Compressive Deconvolution of MRI Imaging via ℓ_1-ℓ_2 Regularization

by

TALON JOHNSON

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2021

ACKNOWLEDGEMENTS

I would like to first thank my family for always being understanding and supportive of me in all of my endeavors during my time as I matriculated through my graduate studies. The dedication you have towards me has greatly impacted the way perceive myself as a scholar, a mathematician, but most in importantly as an individual. Without your love, I would not be where I am today.

Secondly, I wish to thank my advisor, Dr. Jianzhong Su, for his guidance and expertise throughout my graduate-level mathematical and research journey. Without his support, I wouldn't have been able to navigate the Ph.D. and thesis process. I'm genuinely grateful for everything he has done in my graduate studies here at UTA. I'd also like to thank my advising committee, Dr. Hristo Kojouharov, Dr. Ren-Cang Li, Dr. Li Wang, and Dr. Jimin Ren, for their interest in my research and for taking the time to serve in my comprehensive and dissertation.

I would like to thank Lona, Libby, Laura, and Michael for their time and assistance throughout my graduate career.Additionally, I would like to thank Dwight, Ariel, Omomayowa, John, Crystal, Anthony, Amanda, Saul, Michelangelo, and many others, for their help and encouragement during my five years as a graduate student here at UTA. As well as the fun times we got to spend with each other outside of the math department. You all are the best.

Finally, I would also like to extend my appreciation to Dr. Duane Cooper and Dr. Shelby Wilson for encouraging me to pursue a Ph.D. They continue to support and provide me with insight to this very day.

July 30, 2021

ABSTRACT

Compressive Deconvolution of MRI Imaging via ℓ_1-ℓ_2 Regularization

Talon Johnson, Ph.D.

The University of Texas at Arlington, 2021

Supervising Professor: Jianzhong Su

The evolution of technology has drastically impacted the imaging field, particularly magnetic resonance imaging (MRI). Compared to other imaging technologies, MRI offers multiple contrasting mechanisms to distinguish tissues and fat, is radiation-free, and provides anatomical and molecular information about the tissue in question. However, data acquisition times to produce those images require a patient to lie still for a relatively long time. Consequently, it may lead to the voluntary or involuntary movement of the patient due to discomfort. Combined with the underlying issue of inherent noise, MRI is often blurry and contains artifacts. Mathematically, one can describe this behavior as the convolution between the MRI and some unwanted PSF.

In this thesis, we present a new approach to speed up the MR data acquisition through sparse signal reconstruction and deconvolving the unwanted convolution simultaneously. This approach is part of an ever-growing area known as compressive deconvolution. We propose a novel compressive deconvolution method for two dimensional MRI data sets via an $\ell_1 - \ell_2$ regularization via ℓ_1–$magic : TV_2$ and Tikhonov regularization.

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 Problems and Limitations in Medical Imaging

Magnetic resonance imaging (MRI) is a non-invasive way to view anatomical structures inside the body. Strong magnetic and radio waves help produce internal images of the body, unlike X-ray and CT scans that use radiation. Instead, MRI is based on the nuclear magnetic resonance (NMR) phenomenon [9].

A significant disadvantage of MRI is that it requires a patient to lie still for a relatively long time for data acquisition. For children, the elderly, or patients with severe injuries, these long acquisition times can be discomforting, thus resulting in voluntary or involuntary motion. Those motions can lead to blurring artifacts to arise in the resulting MR image. This has been a significant challenge in MRI as these artifacts degrade image quality on top of the fact that MRI is subjugated to a low signal-to-noise ratio (SNR) already. As a result, the health of the tissue in question can be misinterpreted by doctors and may lead to incorrect treatment for a patient [32].

Techniques like parallel imaging [43] and constrained reconstruction methods like compressed sensing [38] have helped speed up the data acquisition process. However, using these methods still leaves an undesired low SNR, and there is still the issue of blurring artifacts due to voluntary or involuntary movement in the system. Other artifacts can come from implants and other medical devices reacting to the magnetic field in the MR machine. Hardware limitations can cause artifacts to arise due to technical defects of MR machine components [33].

In this work, we will tackle the problem of how to handle the data acquisition and blurring artifacts using a novel approach in the area of compressive deconvolution. We first discuss imaging basics and how to mathematically described blurring in imaging systems.

1.2 Imaging Basics

Without loss of generality, an **image** $\boldsymbol{X}$ can be defined as a $n \times n$ real valued matrix, i.e. $X \in \mathbb{R}^{n \times n}$, with the ij^{th} entry is known as a **pixel**, where $i, j = 1, \ldots, n$. We shall denote each ij^{th} pixel of image X as x_{ij}. Each pixel is associated with value, usually called brightness or intensity of that pixel [29]. Image intensity can be in color (RGB, HSV, or CMYK), but for this work we will concern ourselves with grayscale intensity images. Pixel values in grayscale images usually take on integer values in the range $[0, 255]$, where 0 is black and 255 is white.

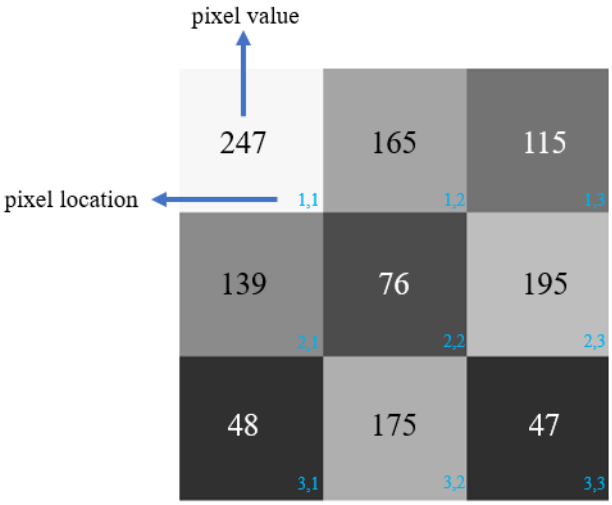

Figure 1.1: 3-by-3 grayscale image with pixel value at corresponding ij^{th} pixel location.

We will use MATLAB in order to perform several image processing and optimization algorithms. These algorithms will require many operations to be done on the pixel values, so it is best to convert images to double-precision floating-point numbers in the interval $[0, 1]$ [30], and this can be achieved by using MATLAB's built-in function `rescale` to do so. Several other built-in functions that are used for image processing in MATLAB included:

- `im2double` and,
- `mat2gray`.

They will serve to convert our images to double-precision floating-point numbers with grayscale intensity.

1.3 Image Blurring

A major concern that occurs in imaging processing is significant degradation of a captured image due to blurring and is often unavoidable. The blurring in images can arise from multiple source, such as limitation in optical systems, camera and object motion, limited aperture size and environmental effects [1, 5, 30]. One can describe image blurring as a linear mathematical model. Recall that the image X can be thought of as an $n \times n$ matrix whose entries are pixel values that represents the light intensity in ij^{th} position. We can arrange those pixels as a vector of length n^2 by stacking the columns of the matrix on top of each other. That is,

$$x = \text{vec}(X) = \begin{bmatrix} x_{11} \\ \vdots \\ x_{n1} \\ \vdots \\ x_{1n} \\ \vdots \\ x_{nn} \end{bmatrix}, \tag{1.1}$$

where x is the **vectorization** of X, denoted by $\text{vec}(X)$. In MATLAB, this can be done by simply typing `X(:)` in the command window. Similarly, we can denote the blurred representation of X as the matrix $B \in \mathbb{R}^{n\times n}$ and its vectorization as

$$b = \text{vec}(B) = \begin{bmatrix} b_{11} \\ \vdots \\ b_{n1} \\ \vdots \\ b_{1n} \\ \vdots \\ b_{nn} \end{bmatrix}. \tag{1.2}$$

The relationship between x and b is given by the existence of a large matrix $C \in \mathbb{R}^{N\times N}$, with $N = n^2$, such that

$$Cx = b. \tag{1.3}$$

The matrix C in (1.3) represents the blurring process that degrades the true image X to the the blurred image B.

Now we know that such a matrix existence, the natural question is: how do we obtain C? So suppose that we consider an all black image expect at a single bright pixel (pixel value is 1). Applying a blurring operation to this image will cause the bright pixel to be spread over its neighboring pixels, as illustrated by Figure 1.2. This pixel is known as the **point source**, and the function that describes the blurring affect on the image coming from the point source is known as the **point spread function (PSF)** [30].

One could think of the point source as a matrix with entries all equal zero, expect at the single entry whose entry is 1. With this in mind, we can obtain all the information about matrix C by finding all of its columns. This can be done by computing Ce_i for $i = 1, \ldots, N$. Here, e_i is the i^{th} unit vector. However, this task is meticulous task for large images sizes. In the next section, we will discus important properties of the PSF and various common PSFs in imaging systems.

