =A((c/4)+1:(c/4)+d,:);
32 v_x = A(((c/4)+d)+1:(c/4)+2*d,:);
33 v_y = A(((c/4)+2*d)+1:c,:);
34
35
36
37
38 thetaG1 = (0 .* v_x) + (1 .* v_y);
39
40 thetaG2 = (1 .* v_x) + (1 .* v_y);
41
42 thetaG3 = (1 .* v_x) + (-1 .* v_y);
43
44 thetaG4 = (1 .* v_x) + (0 .* v_y);
45
46
47 R=n/8;
48 v_max=1;
49 v_y_m=zeros(n,n);
50 p_m= zeros(n,n);
51
52
53
54 proj1_temp =  fft2( (((p_real .* cos(thetaG1) )
55                       + (p_imag .* sin(thetaG1) ))
56               + (1i*((p_imag .* cos(thetaG1) )
57                - (p_real .* sin(thetaG1 ) )))),n,n);
58
59
60 proj2_temp =  fft2( (((p_real .* cos(thetaG2) )
61               + (p_imag .* sin(thetaG2) ))
62                + (1i*((p_imag .* cos(thetaG2) )
63                - (p_real .* sin(thetaG2 ) )))),n,n );
64
65
66 proj3_temp =  fft2( (((p_real .* cos(thetaG3) )
67                         + (p_imag .* sin(thetaG3) ))
68                         + (1i*((p_imag .* cos(thetaG3) )
69                          - (p_real .* sin(thetaG3 ) )))),n,n);
70
71
72 proj4_temp =  fft2( (((p_real .* cos(thetaG4) )
73                          + (p_imag .* sin(thetaG4) ))
74                          + (1i*((p_imag .* cos(thetaG4) )
75                          - (p_real .* sin(thetaG4 ) )))),n,n);
76
77
```

```
78
79
80
81  proj1=zeros(n,n);
82
83
84  proj4=zeros(n,n);
85
86
87
88  %Vertical sample
89
90  proj1(:,1:(n/4)+1)=proj1_temp(:,1:(n/4)+1);
91
92  %Diagonal1 sample
93  proj2 =  diag(diag(proj2_temp))
94               +diag(diag(proj2_temp,1),1)
95               +diag(diag(proj2_temp,-1),-1)
96               +diag(diag(proj2_temp,2),2)
97               +diag(diag(proj2_temp,-2),-2);
98
99  %Diagonal2 sample
100 proj3=fliplr(diag(diag(fliplr(proj3_temp)))
101 +diag(diag(fliplr(proj3_temp),1),1)
102 +diag(diag(fliplr(proj3_temp),-1),-1)
103 +diag(diag(fliplr(proj3_temp),2),2)
104 +diag(diag(fliplr(proj3_temp),-2),-2));
105
106 %Horizontal Sample
107 proj4(1:(n/4)+1,:)=proj4_temp(1:(n/4)+1,:);
108
109
110 %-------------------------------------------------------------
111 %------------------ Measurement ------------------------------
112 %-------------------------------------------------------------
113 for l = 1:n
114 for m = 1:n
115
116
117         if (distance([l,m], [n/2,m]) ≥ R)
118             p_m(l,m) = 1i;
119             v_x_m(l,m) = 0;
120
121         else
122
123             p_m(l,m) =  2;
124             v_x_m(l,m) = v_max*(1-((distance([l,m], ...
                    [n/2,m]))^2/R^2));
```

```
125
126
127             end
128
129     end
130
131     end
132
133
134
135     p_m_real=awgn(real(p_m),snr,'measured');
136
137     p_m_imag=awgn(imag(p_m),snr,'measured');
138
139
140     v_x_m_ur=awgn(real(v_x_m),snr,'measured');
141
142     v_y_m_ur=awgn(real(v_y_m),snr,'measured');
143
144
145
146     m1_temp=awgn(p_m.*(exp(-1i*(0 .* v_x_m + 1 .* ...
            v_y_m))),snr,'measured');
147     m2_temp=awgn(p_m.*(exp(-1i*(1 .* v_x_m + 1 .* ...
            v_y_m))),snr,'measured');
148     m3_temp=awgn(p_m.*(exp(-1i*(1 .* v_x_m + -1 .* ...
            v_y_m))),snr,'measured');
149     m4_temp=awgn(p_m.*(exp(-1i*(1 .* v_x_m + 0 .* ...
            v_y_m))),snr,'measured');
150
151
152
153
154
155     ftm1=fft2(m1_temp,n,n);
156     ftm2=fft2(m2_temp,n,n);
157     ftm3=fft2(m3_temp,n,n);
158     ftm4=fft2(m4_temp,n,n);
159
160
161
162     m1=zeros(n,n);
163     m4=zeros(n,n);
164
165
166
167     %Vertical sample
168
```

```
169 m1(:,1:(n/4)+1)=ftm1(:,1:(n/4)+1);
170
171 %Diagonal1 sample
172 m2=diag(diag(ftm2))
173 +diag(diag(ftm2,1),1)
174 +diag(diag(ftm2,-1),-1)
175 +diag(diag(ftm2,2),2)
176 +diag(diag(ftm2,-2),-2);
177
178 %Diagonal2 sample
179 m3=fliplr(diag(diag(fliplr(ftm3)))+diag(diag(fliplr(ftm3),1),1)
180 +diag(diag(fliplr(ftm3),-1),-1)
181 +diag(diag(fliplr(ftm3),2),2)
182 +diag(diag(fliplr(ftm3),-2),-2));
183
184 %Horizontal Sample
185 m4(1:(n/4)+1,:)=ftm4(1:(n/4)+1,:);
186
187
188 %-------------------------------------------------
189 %------------------  Fit to Data ----------
190 %-------------------------------------------------
191
192
193
194
195
196
197
198         real_G1 = real(proj1) - real(m1);
199         imag_G1 = imag(proj1) - imag(m1);
200
201         real_G2 = real(proj2) - real(m2);
202         imag_G2 = imag(proj2)- imag(m2);
203
204         real_G3 = real(proj3) - real(m3);
205         imag_G3 = imag(proj3) - imag(m3);
206
207         real_G4 = real(proj4)- real(m4);
208         imag_G4 = imag(proj4) - imag(m4);
209
210
211
212         f1 = (real_G1 .* real_G1) + (imag_G1 .* imag_G1);
213
214         f2 = (real_G2 .* real_G2) + (imag_G2  .* imag_G2);
215
216         f3 = (real_G3 .* real_G3) + (imag_G3  .* imag_G3);
```