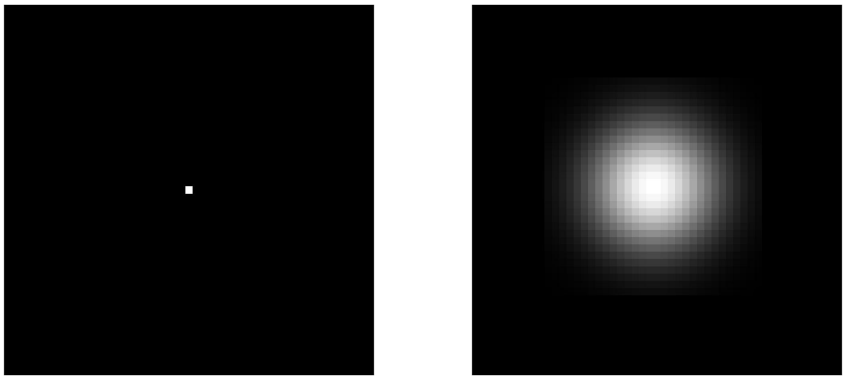

Figure 1.2: Left: Single Bright Pixel (Point Source). Right: The Blurred Point Source (Point Spread Function).

1.4 Properties of Blurs and Common Blur Types in Imaging Systems

The blurring of an image by some PSF is a localized phenomenon. In other words, the light intensity of the PSF is confided to a tiny area around the center of the PSF, the point source, and outside a certain radius, the intensity is essentially zero [30]. In Figure 1.2, the PSF is 0 outside 30 pixels from the center. Additionally, the shape of the PSF is independent from its location in the image. Figure 1.3 illustrates this. In this case, the PSF is known to be **spatially invariant** [30, 40]. In this thesis, we assume that this is case for all the PSFs we deal with as this characteristic occurs often in imaging systems.

Often, the PSF can be represented as an array of a much smaller dimension than of that of the blurred image. This is due to the local and linear nature of the PSF, resulting in storage conservation. As an example, in Figure 1.2 the PSF has size 30×30 compared to the 120×120 image size. We will simply refer to the **PSF array** as P with elements p_{ij}. As a remark, for deblurring algorithms will require that P will need to be the same size as B. For small P, the array will be embedded into a larger array of zeros; this process is known as **zero padding**.

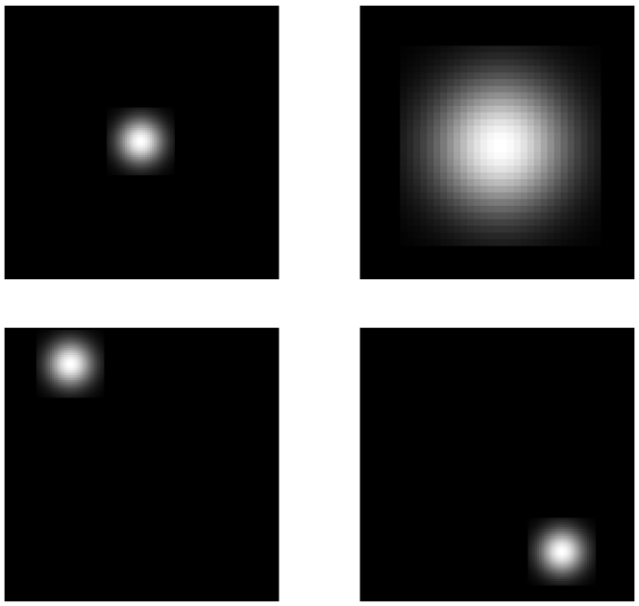

Figure 1.3: Top Left: Blurring of a single pixel. Top Right: Zoom of blurred pixel. Bottom Row: Spatial invariance of the pixel.

As discussed already, blurring is unavoidable in many imaging systems, including MRI. Here, we will discuss common types of blur that are encountered in the real-world and their analytical models that describe them, thus allowing P to be constructed from a function rather than experimentation.

Common Types of Blur

1. **Motion Blur:** This type of blur is a common result of camera panning or a fast moving object. In the case of **horizontal motion blur**, the point source is smeared into a horizontal line. Light is disturbed over a line that covers r pixels, in which the magnitude of each nonzero elements of P is r^{-1}. A similar argument can be made about **vertical motion blur**.
2. **Out-of-focus Blur:** This type of blur is the result of various parameters that affect the camera aperture. These parameters included: focal length, camera aperture size, distance between the camera and the observed scene, and

diffraction effects. The elements of array P associated with this type of blur can be given by the mathematical expression

$$p_{ij} = \begin{cases} 1/(\pi r^2), & \text{if } (i-k)^2 + (j-l)^2 \leq r^2 \\ 0, & \text{elsewhere,} \end{cases} \tag{1.4}$$

where (k, l) is the center of P and r is the radius of the blur.

3. **Atmospheric Turbulence Blur:** This type of blur represents a long term exposure to the atmosphere and is model by a two-dimensional Gaussian function [45] and the elements of the P are given by

$$p_{ij} = K \exp\left(- \frac{(i-k)^2 + (j-l)^2}{2\sigma^2} \right), \tag{1.5}$$

where (k, l) is the center of P, σ is the standard deviation of the Gaussian distribution, and $K = \frac{1}{2\pi\sigma^2}$ is the normalizing constant.

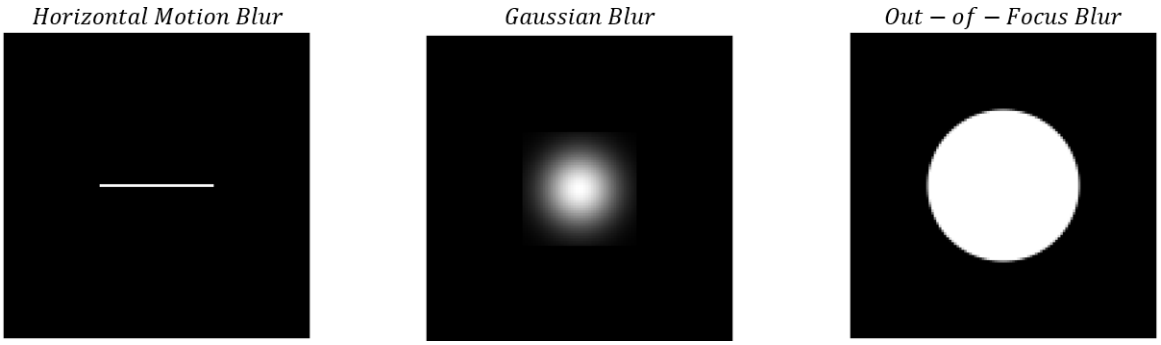

Figure 1.4: Common types of blurs discussed in this section.

Again, we can take the cumbersome approach to construct matrix C column by column for any specified PSF by simply aligning elements of P in the appropriate position and leaving zeros elsewhere in the column. Additionally, once given the true

image X and matrix C, we can compute the blurred image B one pixel at a time the following way

$$b_i = e_i^T b = e_i^T C x, \tag{1.6}$$

for $i = 1, \ldots, N$. Thus, we only need to work with rows of C to compute each pixel of the blurred image, b_i, as a weighted sum of the corresponding pixel, x_i, and its neighboring pixels in the true image. The weights are the elements of the rows in C. We make note that the weights are also given by the elements in P and the weighted sum operation is known as the **convolution**.

1.5 Boundary Conditions

When discussing the PSF in the previous section, we learned that a blurred pixel in B, call it b_{ij}, is the weighted sum of the corresponding pixel x_{ij} and its neighboring pixels. Noting this fact, a potential problem will occur at pixels b_{ij} near the boundary. This is because the PSF "spills over the edge" at the image boundary, in which information of the true image X is lost in the blurred image B that is recorded [30]; refer to Figure 1.5. That is, the weighted sum that contributes to the value of b_{ij}, may not take into account the values of the neighboring pixels that lie outside of the field of view.

For deblurring algorithms, this could lead to unwanted artifacts in the reconstruction of X near the boundary. Many good image deblurring algorithms take into account the behavior outside the boundary of the true image. However, most of the time, we do not know what that behavior is. Instead, we make certain assumption about this behavior and impose **boundary conditions** into the blurring model 1.3. We now discuss two of most common types of boundary conditions used: the zero and periodic boundary conditions.

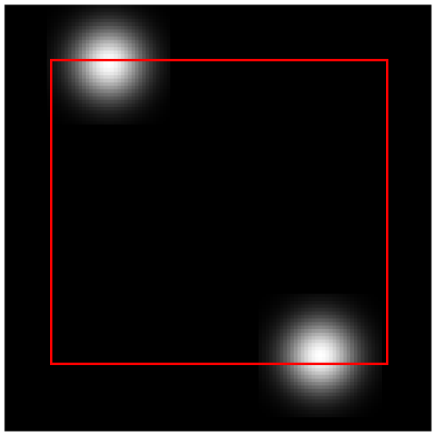

Figure 1.5: The PSF spilling over the edge of the image boundary (the red line). Those pixels outside the edge contribute to the blurred image.

Boundary Conditions

1. **Zero:** The **zero boundary condition** assumes that the exact image is black outside the field of view. This can be pictured as embedding the image X in a larger matrix of the form

$$\dot{X} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where $0 \in \mathbb{R}^{n \times n}$ represents the zero matrix.

2. **Periodic:** The **periodic boundary condition** assumes that image X repeats itself in all direction. One can think of this type of boundary condition as embedding X into a larger image consisting of copies of X:

$$\dot{X} = \begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}.$$

1.6 Special Structured Matrices

Now that we have laid out some fundamental components about the image blurring model, we are now ready to find a better way to construct the blurring process that is represented by matrix C. The structure of C will be dependent

on the boundary conditions we impose. We first discuss how these structures can arise from the one-dimensional convolution and then we will extend upon this in the two-dimensional case.

1.6.1 One-Dimensional Convolution, Toeplitz, and Circulant Matrices

Recall that a blurred image can be thought of as convolution between the true image and a specified PSF. In the one dimensional case, given two integrable functions $p(s)$ and $x(s)$, the convolution of p and x, denoted $p \circledast x$, produces a third function b defined by

$$b(s) = (p \circledast x)(s) = \int_{-\infty}^{\infty} p(s-t)x(t)\,dt, \tag{1.7}$$

for $t \in \mathbb{R}$. Equation 1.7 is also known as the **continuous convolution**. The function $b(s)$ is the weighted average of the values of $x(t)$, where the weights are given by the function $p(t)$. Graphically, the graph of b can produce by reflecting the graph of $p(t)$ to obtain $p(-t)$, shifting the reflected graph to obtain $p(s-t)$, and then sliding the graph of $p(s-t)$ across the graph of $x(t)$, stopping at each t to compute the integral of the product where the two graphs intersect [23].

The **discrete convolution** can be given as the summation over a finite number of terms. We do not give this formulation explicitly, but instead, demonstrate with an example how to compute the convolution of a one-dimensional image and PSF based

on the approach presented in [30]. Suppose the true image and PSF are represented respectively, by the one-dimensional arrays

$$x = \begin{bmatrix} w_1 \\ w_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline w_3 \\ w_4 \end{bmatrix} \quad \text{and } p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix},$$

where the w_i's represents pixels in the original scene that are outside the field of view. The corresponding blurred image is given by

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}.$$

One can compute $p \circledast x$ to produce b as follows :

1. Rotate the PSF, p, 180 degrees about the center.
2. Match coefficients of the rotated PSF array with those in x by placing the center of the PSF array over the i^{th} entry in x.
3. Multiply the corresponding components, and sum them to get the i^{th} entry of b.

Assuming that the center of p is p_3 we have that b_i entries are given by the system

$$\begin{aligned} b_1 &= p_5 w_1 + p_4 w_2 + p_3 x_1 + p_2 x_2 + p_1 x_3 \\ b_2 &= p_5 w_2 + p_4 x_1 + p_3 x_2 + p_2 x_3 + p_1 x_4 \\ b_3 &= p_5 x_1 + p_4 x_2 + p_3 x_3 + p_2 x_4 + p_1 x_5 \\ b_4 &= p_5 x_2 + p_4 x_3 + p_3 x_4 + p_2 x_5 + p_1 w_3 \\ b_5 &= p_5 x_3 + p_4 x_4 + p_3 x_5 + p_2 w_3 + p_1 w_4. \end{aligned} \tag{1.8}$$

We now can write the convolution as a matrix-vector multiplication,

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_5 & p_4 & p_3 & p_2 & p_1 & & & & \\ & p_5 & p_4 & p_3 & p_2 & p_1 & & & \\ & & p_5 & p_4 & p_3 & p_2 & p_1 & & \\ & & & p_5 & p_4 & p_3 & p_2 & p_1 & \\ & & & & p_5 & p_4 & p_3 & p_2 & p_1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \hline x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \hline w_3 \\ w_4 \end{bmatrix}. \tag{1.9}$$

Recall that the pixels outside the field of views, the w_i's, also contribute to the pixels of the observed blurred image. Again, we do not know what the behavior is outside the of the field, so we impose boundary conditions. Using the boundary conditions we discussed in Section 1.5 we have the following:

- Zero Boundary Condition. Here, $w_i = 0$, and thus (1.9) can be rewritten as

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_3 & p_2 & p_1 & & \\ p_4 & p_3 & p_2 & p_1 & \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ & p_5 & p_4 & p_3 & p_2 \\ & & p_5 & p_4 & p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}. \tag{1.10}$$

  Notice that the matrix C formed has entries such that $c_{ij} = c_{i+1,j+1}$. This type of matrix is known as a **Toeplitz matrix** [27].

- Periodic Boundary Conditions. In this case, we assume that the one-dimensional image is embedded into a larger image whose made up of replicas of itself in all directions. Thus, $w_1 = p_4$, $w_2 = p_5$, $w_3 = p_1$, and $w_4 = p_2$ and (1.9) can be rewritten as

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} p_3 & p_2 & p_1 & p_5 & p_4 \\ p_4 & p_3 & p_2 & p_1 & p_5 \\ p_5 & p_4 & p_3 & p_2 & p_1 \\ p_1 & p_5 & p_4 & p_3 & p_2 \\ p_2 & p_1 & p_5 & p_4 & p_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}. \tag{1.11}$$

Here the matrix C is a special type of Toeplitz matrix known as a **circulant matrix**. Notice that every row of a circulant matrix is a cyclic right shift of the one above it..

1.6.2 Two Dimensional Convolution, Block Toeplitz with Toeplitz Blocks (BTTB), and Block Circulant with Circulant Blocks (BCCB) Matrices

A recorded 2D blurred image B, can be thought of as the convolution of a 2D PSF array, P, with the 2D true image, X. Computing the 2D convolution has a similar procedure as the 1D case. To compute pixel b_{ij}, the procedure goes as follows:

1. Rotate the PSF, P, 180 degrees about the center.
2. Slide the center element of P so that it lies on top of the corresponding pixel value, x_{ij}.
3. Match coefficients of rotated P and X.
4. Multiply the corresponding components, and sum them to get the ij^{th} entry of B.

As an example, let

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, \; P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}, \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix},$$

with p_{22} being the center of P. Now suppose that we want to compute b_{22} corresponding to x_{22}. Following the first three step in procedure yields

$$\begin{bmatrix} x_{11}p_{33} & x_{12}p_{32} & x_{13}p_{31} \\ x_{21}p_{23} & x_{22}p_{22} & x_{23}p_{21} \\ x_{31}p_{13} & x_{32}p_{12} & x_{33}p_{11} \end{bmatrix},$$

and b_{22} is the weighted sum of x_{22} and its neighboring pixel value (step 4), that is,

$$b_{22} = p_{33}x_{11} + p_{32}x_{12} + p_{31}x_{13} +$$

$$p_{23}x_{21} + p_{22}x_{22} + p_{21}x_{23} +$$

$$p_{13}x_{31} + p_{12}x_{32} + p_{11}x_{33}.$$

Computing b_{22} is simple as all the neighboring pixels that contribute to its value are within the field of view and are within the radius of P. However, let us consider what happens at the edge, specifically, pixel b_{11}. Following the same procedure yields

$$\begin{matrix} \underline{?}p_{33} & \underline{?}p_{32} & \underline{?}p_{31} \\ \underline{?}p_{23} & x_{11}p_{22} & x_{12}p_{21} & x_{13} \\ \underline{?}p_{13} & x_{21}p_{12} & x_{22}p_{11} & x_{23} \\ & x_{31} & x_{32} & x_{33} \end{matrix},$$

with

$$b_{11} = p_{33}\underline{?} + p_{32}\underline{?} + p_{31}\underline{?} +$$
$$p_{23}\underline{?} + p_{22}x_{11} + p_{21}x_{12} +$$
$$p_{13}\underline{?} + p_{12}x_{21} + p_{11}x_{22}.$$

We again impose boundary conditions for the missing pixel values outside of the field of view. We have the following

- Zero Boundary Condition.

$$\begin{matrix} \underline{0}p_{33} & \underline{0}p_{32} & \underline{0}p_{31} \\ \underline{0}p_{23} & x_{11}p_{22} & x_{12}p_{21} & x_{13} \\ \underline{0}p_{13} & x_{21}p_{12} & x_{22}p_{11} & x_{23} \\ & x_{31} & x_{32} & x_{33} \end{matrix},$$

  with

$$b_{11} = p_{33}\underline{0} + p_{32}\underline{0} + p_{31}\underline{0} +$$
$$p_{23}\underline{0} + p_{22}x_{11} + p_{21}x_{12} +$$
$$p_{13}\underline{0} + p_{12}x_{21} + p_{11}x_{22}.$$

If we carry out this procedure with this boundary condition for all the remaining elements of B, we get the relationship between $b = \text{vec}(B)$ and $x = \text{vec}(X)$ is given by the matrix-vector multiplication

$$\underbrace{\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ \hline b_{21} \\ b_{22} \\ b_{32} \\ \hline b_{13} \\ b_{23} \\ b_{33} \end{bmatrix}}_{b=\text{vec}(B)} = \underbrace{\left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & & p_{21} & p_{11} & & & & \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & & & \\ & p_{32} & p_{22} & & p_{31} & p_{21} & & & \\ \hline p_{23} & p_{13} & & p_{22} & p_{12} & & p_{21} & p_{11} & \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ & p_{33} & p_{23} & & p_{32} & p_{22} & & p_{31} & p_{21} \\ \hline & & & p_{23} & p_{13} & & p_{22} & p_{12} & \\ & & & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ & & & & p_{33} & p_{23} & & p_{32} & p_{22} \end{array}\right]}_{C} \underbrace{\begin{bmatrix} x_{11} \\ x_{12} \\ x_{31} \\ \hline x_{21} \\ x_{22} \\ x_{32} \\ \hline x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}}_{x=\text{vec}(X)} . \tag{1.12}$$

Notice that the matrix in (1.12) has a block structure. In particular, it has a block Toeplitz structure and within each block lies a Toeplitz matrix. In general, the zero boundary condition forms a block structure matrix C known as a **block Toepliz with Toeplitz blocks (BTTB)** [30].