```
217
218         f4 = (real_G4 .* real_G4) + (imag_G4 .* imag_G4);
219
220
221
222         f_obj=reshape((f1+f2+f3+f4),1,[]);
223
224
225 %-------------------------------------------------------
226 %------------------  Gradient ------------
227 %-------------------------------------------------
228
229
230
231
232
233
234
235 % Gradient w.r.t real(p) along directions G1,G2,G3,G4
236
237
238             G1_real_p=cos(thetaG1) - (1i*(sin(thetaG1)));
239             G2_real_p=cos(thetaG2) - (1i*(sin(thetaG2)));
240             G3_real_p=cos(thetaG3) - (1i*(sin(thetaG3)));
241             G4_real_p=cos(thetaG4) - (1i*(sin(thetaG4)));
242
243
244
245  % Gradient w.r.t imag(p) along directions G1,G2,G3,G4
246
247
248
249             G1_imag_p=sin(thetaG1) + 1i*(cos(thetaG1 ));
250             G2_imag_p=sin(thetaG2) + 1i*(cos(thetaG2 ));
251             G3_imag_p=sin(thetaG3) + 1i*(cos(thetaG3 ));
252             G4_imag_p=sin(thetaG4) + 1i*(cos(thetaG4 ));
253
254
255
256
257 % Gradient w.r.t vx along directions G1,G2,G3,G4
258
259
260
261 del_g1_vx_e1=( -1 .* p_real .* sin(thetaG1 ) .* 0  + p_imag ...
       .* cos(thetaG1) .* 0);
262
```

```
263 del_g1_vx_e2=1i*( -1 .* p_imag .* sin(thetaG1) .* 0 - ...
      p_real .* cos(thetaG1) .* 0);
264
265 del_g2_vx_e1=( -1 .* p_real .* sin(thetaG2 ) .* 1 + p_imag ...
      .* cos(thetaG2) .* 1);
266
267 del_g2_vx_e2=1i*( -1 .* p_imag .* sin(thetaG2 ) .* 1 - ...
      p_real .* cos(thetaG2) .* 1);
268
269
270 del_g3_vx_e1=( -1.* p_real .* sin(thetaG3 ) .* 1 + p_imag ...
      .* cos(thetaG3) .* 1);
271
272 del_g3_vx_e2=1i*( -1 .* p_imag .* sin(thetaG3  .* 1) - ...
      p_real .* cos(thetaG3) .* 1);
273
274
275 del_g4_vx_e1=( -1 .* p_real .* sin(thetaG4) .* 1 + p_imag ...
      .* cos(thetaG4) .* 1);
276
277 del_g4_vx_e2=1i*( -1 .* p_imag .* sin(thetaG4 ) .* 1 - ...
      p_real .* cos(thetaG4) .* 1);
278
279
280
281             G1_vx=del_g1_vx_e1+del_g1_vx_e2;
282
283             G2_vx=del_g2_vx_e1+del_g2_vx_e2;
284
285             G3_vx=del_g3_vx_e1+del_g3_vx_e2;
286
287             G4_vx=del_g4_vx_e1+del_g4_vx_e2;
288
289
290
291
292
293  % Gradient w.r.t vy along directions G1,G2,G3,G4
294
295
296 del_g1_vy_e1=( -1* p_real .* sin(thetaG1 ) .* 1 + p_imag .* ...
      cos(thetaG1) .* 1);
297
298 del_g1_vy_e2=1i*( -1* p_imag .* sin(thetaG1 ) .* 1 - p_real ...
      .* cos(thetaG1) .* 1);
299
300
```

```
301 del_g2_vy_e1=(-1* p_real .* sin(thetaG2) .* 1 + p_imag .* ...
      cos(thetaG2) .* 1);
302
303 del_g2_vy_e2=1i*( -1* p_imag .* sin(thetaG2 ) .* 1 - p_real ...
      .* cos(thetaG2) .* 1);
304
305
306 del_g3_vy_e1=(-1* p_real .* sin(thetaG3) .* -1 + p_imag .* ...
      cos(thetaG3) .* -1);
307
308 del_g3_vy_e2=1i*(-1* p_imag .* sin(thetaG3) .* -1 - p_real ...
      .* cos(thetaG3) .* -1);
309
310
311 del_g4_vy_e1=( -1* p_real .* sin(thetaG4 ) .* 0 + p_imag .* ...
      cos(thetaG4) .* 0);
312
313 del_g4_vy_e2=1i*(-1* p_imag .* sin(thetaG4 ) .* 0 - p_real ...
      .* cos(thetaG4) .* 0);
314
315
316
317
318             G1_vy = del_g1_vy_e1+del_g1_vy_e2;
319
320
321             G2_vy = del_g2_vy_e1+del_g2_vy_e2;
322
323             G3_vy = del_g3_vy_e1+del_g3_vy_e2;
324
325
326             G4_vy = del_g4_vy_e1+del_g4_vy_e2;
327
328
329   %-------------------------------------------------------------
330
331
332
333             p1=proj1 - m1;
334             p2=proj2 - m2;
335             p3=proj3 - m3;
336             p4=proj4 - m4;
337
338
339
340             del_fu_G1=2*ifft2(p1,n,n);
341             del_fu_G2=2*ifft2(p2,n,n);
342             del_fu_G3=2*ifft2(p3,n,n);
```

```
343             del_fu_G4=2*ifft2(p4,n,n);
344
345
346
347
348
349
350
351
352
353 DEL_F_real_p_REAL = (real(del_fu_G1) .* real(G1_real_p)) + ...
       (real(del_fu_G2) .* real(G2_real_p)) + (real(del_fu_G3) ...
       .* real(G3_real_p)) + (real(del_fu_G4) .* real(G4_real_p));
354
355
356 DEL_F_imag_p_REAL = (real(del_fu_G1) .* real(G1_imag_p)) + ...
       (real(del_fu_G2) .* real(G2_imag_p)) + (real(del_fu_G3) ...
       .* real(G3_imag_p)) + (real(del_fu_G4) .* real( G4_imag_p));
357
358
359 DEL_F__vx_REAL    = (real(del_fu_G1) .* real(G1_vx))     + ...
       (real(del_fu_G2) .* real(G2_vx)) + (real(del_fu_G3) .* ...
       real(G3_vx)) + (real(del_fu_G4) .* real( G4_vx));
360
361
362 DEL_F__vy_REAL     = (real(del_fu_G1) .* real(G1_vy))    + ...
       (real(del_fu_G2) .* real(G2_vy)) + (real(del_fu_G3) .* ...
       real(G3_vy)) + (real(del_fu_G4) .* real(G4_vy));
363
364
365
366
367 DEL_F_real_p_IMAG = (imag(del_fu_G1) .* imag(G1_real_p)) + ...
       (imag(del_fu_G2) .* imag(G2_real_p)) + (imag(del_fu_G3) ...
       .* imag(G3_real_p)) + (imag(del_fu_G4) .* imag(G4_real_p));
368
369
370 DEL_F_imag_p_IMAG = (imag(del_fu_G1) .* imag(G1_imag_p)) + ...
       (imag(del_fu_G2) .* imag(G2_imag_p)) + (imag(del_fu_G3) ...
       .* imag(G3_imag_p)) + (imag(del_fu_G4) .* imag(G4_imag_p));
371
372
373 DEL_F__vx_IMAG     = (imag(del_fu_G1) .* imag(G1_vx)) + ...
       (imag(del_fu_G2) .* imag(G2_vx)) + (imag(del_fu_G3) .* ...
       imag(G3_vx)) + (imag(del_fu_G4) .* imag(G4_vx));
374
375
```