- Periodic Boundary Condition. Imposing the periodic boundary condition yields us with

$$\begin{array}{ccc} \underline{x_{33}}p_{33} & \underline{x_{31}}p_{32} & \underline{x_{32}}p_{31} \\ \underline{x_{13}}p_{23} & \multicolumn{2}{c}{\multirow{2}{*}{$\begin{bmatrix} x_{11}p_{22} & x_{12}p_{21} & x_{13} \\ x_{21}p_{12} & x_{22}p_{11} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$}} \\ \underline{x_{23}}p_{13} & & \end{array},$$

with

$$\begin{aligned} b_{11} = p_{33}\underline{x_{33}} + p_{32}\underline{x_{32}} + p_{31}\underline{x_{31}} + \\ p_{23}\underline{x_{13}} + p_{22}x_{11} + p_{21}x_{12} + \\ p_{13}\underline{x_{23}} + p_{12}x_{21} + p_{11}x_{22}. \end{aligned}$$

Again, carrying out this procedure with this boundary condition for all the remaining elements of B, we get the the matrix-vector multiplication

$$\underbrace{\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ \hline b_{21} \\ b_{22} \\ b_{32} \\ \hline b_{13} \\ b_{23} \\ b_{33} \end{bmatrix}}_{b=\mathrm{vec}(B)} = \underbrace{\left[\begin{array}{ccc|ccc|ccc} p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & p_{13} \\ p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & p_{23} \\ \hline p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} & p_{21} & p_{11} & p_{31} \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} & p_{11} & p_{31} & p_{21} \\ \hline p_{21} & p_{11} & p_{31} & p_{23} & p_{13} & p_{33} & p_{22} & p_{12} & p_{32} \\ p_{31} & p_{21} & p_{11} & p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} \\ p_{11} & p_{31} & p_{21} & p_{13} & p_{33} & p_{23} & p_{12} & p_{32} & p_{22} \end{array}\right]}_{C} \underbrace{\begin{bmatrix} x_{11} \\ x_{12} \\ x_{31} \\ \hline x_{21} \\ x_{22} \\ x_{32} \\ \hline x_{13} \\ x_{23} \\ x_{33} \end{bmatrix}}_{x=\mathrm{vec}(X)}. \tag{1.13}$$

Another block structure is presented in (1.13). The block structure matrix C, when periodic boundary conditions are imposed, is known as a **block circulant with circulant blocks (BCCB)** [30].

An important feature of the BCCB matrices is that they are normal; that is $C^*C = CC^*$, where * represents the conjugate transpose. Hence, they have **unitary spectral decomposition** or unitary eigendecomposition

$$C = U\Lambda U^*,$$

where U is a unitary matrix and Λ diagonal matrix whose entries are the eigenvalues of C. In particular,

$$C = \mathcal{F}^*\Lambda\mathcal{F}, \tag{1.14}$$

where $\mathcal{F}$ is the two-dimensional discrete Fourier transformation (DFT) matrix. The matrices $\mathcal{F}$ and $\mathcal{F}^*$ have the convenient property of performing matrix-vector multiplication without explicitly constructing $\mathcal{F}$ and $\mathcal{F}^*$, by using the Fast Fourier Transform (FFT). In MATLAB, the functions `fft2` and `ifft2` can be used to matrix-vector

multiplication of $\mathcal{F}$ and $\mathcal{F}^*$, respectively. Note, implementation of `fft2` and `ifft2` involve a sclaing factor, that is,

$$\texttt{fft2}(X) \longleftrightarrow \sqrt{N}\mathcal{F}x,$$

and

$$\texttt{ifft2}(X) \longleftrightarrow \frac{1}{\sqrt{N}}\mathcal{F}x.$$

Additionally, we can compute the eigenvalues of C easily since the first column of $\mathcal{F}$ are made up of 1's. So let denote the first column of C and $\mathcal{F}$ by c_1 and f_1 respectively. Notice that,

$$C = \mathcal{F}^*\Lambda\mathcal{F} \Rightarrow \mathcal{F}C = \Lambda\mathcal{F} \Rightarrow \mathcal{F}c_1 = \Lambda f_1 = \frac{1}{\sqrt{N}}\lambda,$$

where $\lambda \in \mathbb{R}^N$ is a vector containing all the eigenvalues of C. Thus, we can compute the eigenvalues of C by multiplying the matrix $\sqrt{N}\mathcal{F}$ by the first column of C, scaled by the square of the dimension. We can find the first column of C by the PSF and the MATLAB function `cirschift`, refer to [30], and we can compute b and efficiently since

$$b = Cx = \mathcal{F}^*\Lambda\mathcal{F}x. \tag{1.15}$$

1.6.3 Separable Blurs

In image processing, some PSF are capable to be decomposed in to their vertical and horizontal components. These types PSF are called **seperable** [30, 36]. So given a separable $m \times n$ PSF array P, we can rewrite P as

$$P = vh^T = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} \begin{bmatrix} h_1 & h_2 & \cdots & h_n \end{bmatrix},$$

where v represent the vertical components of the PSF that blurs across the columns of the image, h represents the horizontal components of the PSF that blurs the rows of the image.

It is known that if P is separable, then the P has rank 1 [30, 36] with entries given by $p_{ij} = v_i h_j$. Injecting this assertion into the expression in (1.12), matrix C takes the form

$$
\begin{aligned}
C &= \left[\begin{array}{ccc|ccc|ccc}
v_2h_2 & v_1h_2 & & v_2h_1 & v_1h_1 & & & & \\
v_3h_2 & v_2h_2 & v_1h_2 & v_3h_1 & v_2h_1 & v_1h_1 & & & \\
 & v_3h_2 & v_2h_2 & & v_3h_1 & v_2h_1 & & & \\
\hline
v_2h_3 & v_1h_3 & & v_2h_2 & v_1h_2 & & v_2h_1 & v_1h_1 & \\
v_3h_3 & v_2h_3 & v_1h_3 & v_3h_2 & v_2h_2 & v_1h_2 & v_3h_1 & v_2h_1 & v_1h_1 \\
 & v_3h_3 & v_2h_3 & & v_3h_2 & v_2h_2 & & v_3h_1 & v_2h_1 \\
\hline
 & & & v_2h_3 & v_1h_3 & & v_2h_2 & v_1h_2 & \\
 & & & v_3h_3 & v_2h_3 & v_1h_3 & v_3h_2 & v_2h_2 & v_1h_2 \\
 & & & & v_3h_3 & v_2h_3 & & v_3h_2 & v_2h_2
\end{array}\right] \\
&= \left[\begin{array}{ccc|ccc|ccc}
h_2 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & & & h_1 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & & & 0 & & \\
\hline
h_3 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & & & h_2 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & & & h_1 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & & \\
\hline
0 & & & h_3 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & & & h_2 \begin{bmatrix} v_2 & v_1 & \\ v_3 & v_2 & v_1 \\ & v_3 & v_2 \end{bmatrix} & &
\end{array}\right] \\
&= \left[\begin{array}{c|c|c}
h_2C_v & h_1C_v & 0 \\
\hline
h_3C_v & h_2C_v & h_1C_v \\
\hline
0 & h_3C_v & h_2C_v
\end{array}\right] \\
&= C_h \otimes C_v,
\end{aligned}
$$

the symbol $\otimes$ denotes the Kronecker product. In general, for any other boundary conditions, the matrix C for separable PSFs has the form

$$
C = C_h \otimes C_v = \begin{bmatrix}
h_{11}C_v & h_{12}C_v & \cdots & h_{1n}C_v \\
h_{21}C_v & h_{22}C_v & \cdots & h_{2n}C_v \\
\vdots & \vdots & & \vdots \\
h_{n1}C_v & h_{n2}C_v & \cdots & h_{nn}C_v
\end{bmatrix},
$$

where C_v is an $m \times m$ matrix and C_h is an $n \times n$ matrix. Recall, we assume that C is of size $n^2 \times n^2$, hence C_v and C_h are both of size $n \times n$.

Exploiting properties of the Kronecker, we can now write the blurring model in (1.3) as the matrix-matrix relation

$$B = C_v X C_h^T. \tag{1.16}$$

What (1.16) tells us is that we can produce B by first performing a convolution of the columns of X with the vertical components of P, then perform another convolution with the resulting rows with the horizontal components of P. Since matrix multiplication is associative, we can perform the convolution in vice versa.

1.6.4 Summary of Matrix Structures

The table below summarize the important structures for matrix C we have built up in this chapter based on the characteristics of the PSF and the type of boundary conditions we impose in the blurring model. For the remainder of this thesis, we call matrix C the **convolution or blurring matrix associated with the PSF**.

Matrix Structure of C		
Boundary Condition	Nonseparable PSF	Separable PSF
Zero	BTTB	Kronecker Product of Toeplitz matrices
Periodic	BCCB	Kronecker Product of circulant matrices

(1.17)

CHAPTER 2

COMPRESSIVE DECONVOLUTION

Another concern in image processing is to be able to store an image efficiently through highly compressed measurements without losing image fidelity and quality. However, it is often the case to acquire an accurate representation of the recorded image is achieved through enforcing the **Shannon-Nyquist theorem**. The Shannon-Nyquist theorem states that the image sampling rate must be at least twice the maximum frequency value of the image [23]. This theorem makes one assume that it is impossible to reconstruct an image without the sacrificing image fidelity or quality using highly compressed measurements, or so it would seem.