```
376 DEL_F__vy_IMAG     = (imag(del_fu_G1) .* imag(G1_vy)) + ...
       (imag(del_fu_G2) .* imag(G2_vy)) + (imag(del_fu_G3) .* ...
       imag(G3_vy)) + (imag(del_fu_G4) .* imag(G4_vy));
377
378
379
380
381             DEL_F_real_p = DEL_F_real_p_REAL + ...
                   DEL_F_real_p_IMAG;
382
383             DEL_F_imag_p = DEL_F_imag_p_REAL + ...
                   DEL_F_imag_p_IMAG;
384
385             DEL_F__vx =  DEL_F__vx_REAL + DEL_F__vx_IMAG;
386
387             DEL_F__vy = DEL_F__vy_REAL + DEL_F__vy_IMAG;
388
389
390
391
392 %-------------------------------------------------------------------
393 %------ Gradient for Regularization ---------------
394 %-------------------------------------------------------------------
395
396
397
398
399
400
401  %------------------ Middle ---------
402
403
404
405 for i=2:n-1
406
407     for j=2:n-1
408
409
410
411         DEL_R_real_p(i,j) = (2 * lamda_1)*(  p_real(i,j+1) ...
               - p_real(i,j)  + p_real(i,j-1) - p_real(i,j) + ...
               p_real(i+1,j)-p_real(i,j)  + ...
               p_real(i-1,j)-p_real(i,j) );
412
413         DEL_R_imag_p(i,j) = (2 * lamda_1)*(  p_imag(i,j+1) ...
               - p_imag(i,j)  + p_imag(i,j-1) - p_imag(i,j) + ...
               p_imag(i+1,j)-p_imag(i,j)  + ...
               p_imag(i-1,j)-p_imag(i,j) ) ;
```

```
414
415          DEL_R__vx(i,j) =     (2 * lamda_2)*(  v_x(i,j+1) - ...
                 v_x(i,j)        +  v_x(i,j-1)-v_x(i,j)        + ...
                 v_x(i+1,j)-v_x(i,j)    + v_x(i-1,j)-v_x(i,j) );
416
417          DEL_R__vy(i,j) =     (2 * lamda_2)*(  ...
                 v_y(i,j+1)-v_y_m(i,j)      +  ...
                 v_y(i,j-1)-v_y_m(i,j)       + ...
                 v_y(i+1,j)-v_y_m(i,j)        + ...
                 v_y(i-1,j)-v_y(i,j) );
418
419
420
421       end
422
423  end
424
425
426   %------------------ Left ------------
427
428
429   j=1;
430
431      for i=2:n-1
432
433
434
435          DEL_R_real_p(i,j) = (2 * lamda_1)*( p_real(i,j+1) ...
                 -p_real(i,j)  + p_real(i+1,j)-p_real(i,j) +  ...
                 p_real(i-1,j)-p_real(i,j)  );
436
437          DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                 p_imag(i,j+1)-p_imag(i,j)  + ...
                 p_imag(i+1,j)-p_imag(i,j) + ...
                 p_imag(i-1,j)-p_imag(i,j) ) ;
438
439          DEL_R__vx(i,j) =     (2 * lamda_2)*( ...
                 v_x(i,j+1)-v_x(i,j)       + v_x(i+1,j)-v_x(i,j) + ...
                 v_x(i-1,j)-v_x(i,j) );
440
441          DEL_R__vy(i,j) =     (2 * lamda_2)*( ...
                 v_y(i,j+1)-v_y(i,j)       + v_y(i+1,j)-v_y(i,j) + ...
                 v_y(i-1,j)-v_y(i,j)  );
442
443
444
445       end
446
```

```
447
448
449
450
451
452   %------------------ Right ----------
453
454
455   j=n;
456
457      for i=2:n-1
458
459
460
461          DEL_R_real_p(i,j) = (2 * lamda_1)*(  ...
                 (p_real(i,j-1)-p_real(i,j)) + ...
                 (p_real(i+1,j)-p_real(i,j))  + ...
                 (p_real(i-1,j)-p_real(i,j)) );
462
463          DEL_R_imag_p(i,j) = (2 * lamda_1)*(  ...
                 (p_imag(i,j-1)-p_imag(i,j)) + ...
                 (p_imag(i+1,j)-p_imag(i,j))  + ...
                 (p_imag(i-1,j)-p_imag(i,j)) ) ;
464
465          DEL_R__vx(i,j) =    (2 * lamda_2)*(  ...
                 (v_x(i,j-1)-v_x(i,j))       + ...
                 (v_x(i+1,j)-v_x(i,j))      + ...
                 (v_x(i-1,j)-v_x(i,j)) );
466
467          DEL_R__vy(i,j) =    (2 * lamda_2)*(  ...
                 (v_y(i,j-1)-v_y(i,j))      + ...
                 (v_y(i+1,j)-v_y(i,j))       + ...
                 (v_y(i-1,j)-v_y(i,j)) );
468
469
470
471       end
472
473
474
475
476
477
478
479
480   %------------------ Top ----------------
481
482
```

```
483 i=1;
484
485     for j=2:n-1
486
487
488
489         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
                (p_real(i,j+1)-p_real(i,j)) + ...
                (p_real(i,j-1)-p_real(i,j)) + ...
                (p_real(i+1,j)-p_real(i,j))   );
490
491         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                (p_imag(i,j+1)-p_imag(i,j)) + ...
                (p_imag(i,j-1)-p_imag(i,j)) + ...
                (p_imag(i+1,j)-p_imag(i,j))   ) ;
492
493         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
                (v_x(i,j+1)-v_x(i,j))       + ...
                (v_x(i,j-1)-v_x(i,j))       + ...
                (v_x(i+1,j)-v_x(i,j))       );
494
495         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
                (v_y(i,j+1)-v_y(i,j))       + ...
                (v_y(i,j-1)-v_y(i,j))      + ...
                (v_y(i+1,j)-v_y(i,j))        );
496
497
498
499      end
500
501
502
503  %------------------ Bottom ----------
504
505
506
507 i=n;
508
509     for j=2:n-1
510
511
512
513         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
                (p_real(i,j+1)-p_real(i,j)) + ...
                (p_real(i,j-1)-p_real(i,j))   + ...
                (p_real(i-1,j)-p_real(i,j)) );
514
```

```
515         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                (p_imag(i,j+1)-p_imag(i,j)) + ...
                (p_imag(i,j-1)-p_imag(i,j))   + ...
                (p_imag(i-1,j)-p_imag(i,j)) ) ;
516
517         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
                (v_x(i,j+1)-v_x(i,j))       + ...
                (v_x(i,j-1)-v_x(i,j))         + ...
                (v_x(i-1,j)-v_x(i,j)) );
518
519         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
                (v_y(i,j+1)-v_y(i,j))       + ...
                (v_y(i,j-1)-v_y(i,j))         + ...
                (v_y(i-1,j)-v_y(i,j)) );
520
521
522
523       end
524
525
526
527
528   %------------------ TL------------
529
530
531    i=1;
532    j=1;
533
534
535
536         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
                (p_real(i,j+1)-p_real(i,j))  + ...
                (p_real(i+1,j)-p_real(i,j))   );
537
538         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                (p_imag(i,j+1)-p_imag(i,j))  + ...
                (p_imag(i+1,j)-p_imag(i,j))   ) ;
539
540         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
                (v_x(i,j+1)-v_x(i,j))        + ...
                (v_x(i+1,j)-v_x(i,j))       );
541
542         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
                (v_y(i,j+1)-v_y(i,j))        + ...
                (v_y(i+1,j)-v_y(i,j))        );
543
544
545
```

```
546
547 %------------------ TR  ----------------
548
549 i=1;
550 j=n;
551
552         DEL_R_real_p(i,j) = (2 * lamda_1)*(  ...
                (p_real(i+1,j)-p_real(i,j))  + ...
                (p_real(i,j-1)-p_real(i,j)) );
553
554         DEL_R_imag_p(i,j) = (2 * lamda_1)*(   ...
                (p_imag(i+1,j)-p_imag(i,j))  + ...
                (p_imag(i,j-1)-p_imag(i,j)) ) ;
555
556         DEL_R__vx(i,j) =    (2 * lamda_2)*(  ...
                (v_x(i+1,j)-v_x(i,j))      + ...
                (v_x(i,j-1)-v_x(i,j)) );
557
558         DEL_R__vy(i,j) =    (2 * lamda_2)*(   ...
                (v_y(i+1,j)-v_y(i,j))     + ...
                (v_y(i,j-1)-v_y(i,j)) );
559
560
561
562
563
564
565 %------------------ BL ----------------
566
567 i=n;
568 j=1;
569
570
571         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
                (p_real(i,j+1)-p_real(i,j))  + ...
                (p_real(i-1,j)-p_real(i,j)) );
572
573         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                (p_imag(i,j+1)-p_imag(i,j))  + ...
                (p_imag(i-1,j)-p_imag(i,j)) ) ;
574
575         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
                (v_x(i,j+1)-v_x(i,j))        + ...
                (v_x(i-1,j)-v_x(i,j)) );
576
577         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
                (v_y(i,j+1)-v_y(i,j))        + ...
                (v_y(i-1,j)-v_y(i,j)) );
```

```
578
579
580
581
582  %------------------ BR -----------------
583
584
585  i=n;
586  j=n;
587
588
589
590          DEL_R_real_p(i,j) = (2 * lamda_1)*(  ...
                 (p_real(i,j-1)-p_real(i,j))   + ...
                 (p_real(i-1,j)-p_real(i,j)) );
591
592          DEL_R_imag_p(i,j) = (2 * lamda_1)*(  ...
                 (p_imag(i,j-1)-p_imag(i,j))   + ...
                 (p_imag(i-1,j)-p_imag(i,j)) ) ;
593
594          DEL_R__vx(i,j) =    (2 * lamda_2)*(  ...
                 (v_x(i,j-1)-v_x(i,j))         + ...
                 (v_x(i-1,j)-v_x(i,j)) );
595
596          DEL_R__vy(i,j) =    (2 * lamda_2)*(  ...
                 (v_y(i,j-1)-v_y(i,j))         + ...
                 (v_y(i-1,j)-v_y(i,j)) );
597
598  %-------------------------------------------------------
599  %------- Regularization ----------
600  %---------------------------------------
601
602
603
604
605
606   p_real_r_x= 0;
607   p_imag_r_x=0;
608   v_x_r_x=0;
609   v_y_r_x=0;
610
611
612   p_real_r_y= 0;
613   p_imag_r_y=0;
614   v_x_r_y=0;
615   v_y_r_y=0;
616
617  for i=1:n
```

```
618
619      for j=1:n-1
620
621
622
623          p_real_r_x =+ (real(p_m(i,j+1))-real(p_m(i,j))) * ...
                 (real(p_m(i,j+1))-real(p_m(i,j)));
624          p_imag_r_x =+ (imag(p_m(i,j+1))-imag(p_m(i,j))) * ...
                 (imag(p_m(i,j+1))-imag(p_m(i,j)))      ;
625          v_x_r_x =+ (v_x_m(i,j+1)-v_x_m(i,j) * ...
                 v_x_m(i,j+1)-v_x_m(i,j));
626          v_y_r_x =+ (v_y_m(i,j+1)-v_y_m(i,j) * ...
                 v_y_m(i,j+1)-v_y_m(i,j));
627
628
629
630       end
631
632  end
633
634
635
636  for j=1:n
637       for i=1:n-1
638
639
640               p_real_r_y =+ ...
                      ((real(p_m(i+1,j))-real(p_m(i,j)) )* ...
                      (real(p_m(i+1,j))-real(p_m(i,j)))) ;
641               p_imag_r_y =+ ...
                      ((imag(p_m(i+1,j))-imag(p_m(i,j)) )* ...
                      (imag(p_m(i+1,j))-imag(p_m(i,j))) );
642               v_x_r_y =+ ((v_x_m(i+1,j)-v_x_m(i,j)) * ...
                      (v_x_m(i+1,j)-v_x_m(i,j)));
643               v_y_r_y =+ ((v_y_m(i+1,j)-v_y_m(i,j) ) * ...
                      (v_y_m(i+1,j)-v_y_m(i,j)) );
644
645
646       end
647  end
648
649
650  p_real_r = p_real_r_x + p_real_r_y;
651  p_imag_r = p_imag_r_x + p_imag_r_y;
652  v_x_r = v_x_r_x + v_x_r_y;
653  v_y_r = v_y_r_x + v_y_r_y;
654
655
```

```
656
657
658 f=sum(f_obj)+(lamda_1*p_real_r)
659     +(lamda_1*p_imag_r)
660     +(lamda_2*v_x_r)+(lamda_2*v_y_r);
661
662
663 g=[(DEL_F_real_p + DEL_R_real_p ) ;(DEL_F_imag_p + ...
       DEL_R_imag_p);(DEL_F__vx + DEL_R__vx);(DEL_F__vy + ...
       DEL_R__vy)];
```

### A2: Objective function in Image space