2.1 Compressed Sensing

The theory of **compressed sensing** has been used in a wide variety of applications in mathematics, computer science, radiology, and engineering which include image processing [21], magnetic resonance imaging (MRI) [55], synthetic aperture radar (SAR) imaging [54], and more. Through the umbrella of compressed sensing, far fewer measurements than what Shannon-Nyquist imposes are needed to get an approximate reconstruct of the desired image without much loss of meaning and granularity [10]-[13]. Given image $X \in \mathbb{R}^{n \times n}$, the standard compressed sensing linear model is given by

$$b = Ax + e, \tag{2.1}$$

where is $A \in \mathbb{R}^{m \times N}$ is the measurement matrix, $x \in \mathbb{R}^N$ is the vectorization of X, $e \in \mathbb{R}^m$ represents noise, and $b \in \mathbb{R}^m$ are compressed observed measurements. Note that $m << N = n^2$. Figure 2.1 helps illustrate this model.

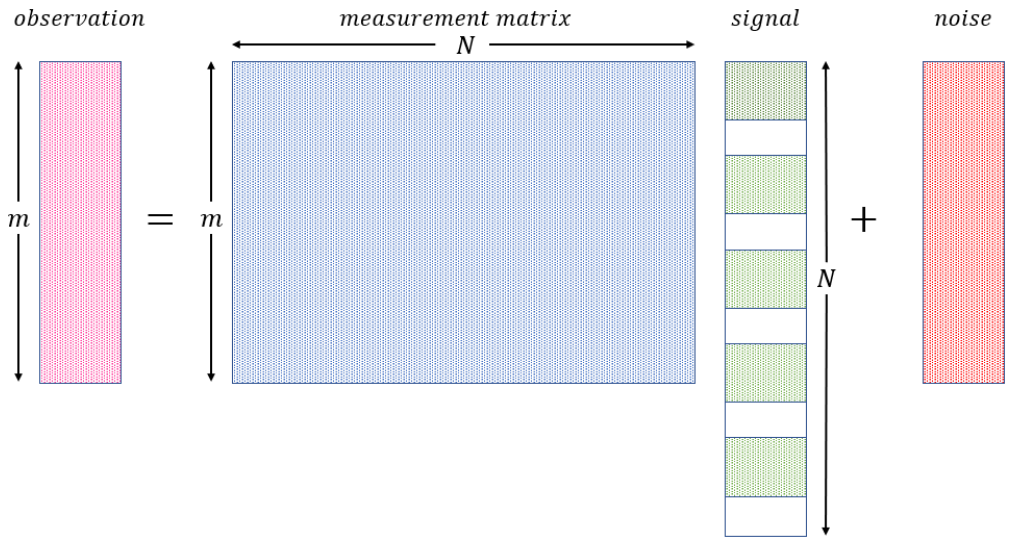

Figure 2.1: Compressed Sensing Model for sparse signal/image recovery.

Clearly, (2.1) presents us with an ill-posed problem. However, compressed sensing theory tells us that we can recover a unique solution x from measurements b provided that

- the image must have a sparse representation in a known basis (i.e., Fourier, wavelet, curvelet, etc.) and,
- the sampling is incoherent (random) which is conveyed through the **restricted isometry property** in [10].

The presence of noise allows one to see that sparsity is never really achieve. Fortunately, many images tend to be naturally sparse or almost sparse in a certain basis. That is to say that most of the coefficients are arbitrarily small, and only a few large coefficients are needed to represent the image. For example, the image of the "Cameraman" in

Figure 2.2 has been been taken in to the Fourier space, also referred to as the **k-space** in MRI [39]. Most of the Fourier coefficients in that space are small in magnitude, and only a few relatively large coefficients contain most of the information about the image. Here, we knock out coefficients whose magnitude are smaller than a certain tolerance (10^{-3}), and only 12% of the largest coefficients are needed to recover a decent approximation to the original image.

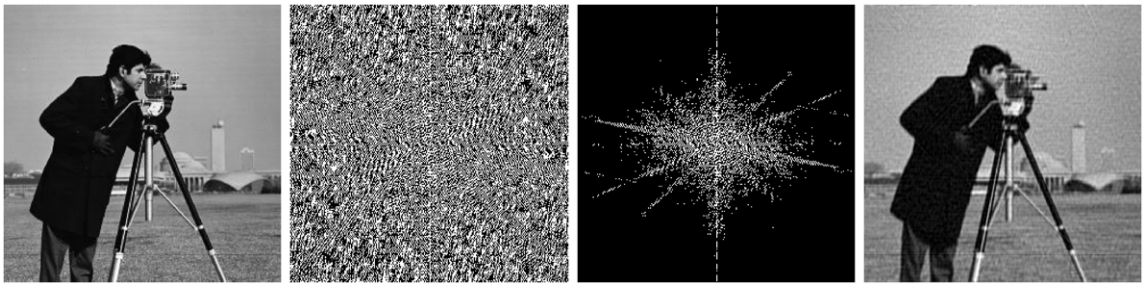

Figure 2.2: Left: Original Image. Mid-Left: Fully Sampled Image in k-space. Mid-Right 12% Sampling Right. Right: Recovered Image

Even though, the signal of interest is sparse, we do not know where the nonzero elements are located as a priori. There have been several algorithms proposed to solve this problem. Such algorithms incluede, ℓ_1-magic [14], basis pursuit denoising [16], and the Least Absolute Shrinkage and Selection Operator (LASSO) [48]. The first of which we will discuss further in Chapter 4 and extend upon in Chapter 5 in this thesis.

2.2 Deconvolution

In Chapter 1, we discussed that the blurring process is an inherent problem in imaging. The recorded blurred image, B, can be thought of as the convolution of the true image X and an undesired PSF, which can be further extended to a linear

system as in 1.3. The natural question is: can we recover X from B that is related by an undesired PSF? The process of recovering X from blurred measurements B is known as **deconvolution** [46], also referred to as **deblurring**.

Referring back to the model 1.3, the standard deconvolution model with noisy blurred measurements is given by

$$b = Cx + e, \tag{2.2}$$

where $C \in \mathbb{R}^{N\times N}$ denotes the convolution matrix associated with the PSF caused by a shaky camera, lack of focus, atmospheric turbulence, rippling, etc., $x \in \mathbb{R}^N$ and $e \in \mathbb{R}^N$ represents the vectorization of true image $X \in \mathbb{R}^{n\times n}$ signal and noise, respectively.

As with the compressed sensing, the objective of deconvolution is not an easy task to achieve. The reasons for this is that deconvolution is

- an ill-posed inverse problem, and
- in many situations the PSF is usually an unknown a priori.

Methods that assume the the PSF is known are called **non-blind deconvolution** methods [19], such as the Wiener filtering [53] and the Richardson-Lucy algorithms [44]. Analogously, **blind deconvolution** methods assume that the PSF is unknown and these methods fall under one of two categories: **a priori identification** or **joint identification** methods [8]. In our work, we assume that the PSF is known. Thus, we will be developing a non-blind deconvolution algorithm.

2.3 Compressive Deconvolution

In recent years, there have been algorithms developed to solve the compressed sensing and deconvolution problem jointly, rather than a sequential approach. A problem of this type is known as **compressive deconvolution**. This area is relatively

new and has shown applications in image processing [1] and medical ultrasound imaging [17].

The goal of compressive deconvolution is to recover image X from the recorded compressed and blurred measurements B. Naturally, another question is introduce: are blurred images compressible? It has been shown, that the blurring of an image by some PSF results in faster decay rates in the magnitude of the coefficients in some transformation basis [1]. We demonstrates this behavior of the sorted Fourier coefficient for the Cameraman image in Figure 2.3.

Figure 2.3: Comparison of sparsity of the original Cameraman image and its blurred version in the Fourier Domain (k-space). Left: Original Image. Middle: Blurred image with Gaussian PSF (17×17) with standard deviation 7. Right: Magnitude decay rates of the sorted Fourier coefficients.

The standard compressive deconvolution formulation can be given by

$$b = ACx + e. \tag{2.3}$$

We plan to solve (2.3) by developing a novel compressive deconvolution alternating minimization algorithm that joints $\ell_1 - magic : TV_2$ and Tikhonov regularization.

CHAPTER 3

MATHEMATICAL BACKGROUND

In this chapter, we will review some numerical optimization methods we will use to solve (5.1). We first discuss some background on how to solve the optimization problem

$$\min_x f(x) \text{ subject to } \|Ax - b\|_2 \leq \epsilon,$$

where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ and $f(x) : \mathbb{R}^n \mapsto \mathbb{R}$, is some convex objective function, for 2D image reconstruction. Methods included the Conjugate Gradient (CG) method, the Newton method, and the interior point method via log-barrier function.

We will then discuss techniques to solve the *least squares approximation* problem

$$\min_x \|Cx - d\|_2,$$

where $C \in \mathbb{R}^{m \times n}$ and $d \in \mathbb{R}^m$, for image deblurring. The methods used will be the singular value decomposition (SVD) and spectral filtering via truncated SVD (TSVD) and Tikhonov regularization.