```
1
2  function [f g] = objFun(A)
3  % n     – The size of the Image
4  % R     – Radius of the vessel
5  % V_max – Maximum velocity of blood
6  % p     – Proton density matrix
7  % v_x   –  Matrix containg x components of Velocity vector
8  % v_y   –  Matrix containg y components of Velocity vector
9  % (G_0,G_1,G_neg_1) – Gradient Sensitization vectors
10 %
11
12 n=32;
13 lamda_1=1e-4;
14 lamda_2=1e-1;
15
16
17 [c,d] = size(A);
18 p_real   = A(1:c/4,:);
19 p_imag = A((c/4)+1:(c/4)+d,:);
20 v_x = A(((c/4)+d)+1:(c/4)+2*d,:);
21 v_y = A(((c/4)+2*d)+1:c,:);
22
23 R=n/8;
24 v_max=1;
25 v_y_m=zeros(n,n);
26 p_m= zeros(n,n);
27 snr=25;
28 G_0=zeros(n,n);
29 G_1=ones(n,n);
30 G_neg_1=-(ones(n,n));
31
32
```

```
33
34 %-------------------------------------------------------
35 %------- Measurement ----------
36 %-------------------------------------------
37
38 for l = 1:n
39     for m = 1:n
40
41         if (distance([l,m], [n/2,m]) ≥ R)
42             p_m(l,m) = 1i;
43             v_x_m(l,m) = 0;
44
45         else
46
47             p_m(l,m) =  2.5;
48             v_x_m(l,m) = v_max*(1-((distance([l,m], ...
                   [n/2,m]))^2/R^2));
49
50
51         end
52
53
54     end
55 end
56
57
58 [c,d] = size(A);
59 p_real   = A(1:c/4,:);
60 p_imag = A((c/4)+1:(c/4)+d,:);
61 v_x = A(((c/4)+d)+1:(c/4)+2*d,:);
62 v_y = A(((c/4)+2*d)+1:c,:);
63
64
65 m1=p_m.*(exp(-1i*(G_0 .* v_x_m + G_1 .* v_y_m)));
66 m2=p_m.*(exp(-1i*(G_1 .* v_x_m + G_1 .* v_y_m)));
67 m3=p_m.*(exp(-1i*(G_1 .* v_x_m + G_neg_1 .* v_y_m)));
68 m4=p_m.*(exp(-1i*(G_1 .* v_x_m + G_0 .* v_y_m)));
69
70
71
72
73
74
75 k=1;
76
77  for i=1:n
78     for j=1:n
79
```

```
80
81
82          real_G1 = (p_real(i,j)*cos((G_0(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j))) ) + ...
                (p_imag(i,j)*sin((G_0(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j)) ) ) - real(m1(i,j));
83          imag_G1 = (p_imag(i,j)*cos((G_0(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j))) ) - ...
                (p_real(i,j)*sin((G_0(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j)) ) ) - imag(m1(i,j));
84
85          real_G2 = (p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j))) ) + ...
                (p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j)) ) ) - real(m2(i,j));
86          imag_G2 = (p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j))) ) - ...
                (p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
                (G_1(i,j)*v_y(i,j)) ) ) - imag(m2(i,j));
87
88          real_G3 = (p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
                (G_neg_1(i,j)*v_y(i,j))) ) + ...
                (p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
                (G_neg_1(i,j)*v_y(i,j)) ) ) - real(m3(i,j));
89          imag_G3 = (p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
                (G_neg_1(i,j)*v_y(i,j))) ) - ...
                (p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
                (G_neg_1(i,j)*v_y(i,j)) ) ) - imag(m3(i,j));
90
91          real_G4 = (p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
                (G_0(i,j)*v_y(i,j))) ) + ...
                (p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
                (G_0(i,j)*v_y(i,j)) ) ) - real(m4(i,j));
92          imag_G4 = (p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
                (G_0(i,j)*v_y(i,j))) ) - ...
                (p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
                (G_0(i,j)*v_y(i,j)) ) ) - imag(m4(i,j));
93
94
95
96          f1 = (real_G1)^2 + (imag_G1)^2;
97
98          f2 = (real_G2)^2 + (imag_G2)^2;
99
100         f3 = (real_G3)^2 + (imag_G3)^2;
101
102         f4 = (real_G4)^2 + (imag_G4)^2;
103
```

```
104
105
106          f_obj(k)=f1+f2+f3+f4;
107
108          k=k+1;
109
110   % Gradient w.r.t real(p) along directions G1,G2,G3,G4
111
112  G1_real_p = 2*(real_G1)*cos((G_0(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j))) - ...
        2*(imag_G1)*sin((G_0(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j)) );
113
114  G2_real_p = 2*(real_G2)*cos((G_1(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j))) - ...
        2*(imag_G2)*sin((G_1(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j)) );
115
116  G3_real_p = 2*(real_G3)*cos((G_1(i,j)*v_x(i,j) ) + ...
        (G_neg_1(i,j)*v_y(i,j))) - ...
        2*(imag_G3)*sin((G_1(i,j)*v_x(i,j) ) + ...
        (G_neg_1(i,j)*v_y(i,j)) );
117
118  G4_real_p = 2*(real_G4)*cos((G_1(i,j)*v_x(i,j) ) + ...
        (G_0(i,j)*v_y(i,j))) - ...
        2*(imag_G4)*sin((G_1(i,j)*v_x(i,j) ) + ...
        (G_0(i,j)*v_y(i,j)) );
119
120
121  % Gradient w.r.t imag(p) along directions G1,G2,G3,G4
122
123  G1_imag_p = 2*(real_G1)*sin((G_0(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j))) + ...
        2*(imag_G1)*cos((G_0(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j)) );
124
125  G2_imag_p = 2*(real_G2)*sin((G_1(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j))) + ...
        2*(imag_G2)*cos((G_1(i,j)*v_x(i,j) ) + ...
        (G_1(i,j)*v_y(i,j)) );
126
127  G3_imag_p = 2*(real_G3)*sin((G_1(i,j)*v_x(i,j) ) + ...
        (G_neg_1(i,j)*v_y(i,j))) + ...
        2*(imag_G3)*cos((G_1(i,j)*v_x(i,j) ) + ...
        (G_neg_1(i,j)*v_y(i,j)) );
128
129  G4_imag_p = 2*(real_G4)*sin((G_1(i,j)*v_x(i,j) ) + ...
        (G_0(i,j)*v_y(i,j))) + ...
```

```
      2*(imag_G4)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_0(i,j)*v_y(i,j)) );
130
131
132   % Gradient w.r.t vx along directions G1,G2,G3,G4
133
134   G1_vx_e1 = 2*(real_G1)*( -1* ...
      p_real(i,j)*sin((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_0(i,j) +  ...
      p_imag(i,j)*cos((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)))* G_0(i,j));
135
136
137   G1_vx_e2 = 2*(imag_G1)*( -1* ...
      p_imag(i,j)*sin((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_0(i,j) -  ...
      p_real(i,j)*cos((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)))* G_0(i,j));
138
139
140   G2_vx_e1 = 2*(real_G2)*( -1* ...
      p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_1(i,j) +  ...
      p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)))* G_1(i,j));
141
142   G2_vx_e2 = 2*(imag_G2)*( -1* ...
      p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_1(i,j) -  ...
      p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)))* G_1(i,j);
143
144   G3_vx_e1 = 2*(real_G3)*( -1* ...
      p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_neg_1(i,j)*v_y(i,j)) ) *G_1(i,j) +  ...
      p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_neg_1(i,j)*v_y(i,j))))* G_1(i,j);
145
146   G3_vx_e2 = 2*(imag_G3)*( -1* ...
      p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_neg_1(i,j)*v_y(i,j)) ) *G_1(i,j) -  ...
      p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_neg_1(i,j)*v_y(i,j))))* G_1(i,j);
147
148   G4_vx_e1 = 2*(real_G4)*( -1* ...
      p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_0(i,j)*v_y(i,j)) ) *G_1(i,j) +  ...
      p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
```

```
      (G_0(i,j)*v_y(i,j))))* G_1(i,j);
149
150 G4_vx_e2 = 2*(imag_G4)*( -1* ...
      p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_0(i,j)*v_y(i,j)) ) *G_1(i,j) -  ...
      p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_0(i,j)*v_y(i,j))))* G_1(i,j);
151
152  % Gradient w.r.t vy along directions G1,G2,G3,G4
153
154 G1_vy_e1 = 2*(real_G1)*( -1* ...
      p_real(i,j)*sin((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_1(i,j) +  ...
      p_imag(i,j)*cos((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j))))* G_1(i,j);
155
156 G1_vy_e2 = 2*(imag_G1)*( -1* ...
      p_imag(i,j)*sin((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_1(i,j) -  ...
      p_real(i,j)*cos((G_0(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j))))* G_1(i,j);
157
158
159 G2_vy_e1 = 2*(real_G2)*( -1* ...
      p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_1(i,j) +  ...
      p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j))))* G_1(i,j);
160
161 G2_vy_e2 = 2*(imag_G2)*( -1* ...
      p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j)) ) *G_1(i,j) -  ...
      p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_1(i,j)*v_y(i,j))))* G_1(i,j);
162
163 G3_vy_e1 = 2*(real_G3)*( -1* ...
      p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
      (G_neg_1(i,j)*v_y(i,j)) ) *G_neg_1(i,j) +  ...
      p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
      (G_neg_1(i,j)*v_y(i,j))))* G_neg_1(i,j);
164
165  G3_vy_e2 = 2*(imag_G3)*( -1* ...
       p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
       (G_neg_1(i,j)*v_y(i,j)) ) *G_neg_1(i,j) -  ...
       p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
       (G_neg_1(i,j)*v_y(i,j))))* G_neg_1(i,j);
166
167
```

```
168 G4_vy_e1 = 2*(real_G4)*( -1* ...
       p_real(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
       (G_0(i,j)*v_y(i,j)) ) *G_0(i,j) +  ...
       p_imag(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
       (G_0(i,j)*v_y(i,j))))* G_0(i,j);
169
170 G4_vy_e2 = 2*(imag_G4)*( -1* ...
       p_imag(i,j)*sin((G_1(i,j)*v_x(i,j) ) + ...
       (G_0(i,j)*v_y(i,j)) ) *G_0(i,j) -  ...
       p_real(i,j)*cos((G_1(i,j)*v_x(i,j) ) + ...
       (G_0(i,j)*v_y(i,j))))* G_0(i,j);
171
172
173
174 g_real_p(i,j) = G1_real_p+G2_real_p+G3_real_p+G4_real_p;
175
176 g_imag_p(i,j) = G1_imag_p+G2_imag_p+G3_imag_p+G4_imag_p;
177
178 g_vx(i,j)     = G1_vx_e1+G1_vx_e2+G2_vx_e1
179         +G2_vx_e2+G3_vx_e1+G3_vx_e2
180         +G4_vx_e1+G4_vx_e2;
181
182 g_vy(i,j)     =  G1_vy_e1+G1_vy_e2+G2_vy_e1
183         +G2_vy_e2+G3_vy_e1+G3_vy_e2
184         +G4_vy_e1+G4_vy_e2;
185
186
187
188      end
189
190
191  end
192
193
194
195
196
197 %-------------------------------------------------
198 %--------Gradient for Regularization --------
199 %-------------------------------------------------------
200
201
202
203
204
205
206  %------------------ Middle ----------
207
```

```
208
209
210 for i=2:n-1
211
212     for j=2:n-1
213
214
215
216         DEL_R_real_p(i,j) = (2 * lamda_1)*(  p_real(i,j+1) ...
               - p_real(i,j)  + p_real(i,j-1) - p_real(i,j) + ...
               p_real(i+1,j)-p_real(i,j)  + ...
               p_real(i-1,j)-p_real(i,j) );
217
218         DEL_R_imag_p(i,j) = (2 * lamda_1)*(  p_imag(i,j+1) ...
               - p_imag(i,j)  + p_imag(i,j-1) - p_imag(i,j) + ...
               p_imag(i+1,j)-p_imag(i,j)  + ...
               p_imag(i-1,j)-p_imag(i,j) ) ;
219
220         DEL_R__vx(i,j) =    (2 * lamda_2)*(  v_x(i,j+1) - ...
               v_x(i,j)       +  v_x(i,j-1)-v_x(i,j)       + ...
               v_x(i+1,j)-v_x(i,j)    + v_x(i-1,j)-v_x(i,j) );
221
222         DEL_R__vy(i,j) =    (2 * lamda_2)*(  ...
               v_y(i,j+1)-v_y_m(i,j)     +  ...
               v_y(i,j-1)-v_y_m(i,j)      + ...
               v_y(i+1,j)-v_y_m(i,j)       + ...
               v_y(i-1,j)-v_y(i,j) );
223
224
225
226      end
227
228 end
229
230
231  %------------------ Left --------------------
232
233
234  j=1;
235
236     for i=2:n-1
237
238
239
240         DEL_R_real_p(i,j) = (2 * lamda_1)*( p_real(i,j+1) ...
               -p_real(i,j)  + p_real(i+1,j)-p_real(i,j) +  ...
               p_real(i-1,j)-p_real(i,j)  );
241
```