3.1 Linear Conjugate Gradient Method

The Conjugate Gradient method is a very popular iterative method developed by Hestenes and Stiefel [34] to solve the large linear system $Ax = b$ of n linear equations in n steps, where A is symmetric positive definite. Normally, the goal is to find an approximate solution x to the linear system such that the residual

$$r = Ax - b \tag{3.1}$$

is less than some specified tolerance. In [42, p.101-111, p.120-125], this is equivalent to minimizing the following convex quadratic function

$$\min_x f(x) \text{ with } f(x) := \frac{1}{2}x^T Ax - b^T x. \tag{3.2}$$

Notice, taking the gradient of f, that is

$$\nabla f(x) = Ax - b, \tag{3.3}$$

we can see that it is equal to the residual r in (3.1). This means that $-r$ or the *negative gradient direction* is the fastest descent direction. Such a method that finds this descent direction iteratively is called the **method of steepest descent** [7, p. 481]. There is merit to this method in nonlinear systems, but for linear system the method tends to have slow convergence.

As an alternate approach, the CG method iteratively constructs a set of A-orthogonal search direction vectors denoted as $\{p_0, p_1, \ldots, p_n\}$. A set of vectors, $\{p_0, p_n, \ldots, p_n\}$, is called A-orthogonal if

$$p_i^T A p_j = 0, \ \forall i \neq j.$$

This choice of search directions with this property is important as $f(x)$ can be minimized successively along each of the individual search directions. Additionally, each of the approximation is the optimal solution from the subspace spanned by all the search direction vectors up to that point. Each of the successive points can be written as

$$x_{i+1} = x_i + \alpha_i p_i, \tag{3.4}$$

where α_i is how far we move along the direction p_i. This is equivalent to finding the minimum of $h(\alpha_i)$, where

$$
\begin{aligned}
h(\alpha_i) &= f(x_i + \alpha_i p_i) \\
&= \frac{1}{2}(x_i + \alpha_i p_i)^T A(x_i + \alpha_i p_i) - b^T(x_i + \alpha_i p_i) \\
&= \frac{1}{2}(x_i + \alpha_i p_i)^T A(x_i + \alpha_i p_i) - b^T x_i - \alpha_i b^T p_i \\
&= \frac{1}{2}(x_i^T A x_i + \alpha_i x_i^T A p_i + \alpha_i p_i^T A x_i + \alpha_i^2 p_i^T A p_i) - b^T x_i - \alpha_i b^T p_i \\
&= \frac{1}{2}\alpha_i^2 p_i^T A p_i + \alpha_i p_i^T (A x_i - b) + (\frac{1}{2} x_i^T A x_i - b^T x_i) \\
&= \frac{1}{2}\alpha_i^2 p_i^T A p_i + \alpha_i p_i^T (A x_i - b) + f(x_i).
\end{aligned}
$$

For each fixed x_i and α_i, we have that h assumes a minimal value when $h'(\alpha_i) = 0$, because the coefficient of α_i^2, $p_i^T A p_i$, is positive. That is,

$$
\begin{aligned}
h'(\alpha_i) &= \alpha_i p_i^T A p_i + p_i^T (A x_i - b) \\
&= 0,
\end{aligned}
$$

and solving for α_i to get

$$
\alpha_i = \frac{p_i^T r_i}{p_i A p_i}. \tag{3.5}
$$

Notice that we have a way to compute our approximations x_i iteratively as well as the residuals which have expression given by

$$
\begin{aligned}
r_i &= A x_i - b \\
&= A(x_{i-1} + \alpha_{i-1} p_{i-1}) - b \\
&= A x_{i-1} - b + \alpha_{i-1} A p_{i-1} \\
&= r_{i-1} + \alpha_{i-1} A p_{i-1}.
\end{aligned} \tag{3.6}
$$

Equations (3.4), (3.5), and (3.6) are dependent on the search directions p_i. The CG method sets $p_0 = r_0$ and iteratively defines the next search direction as

$$
p_i = -r_i + \beta_i p_{i-1}. \tag{3.7}
$$

Thus α_i can be simplified to

$$\alpha_i = \frac{r_i^T r_i}{p_i A p_i}, \tag{3.8}$$

and β_i is given by [42, p.101-111, p.120-125][7, p. 485] as

$$\beta_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}. \tag{3.9}$$

Thus, the CG method does not have to store the previous search directions, and the complexity of the method reduces to $\mathcal{O}(km)$, where m is number of nonzero entries of A and k is the number of CG iterations [42]. In summary, the CG method can be outlined as

$$\alpha_i = \frac{r_i^T r_i}{p_i A p_i}, \tag{3.10}$$

$$x_{i+1} = x_i + \alpha_i p_i, \tag{3.11}$$

$$r_{i+1} = r_{i-1} + \alpha_{i-1} A p_{i-1} \tag{3.12}$$

$$\beta_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i,} \tag{3.13}$$

$$p_i = -r_i + \beta_i p_{i-1}, \tag{3.14}$$

where $x_0 = 0$ and $p_0 = -r_0 = b - Ax_0$. The CG method will continue to iterate through this outlined procedure until $\frac{\|r_i\|}{\|r_0\|}$ is less than some tolerance or the maximum number of iteration has been reached [25].

3.2 Newton's Method for Unconstrained Minimization Problems

Newton's method is one of the most useful and well-known numerical methods for solving root-finding problems. This method can be extended to multivariate functions and attempts to find a point at which a function's gradient is zero using a quadratic approximation of the function. We will apply Newton's method to iteratively

solve the log barrier form of (5.1) by forming a series of quadratic approximation [3], which will be discussed later in this chapter. Now consider a real-valued function $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$. The second-order Taylor approximation of a single variable function, $f(x)$ near x, is given as follows

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2.$$

The multivariate function analog to this is

$$f(x + \Delta x) \approx f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x)\Delta x, \tag{3.15}$$

where ∇f represents the gradient and $\nabla^2 f$ represents the Hessian of f [2]. Assuming that the Hessian, $\nabla^2 f(x)$, is positive definite, we seek to find the Δx that minimizes $f(x + \Delta x)$. By setting the derivative of the right hand of (3.15) and setting it equal to zero

$$\begin{aligned} 0 &= \frac{d}{d\Delta x}\Big(f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x)\Delta x\Big) \\ &= \nabla f(x)^T + \Delta x^T \nabla^2 f(x), \end{aligned} \tag{3.16}$$

we get an explicit formula form for what is known as the Newton step or Newton direction [31, 42]

$$\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x). \tag{3.17}$$

Thus by iterating over x, the unique minimum is obtained at $x_{n+1} = x + \Delta x_n$, where Δx is given by (3.17). In practice, the step size is usually controlled in order to avoid divergence of the descent direction. A popular approach is to do a backtracking line search [47, p.108] where we add the parameter β and get the modified Newton iteration

$$x_{n+1} = x_n + \beta \cdot \Delta x_n. \tag{3.18}$$

Now that we have our scheme, given by (3.19), the question is now: when do we stop iterating? A typical stopping criterion for Newton's method involves the Newton decrement, denoted by $\lambda(x)$ [3, 6]. At a point x, the Newton decrement is expressed as

$$\lambda(x) = \left[\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right]^{\frac{1}{2}}. \tag{3.19}$$

A typical stopping criterion is given by

$$\frac{\lambda^2(x)}{2} \leq \epsilon, \tag{3.20}$$

where ϵ is a specified tolerance.

3.3 The Interior-Point Method via Log Barrier Function

Recall we wish to solve a problem of the form

$$\begin{aligned} &\min_x \ f_0(x) \\ &\text{subject to } f_i(x) \leq 0 \text{ for } i = 1, \ldots, m. \end{aligned} \tag{3.21}$$

We will convert this constrained problem to an unconstrained one in order for us to apply Newton's Method discussed in Section 3.2. In order for this to happen, we will embed the inequality constraints of (3.21) into the objective function using an interior point method via log-barrier function. Once embedded, we will iteratively solve it using Newton's method.

In order to use Newton's method, it is required that the objective function embedded with the inequality constraints is smooth and differentiable. Additionally,

the minimization problem must be penalized for not meeting the inequality constraints. Consider the indicator function $\mathscr{I} : \mathbb{R} \mapsto \mathbb{R}$,

$$\mathscr{I}(x) = \begin{cases} 0, & x < 0 \\ \infty, & x > 0. \end{cases} \tag{3.22}$$

One can change (3.21) as the following unconstrained problem

$$\min_x \ f_0(x) + \sum_{i=1}^{n} \mathscr{I}(f_i(x)). \tag{3.23}$$

Notice that if any of the inequality constraints is greater than 0, then the indicator function, $\sum_{i=1}^{n} \mathscr{I}(f_i(x))$, is infinity. The minimization problem is indeed penalized for not meeting at least one of the inequality constraints which are now part of the objective function, however, the objective function is clearly not differentiable. Thus, Newton's method can not be applied here.

So now, consider the alternative function

$$\hat{\mathscr{I}}(x) = \frac{-1}{\tau} \log(-x), \tag{3.24}$$

where $\mathbf{dom}(\hat{\mathscr{I}}) = \{x \in \mathbb{R} | x < 0\}$, and $\tau > 0$ is an accuracy parameter [6]. This function is differentiable and it does penalize the objective function. To see the latter condition, we observe some plots for various values of τ.

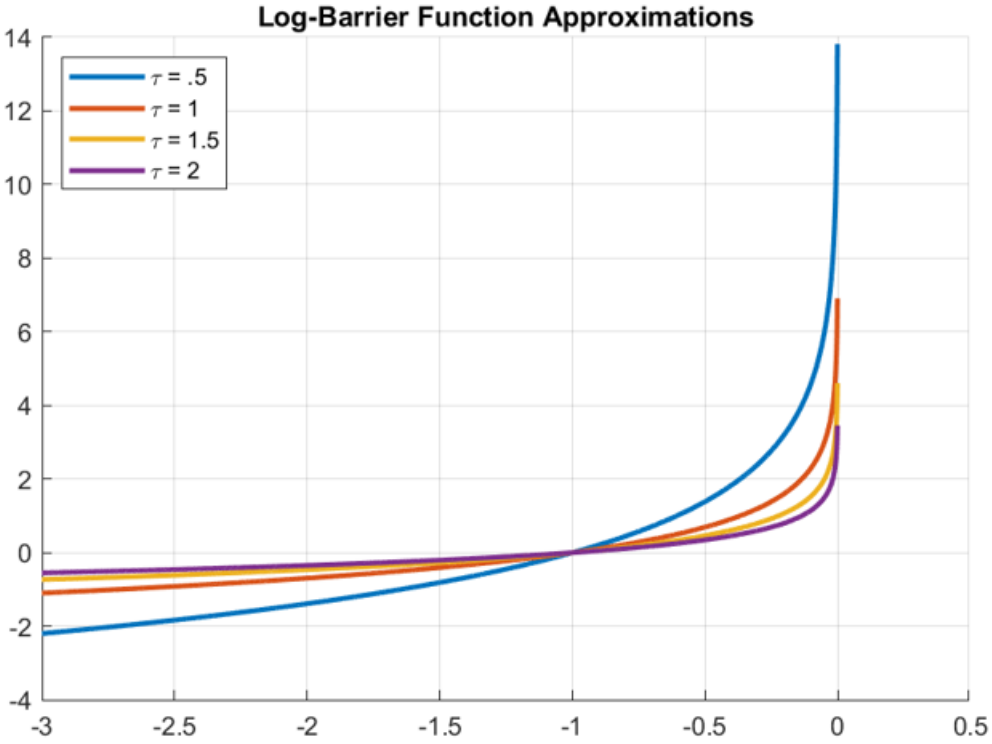

Figure 3.1: The Log-Barrier Function (3.24) for Various τ Values.

For large values of τ, we can get close approximations to the indicator function $\mathscr{I}$ as shown in Figure 3.1. Using this fact, we recast (3.21) as

$$\min_x \ f_0(x) + \frac{1}{\tau}\sum_{i=1}^{m} -\log(-f_i(x)). \tag{3.25}$$

Recall that Newton's method wishes to find a descent direction to minimize of our objective function around a point x using quadratic approximations. We have

$$f_0(x+\Delta x) \approx f(x) + \langle g_x, \Delta x\rangle + \frac{1}{2}\langle H_x \Delta x, \Delta x\rangle, \tag{3.26}$$

where g_x and H_x are the gradient and Hessian of the original objective function, respectively. We take note of the term

$$\phi(x) = \frac{1}{\tau}\sum_{i=1}^{m} -\log(-f_i(x)) \tag{3.27}$$

is known as the **log-barrier** function and the gradient, $\nabla\phi(x)$, and Hessian, $\nabla^2\phi(x)$ as

$$\nabla\phi(x) = \sum_{i=1}^{m} \frac{1}{-f_i(x)} \nabla f_i(x), \tag{3.28}$$

and

$$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{f_i(x)^2} \nabla f_i(x) \nabla f_i(x)^T + \sum_{i=1}^{m} \frac{1}{-f_i(x)} \nabla^2 f_i(x). \tag{3.29}$$

Equations (3.28) and (3.29) will be for later reference in Chapters 4 and 5. Thus, assuming our starting point x is feasible, the direction Δx along which a new approximation to minimize (3.27) is the solution to the following linear system

$$H_x \Delta x = -g_x. \tag{3.30}$$

The system is solved using the CG since the Hessian, H_x, is symmetric positive definite [14].

3.4 Least Squares Approximation

For a given matrix $C \in \mathbb{R}^{m \times n}$, square or rectangular, the goal of the least squares approximation is to find a close approximate solution, x^*, for the system

$$Cx = d,$$

whose has no solution. This is equivalent of trying to find the vector $x \in \mathbb{R}^n$ such that

$$\|Cx^* - d\|_2 = \min_x \|Cx - d\|_2. \tag{3.31}$$

Another interpretation is that we are trying to find the vector Cx^* in the column space of C, which we will denote it as W, that is the closest to d among all of the vectors in W. It turns out that the closest vector to d is the vector x^* such that

$$Cx^* = P_W d, \tag{3.32}$$

where $P_W d$ is known as the **orthogonal projection of d onto W** [4]. A geometric interpretation is given by Figure 3.2.

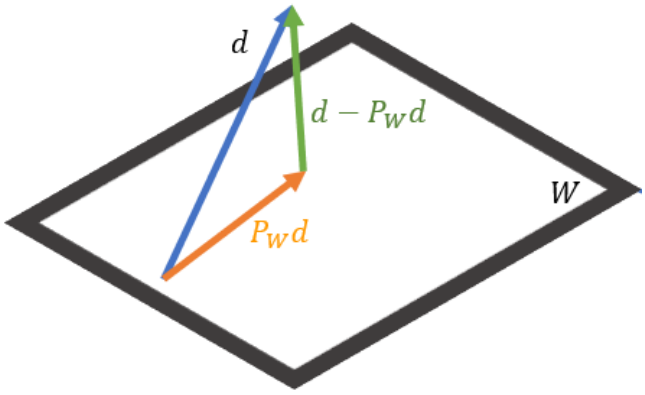

Figure 3.2: The orthogonal projection of d onto W.

Notice that $d - P_W d$ is in the orthogonal complement of W, $W^{\perp}$ and hence $d - P_W d$ is in the nullspace of C^T. In other words, x^* must satisfy the equation

$$C^T(d - Cx^*) = 0, \tag{3.33}$$

which is the same thing as

$$C^T C x^* = C^T d. \tag{3.34}$$

Equation (3.34) is called the **normal equation** and yields the least squares solution

$$x^* = (C^T C)^{-1} C^T d. \tag{3.35}$$

Notice that (3.35) is dependent on the fact that $C^T C$ is invertible. However, $C^T C$ needs not be invertible, or in the context of imaging, $C^T C$ may be huge and difficult to compute its inverse. Another concern is round-off error due to small, nonzero numbers present or when those numbers show up as 0, thus wrecking the process of inverting a matrix [23].

3.4.1 The Singular Value Decomposition and Pseudoinverse

The **singular value decomposition (SVD)** allows for any matrix $C \in \mathbb{R}^{m\times n}$ to be written in the form

$$C = U\Sigma V^T, \tag{3.36}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix whose diagonal elements satisfies $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq \sigma_{r+1} \geq \ldots \sigma_n \geq 0$:

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r & & \\ & & & & \ddots & \\ & & & & & \sigma_n \end{bmatrix}. \tag{3.37}$$

The σ_i's are called the **singular values** of C and are the positive square roots of the eigenvalues of C^TC. Note that σ_r is the smallest nonzero singular value and that the rank of matrix C is equal to the number of positive singular values. The columns u_i of U are called the **left singular vectors**, while the columns v_i of V are called the **right singular vectors**. Since U is orthogonal, then the following holds for the column vectors of U

$$u_i^T u_j = \begin{cases} 0, & i \neq j \\ 1, & i = j, \end{cases}$$

and $U^T = U^{-1}$. A similar argument can be said about the columns of V. The singular value decomposition of C can be expressed as the sum

$$\begin{aligned} C &= U\Sigma V^T \\ &= \begin{bmatrix} u_1 & \ldots & u_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix} \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix} \end{aligned}$$

$$= u_1\sigma_1 v_1^T + \cdots + \sigma_n u_n v_n^T$$
$$= \sum_{i=1}^{n} \sigma_i u_i v_i^T. \tag{3.38}$$

Any matrix can be decomposed this way [37].

An important application of the SVD is that it allows for a generalization of matrix inversion for an any matrix. So for the moment, let's assume that the singular values of C are strictly positive. Thus, the inverse of C is given by

$$C^{-1} = V\Sigma^{-1}U^T$$
$$= \sum_{i=1}^{n} \frac{1}{\sigma_i} v_i u_i^T, \tag{3.39}$$

where $\Sigma^{-1} \in \mathbb{R}^{n\times m}$ is a diagonal matrix. We make note that the diagonal entries of Σ^{-1} are the reciprocals of the diagonal entries of Σ, that is, $\frac{1}{\sigma_i}$. However, this presents a problem as Σ will likely have entries $\sigma_i = 0$. Hence, Σ^{-1} will not exist.