```
242         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                p_imag(i,j+1)-p_imag(i,j)  + ...
                p_imag(i+1,j)-p_imag(i,j) + ...
                p_imag(i-1,j)-p_imag(i,j) ) ;
243
244         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
                v_x(i,j+1)-v_x(i,j)      + v_x(i+1,j)-v_x(i,j) + ...
                v_x(i-1,j)-v_x(i,j) );
245
246         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
                v_y(i,j+1)-v_y(i,j)      + v_y(i+1,j)-v_y(i,j) + ...
                v_y(i-1,j)-v_y(i,j)  );
247
248
249
250      end
251
252
253
254
255
256
257   %------------------ Right --------------------
258
259
260   j=n;
261
262      for i=2:n-1
263
264
265
266         DEL_R_real_p(i,j) = (2 * lamda_1)*(  ...
              (p_real(i,j-1)-p_real(i,j)) + ...
              (p_real(i+1,j)-p_real(i,j))  + ...
              (p_real(i-1,j)-p_real(i,j)) );
267
268         DEL_R_imag_p(i,j) = (2 * lamda_1)*(  ...
              (p_imag(i,j-1)-p_imag(i,j)) + ...
              (p_imag(i+1,j)-p_imag(i,j))  + ...
              (p_imag(i-1,j)-p_imag(i,j)) ) ;
269
270         DEL_R__vx(i,j) =    (2 * lamda_2)*(  ...
              (v_x(i,j-1)-v_x(i,j))       + ...
              (v_x(i+1,j)-v_x(i,j))      + ...
              (v_x(i-1,j)-v_x(i,j)) );
271
272         DEL_R__vy(i,j) =    (2 * lamda_2)*(  ...
              (v_y(i,j-1)-v_y(i,j))      + ...
```

```
            (v_y(i+1,j)-v_y(i,j))        + ...
            (v_y(i-1,j)-v_y(i,j)) );
273
274
275
276     end
277
278
279
280
281
282
283
284
285  %------------------ Top ---------------
286
287
288 i=1;
289
290     for j=2:n-1
291
292
293
294         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
            (p_real(i,j+1)-p_real(i,j)) + ...
            (p_real(i,j-1)-p_real(i,j)) + ...
            (p_real(i+1,j)-p_real(i,j))   );
295
296         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
            (p_imag(i,j+1)-p_imag(i,j)) + ...
            (p_imag(i,j-1)-p_imag(i,j)) + ...
            (p_imag(i+1,j)-p_imag(i,j))   ) ;
297
298         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
            (v_x(i,j+1)-v_x(i,j))       + ...
            (v_x(i,j-1)-v_x(i,j))       + ...
            (v_x(i+1,j)-v_x(i,j))       );
299
300         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
            (v_y(i,j+1)-v_y(i,j))       + ...
            (v_y(i,j-1)-v_y(i,j))      + ...
            (v_y(i+1,j)-v_y(i,j))        );
301
302
303
304     end
305
306
```

```
307
308  %------------------ Bottom ------------------
309
310
311
312 i=n;
313
314     for j=2:n-1
315
316
317
318         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
               (p_real(i,j+1)-p_real(i,j)) + ...
               (p_real(i,j-1)-p_real(i,j))   + ...
               (p_real(i-1,j)-p_real(i,j)) );
319
320         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
               (p_imag(i,j+1)-p_imag(i,j)) + ...
               (p_imag(i,j-1)-p_imag(i,j))   + ...
               (p_imag(i-1,j)-p_imag(i,j)) ) ;
321
322         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
               (v_x(i,j+1)-v_x(i,j))       + ...
               (v_x(i,j-1)-v_x(i,j))         + ...
               (v_x(i-1,j)-v_x(i,j)) );
323
324         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
               (v_y(i,j+1)-v_y(i,j))       + ...
               (v_y(i,j-1)-v_y(i,j))         + ...
               (v_y(i-1,j)-v_y(i,j)) );
325
326
327
328      end
329
330
331
332
333  %------------------ TL------------------------
334
335
336   i=1;
337   j=1;
338
339
340
341         DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
               (p_real(i,j+1)-p_real(i,j))  + ...
```

```
            (p_real(i+1,j)-p_real(i,j))   );
342
343         DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
            (p_imag(i,j+1)-p_imag(i,j))  + ...
            (p_imag(i+1,j)-p_imag(i,j))   ) ;
344
345         DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
            (v_x(i,j+1)-v_x(i,j))        + ...
            (v_x(i+1,j)-v_x(i,j))       );
346
347         DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
            (v_y(i,j+1)-v_y(i,j))        + ...
            (v_y(i+1,j)-v_y(i,j))        );
348
349
350
351
352   %------------------ TR  ---------------------
353
354   i=1;
355   j=n;
356
357         DEL_R_real_p(i,j) = (2 * lamda_1)*(  ...
            (p_real(i+1,j)-p_real(i,j))  + ...
            (p_real(i,j-1)-p_real(i,j)) );
358
359         DEL_R_imag_p(i,j) = (2 * lamda_1)*(   ...
            (p_imag(i+1,j)-p_imag(i,j))  + ...
            (p_imag(i,j-1)-p_imag(i,j)) ) ;
360
361         DEL_R__vx(i,j) =    (2 * lamda_2)*(  ...
            (v_x(i+1,j)-v_x(i,j))      + ...
            (v_x(i,j-1)-v_x(i,j)) );
362
363         DEL_R__vy(i,j) =    (2 * lamda_2)*(   ...
            (v_y(i+1,j)-v_y(i,j))     + ...
            (v_y(i,j-1)-v_y(i,j)) );
364
365
366
367
368
369
370   %------------------ BL ----------------------------
371
372   i=n;
373   j=1;
374
```

```
375
376          DEL_R_real_p(i,j) = (2 * lamda_1)*( ...
                 (p_real(i,j+1)-p_real(i,j))  + ...
                 (p_real(i-1,j)-p_real(i,j)) );
377
378          DEL_R_imag_p(i,j) = (2 * lamda_1)*( ...
                 (p_imag(i,j+1)-p_imag(i,j))  + ...
                 (p_imag(i-1,j)-p_imag(i,j)) ) ;
379
380          DEL_R__vx(i,j) =    (2 * lamda_2)*( ...
                 (v_x(i,j+1)-v_x(i,j))        + ...
                 (v_x(i-1,j)-v_x(i,j)) );
381
382          DEL_R__vy(i,j) =    (2 * lamda_2)*( ...
                 (v_y(i,j+1)-v_y(i,j))        + ...
                 (v_y(i-1,j)-v_y(i,j)) );
383
384
385
386
387      %------------------ BR --------------------
388
389
390      i=n;
391      j=n;
392
393
394
395          DEL_R_real_p(i,j) = (2 * lamda_1)*(  ...
                 (p_real(i,j-1)-p_real(i,j))   + ...
                 (p_real(i-1,j)-p_real(i,j)) );
396
397          DEL_R_imag_p(i,j) = (2 * lamda_1)*(  ...
                 (p_imag(i,j-1)-p_imag(i,j))   + ...
                 (p_imag(i-1,j)-p_imag(i,j)) ) ;
398
399          DEL_R__vx(i,j) =    (2 * lamda_2)*(  ...
                 (v_x(i,j-1)-v_x(i,j))         + ...
                 (v_x(i-1,j)-v_x(i,j)) );
400
401          DEL_R__vy(i,j) =    (2 * lamda_2)*(  ...
                 (v_y(i,j-1)-v_y(i,j))         + ...
                 (v_y(i-1,j)-v_y(i,j)) );
402
403
404
405
406      %-------------------------------------------------
```

```
407 %————————————  Regularization ——————————
408 %————————————————————————————————————————
409
410
411
412
413
414  p_real_r_x= 0;
415  p_imag_r_x=0;
416  v_x_r_x=0;
417  v_y_r_x=0;
418
419
420  p_real_r_y= 0;
421  p_imag_r_y=0;
422  v_x_r_y=0;
423  v_y_r_y=0;
424
425 for i=1:n
426
427     for j=1:n-1
428
429
430
431         p_real_r_x =+ (p_real(i,j+1)-p_real(i,j)) * ...
                (p_real(i,j+1)-p_real(i,j));
432         p_imag_r_x =+ (p_imag(i,j+1)-p_imag(i,j)) * ...
                (p_imag(i,j+1)-p_imag(i,j))      ;
433         v_x_r_x =+ (v_x(i,j+1)-v_x(i,j) * v_x(i,j+1)-v_x(i,j));
434         v_y_r_x =+ (v_y(i,j+1)-v_y(i,j) * v_y(i,j+1)-v_y(i,j));
435
436
437
438      end
439
440 end
441
442
443
444 for j=1:n
445      for i=1:n-1
446
447
448              p_real_r_y =+ ((p_real(i+1,j)-p_real(i,j) )* ...
                    (p_real(i+1,j)-p_real(i,j))) ;
449              p_imag_r_y =+ ((p_imag(i+1,j)-p_imag(i,j) )* ...
                    (p_imag(i+1,j)-p_imag(i,j)) );
```

```
450                 v_x_r_y =+ ((v_x(i+1,j)-v_x(i,j)) * ...
                        (v_x(i+1,j)-v_x(i,j)));
451                 v_y_r_y =+ ((v_y(i+1,j)-v_y(i,j) ) * ...
                        (v_y(i+1,j)-v_y(i,j)) );
452
453
454       end
455   end
456
457
458   p_real_r = p_real_r_x + p_real_r_y;
459   p_imag_r = p_imag_r_x + p_imag_r_y;
460   v_x_r = v_x_r_x + v_x_r_y;
461   v_y_r = v_y_r_x + v_y_r_y;
462
463   f=sum(f_obj)+lamda_1*p_real_r
464       +lamda_1*p_imag_r
465       +lamda_2*v_x_r+lamda_2*v_y_r;
466
467
468   g=[g_real_p;g_imag_p;g_vx;g_vy];
```

### A3: Optimization function call

```
1   options = optimset('GradObj','on','Display','iter',
2                                      'MaxIter',500,'TolX',1e-200,
3                                      'TolFun',1e-200,'MaxFunEvals',
4                                      1000000,'LargeScale','off',
5                                      'UseParallel','always');
6
7
8
9   p_initial= ones(n,n)+1i*zeros(n,n);
10
11
12
13
14  A=[real(p_initial);imag(p_initial);ones(n,n);zeros(n,n)];
15
16
17  tic;
18
19  [x fval] = fminunc('objFunK',A,options);
20
21  toc;
```

```
22
23
24
25
26 [c,d] = size(x);
27 p_real   = (x(1:c/4,:));
28 p_imag = (x((c/4)+1:(c/4)+d,:));
29 p=(p_real+1i*p_imag);
30 v_x = (x(((c/4)+d)+1:(c/4)+2*d,:));
31 v_y = (x(((c/4)+2*d)+1:c,:));
32
33 preal=reshape(v_x,1,[]);
34 pimag=reshape(v_x,1,[]);
35 vx=reshape(v_x,1,[]);
36 vy=reshape(v_x,1,[]);
```

# Bibliography


[Aga03] Vivek Agarwal. Total variation regularization and l-curve method for the selection of regularization parameter. Technical report, 2003.

[BB98] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. IOP Publishing Ltd., 1998.

[Ber69] G. D. Bergland. A guided tour of the fast fourier transform. *IEEE Spectrum*, 6(7):41 –52, 1969.

[BPT88] Mario Bertero, Tomaso A. Poggio, and Vincent Torre. Ill-posed problems in early vision. *Proceeding of the IEEE*, 76(8), August 1988.

[Bri74] E.O. Brigham. *The Fast Fourier Transform*. Prentice-Hall, 1974.

[Hah60] E.L. Hahn. Detection of sea-water motion by nuclear precession. *Journal of Geophysical Research*, 65(1):776–777, 1960.

[Han10] Per Christian Hansen. *Discrete Inverse Problems: Insight and Algorithms*. SIAM, 2010.

[HBTV99] E. Mark Haacke, Robert W. Brown, Michael R. Thompson, and Ramesh Venkatesan. *Magnetic Resonance Imaging:: Physical Principles and Sequence Design*. John Wiley and Sons, Inc, 1999.

[HNO06] Per Christian Hansen, James G. Nagy, and Dianne P. O'Leary. *Deblurring Images: Matrices, Spectra and Filtering*. SIAM, 2006.

[Idi99] Jerome Idier. Regularization tools and models for image and signal reconstruction. In *Proceeding of the 3rd International Conference on Inverse Problems in Engineering*, 1999.

[IL93] John Huston III and Richard L.Ehman. Comparison of time-of-flight and phase-contrast mr neuroangiographic techniques. *Journal of continuing medical education in radiology*, 13:5–19, January 1993.

[Moz08] Mehrad Mozafari. Model-based tissue segmentation from simulated partial k-space MRI data. Master's thesis, McMaster University, 2008.

[NFL86] G.L. Nayer, D.N. Firmin, and D.B. Longmore. Blood flow imaing by cine magnetic resonance. *Journal of Computer Assisted Tomography*, 10:715, 1986.

[Nis96] D.G Nishimura. *Principles of Magnetic Resonance Imaging*. IEEE Press, 1996.

[NMP86] D.G. Nishimura, A. Macovski, and J.M. Pauly. Magnetic resonance angiography. *IEEE transactions on Medical Imaging*, MI-5(3):140, 1986.

[Pau07] John Pauly. Non-cartesian reconstruction. Technical report, Stanford University, Stanford University, 2007.

[Pip99] James G. Pipe. Motion correction with PROPELLER MRI: Application to head motion and free-breathing cardiac imaging. *Magnetic Resonance in Medicine*, 42(5):963–969, 1999.

[TA77] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. Winston, 1977.

[Vog96] C.R. Vogel. Non-convergence of the l-curve regularization parameter selection method. *Inverse problems*, 12:535–547, 1996.