As an alternative, define the matrix $\Sigma^+ \in \mathbb{R}^{n\times m}$ with diagonal entry $\frac{1}{\sigma_1}$ whenever $\sigma_i \neq 0$ and 0 whenever $\sigma_i = 0$. We have

$$\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_1} & & & & & \\ & \ddots & & & & \\ & & \frac{1}{\sigma_r} & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix}. \tag{3.40}$$

This matrix Σ^+ is known as the **pseudoinverse** of Σ [18]. Now we define the pseudoinverse of matrix C to be

$$C^+ = V\Sigma^+U^T. \tag{3.41}$$

Recall that we aim to find the x^* such that solves (3.31) and this was equivalent to finding a solution to the normal equation in (3.34). The pseudoinverse of C gives us the solution to the normal equations as

$$\begin{aligned} x^+ &= C^+ d \\ &= V\Sigma^+ U^T d, \end{aligned} \tag{3.42}$$

and is a optimal solution to (3.31).

3.5 Spectral Filtering

Referring back to the linear system,

$$Cx = d,$$

C is said to be **ill-conditioned** if a small perturbation in d leads to relatively large changes in the solution x. As an alternative definition, C is ill-conditioned if its **condition number** is large. Otherwise, C is said to be **well-conditioned**. The condition number of C, denoted $\kappa(C)$, can be computed as the ratio between the largest and smallest positive singular values of C [23, 30], that is,

$$\kappa(C) = \frac{\sigma_1(C)}{\sigma_r(C)}. \tag{3.43}$$

So let's assume that d is subjugate to noise $e \in \mathbb{R}^m$. Then the linear system we aim to solve is

$$Cx = d, \tag{3.44}$$

with $d = d_{exact} + e$. Applying the SVD, we get the solution (naive)

$$\begin{aligned} x_{naive} &= C^{-1} d \\ &= C^{-1}(d_{exact} + e) \end{aligned}$$

$$= V\Sigma^{-1}U^T d_{exact} + V\Sigma^{-1}U^T e$$
$$= \sum_{i=1}^{n} \frac{u_i^T d_{exact}}{\sigma_i} v_i + \underbrace{\sum_{i=1}^{n} \frac{u_i^T e}{\sigma_i} v_i}_{inverted\ noise} . \tag{3.45}$$

The term, $\frac{u_i^T e}{\sigma_i}$, presents a problem if C is ill-conditioned. As i increases, the σ_i's will decrease, and $\frac{1}{\sigma_i}$'s will increase. This results in the terms $\frac{u_i^T e}{\sigma_i}$ to become large in magnitude and thus amplifying the noise present in the system. This effect will cause the noise term to dominate the computation of the solution x_{naive}.

The effects caused by the division of small singular values must be dampened to control the noise term. One way to do this is through **spectral filtering methods**, also known as **regularization**[22, 23, 30]. The spectral filtering methods work by choosing **filtering factors** ϕ_i such that the solution (filtered or regularized)

$$x_{filt} = \sum_{i=1}^{n} \phi_i \frac{u_i^T d}{\sigma_i} v_i \tag{3.46}$$

produces a desirable solution by reducing the effects of the amplified noise. We now discuss now two spectral filtering methods: the **truncated SVD (TSVD) method** and **Tikhonov regularization**.

3.5.1 Truncated SVD (TSVD)

In the TSVD, we select the k largest positive singular values and approximate C by $C_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$. This is equivalent of saying that $C_k = U_k \Sigma_k V_k$, where U_k and V_k are formed by just by the first k columns of U and V, respectively, and Σ_k is a $k \times k$ diagonal matrix with entries $\sigma_1 \geq \sigma_2 \cdots > \sigma_k > 0$. Note that the rank of C_k equal to $k < r$ and it captures much of the behavior of C [23]. We can interpret the corresponding regularized solution by

$$x_{TSVD} = \sum_{i=1}^{n} \phi_i \frac{u_i^T d}{\sigma_i} v_i,$$

with

$$\phi_i = \begin{cases} 1, & i = 1, \ldots, k \\ 0, & i = k+1 \ldots n. \end{cases} \tag{3.47}$$

The parameter k is known as the **truncation parameter** [30] and determines the number of SVD components maintained in the corresponding regularized solution x_{TSVD}. That is, TSVD enforces an upper bound on the size of the reciprocal of the singular values that are used, thus preventing the division by small σ_i.

Another way to implement the TSVD is to replace σ_i by 0 if $|\sigma_i| \leq \epsilon$, where ϵ is some specified tolerance.

3.5.2 Tikhonov Regularization

In Tikhonov regularization, rather enforcing an upperbound on the singular value, C is modified such that all the singular values are strictly positive. Consider the normal equation

$$(C^T C + \alpha^2 I)x = C^T d, \tag{3.48}$$

where $I \in \mathbb{R}^{n \times n}$ represents the identity matrix and α is known as the **Tikhonov regularization parameter**. Thus, we are interested in solving the minimization problem

$$\begin{aligned} \min_x \left\| \begin{bmatrix} Cx \\ \alpha x \end{bmatrix} - \begin{bmatrix} \tilde{d} \\ 0 \end{bmatrix} \right\|_2^2 &= \min_x \left\| \begin{bmatrix} Cx - d \\ \alpha x \end{bmatrix} \right\|_2^2 \\ &= \{\min_x \|Cx - d\|_2^2 + \alpha^2 \|x\|_2^2\}. \end{aligned} \tag{3.49}$$

Simply inverting $(C^T C + \alpha^2 I)$ yields the result,

$$x_\alpha = (C^T C + \alpha^2 I)^{-1} C^T d. \tag{3.50}$$

However, as stated in Section 3.4, the inverse of the matrix in (3.50) maybe difficult to compute and maybe be subjected to round-off error. So instead, we take the singular value approach to this problem.

Consider the eigendecomposition of $C^TC = VDV^T$, and the SVD of $C = U\Sigma V^T$, where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose entries are the eigenvalues of C^TC. Hence, $(C^TC + \alpha^2 I) = V(D + \alpha^2 I)V^T$, where the matrix $(D + \alpha^2 I)$ is a diagonal matrix whose entries are of the form $\sigma_i^2 + \alpha^2$, where σ_i's are singular values of C.

Using the information above, we compute the solution to (3.49) as follows

$$\begin{aligned} x_\alpha &= (C^TC + \alpha^2 I)^{-1}C^Td \\ &= (C^TC + \alpha^2 I)^{-1}C^Td \\ &= (V(D + \alpha^2 I)V^T)^{-1}(V\Sigma^TU^T)d \\ &= V(D + \alpha^2 I)^{-1}V^TV\Sigma^TU^Td \\ &= V(D + \alpha^2 I)^{-1}\Sigma^TU^T \\ &= \sum_{i=1}^{n}\Big(\frac{\sigma_i^2}{\sigma_i^2 + \alpha^2}\Big)\frac{u_i^Td}{\sigma_i}v_i. \end{aligned} \tag{3.51}$$

Hence, the filter factor is given by,

$$\phi_i = \begin{cases} \frac{\sigma_i^2}{\sigma_i^2+\alpha^2}, & \sigma_i > 0 \\ 0, & \sigma_i = 0. \end{cases} \tag{3.52}$$

Notice that the term $\frac{\sigma_i^2}{\sigma_i^2+\alpha^2} < 1$ also dampens amplified noise associated with large reciprocals values $\frac{1}{\sigma_i}$. In other words, Tikhonov regularization ensures that if we can keep $\|x\|_2$ is small, then noise is kept under control, while at the same making $\|Cx - d\|_2$ small.

The choice of the parameter α is dependent on the singular values of C. In Chapter 5, we will discuss two ways to choose an appropriate α.

Though we will not discuss in full detail, but we can expand the idea of the singular value decomposition and regularization to matrices who have a unitary eigendecomposition, namely BCCB matrices. The naive solution in solution can now be presented by

$$x_{naive} = C^{-1}d \tag{3.53}$$

$$= \tilde{U}^*\Lambda\tilde{U}d \tag{3.54}$$

$$= \sum_{i=1}^{n} \frac{\tilde{u}_i^T d}{\lambda_i}\tilde{u}_i, \tag{3.55}$$

where $\tilde{u_i}$ are basis vectors. The corresponding filtered solution can be given by

$$x_{filt} = \sum_{i=1}^{n} \phi\frac{\tilde{u}_i^T d}{\lambda_i}\tilde{u}_i, \tag{3.56}$$

with filter factor ϕ_i of the forms

$$\phi_i = \begin{cases} 1, & i = 1, \ldots, k \\ 0, & i = k+1 \ldots n. \end{cases} \tag{3.57}$$

and

$$\phi_i = \begin{cases} \frac{|\lambda_i|^2}{|\lambda_i|^2+\alpha^2}, & |\lambda_i| > 0 \\ 0, & |\lambda_i| = 0 \end{cases} \tag{3.58}$$

corresponding to TSVD and Tikhonov regularization methods, respectively.

CHAPTER 4

2D IMAGE RECOVERY

In this chapter, we discuss a popular 2D image reconstruction algorithm known as ℓ_1-*magic* : TV_2, which we will simply refer to it as ℓ_1-*magic* [14]. The authors of [14] attempt to solve the following 2D minimization problem

$$\min_X \text{TV}(X) \text{ subject to } \|\mathcal{A}(X) - B\|_2 \leq \epsilon, \tag{4.1}$$

where $\mathcal{A} : \mathbb{R}^{n\times n}$ is the 2D Sparse Fourier Transformation, $B \in \mathbb{R}^{n\times n}$ is the measured observation of sampled pixels of an image in the k-space and $TV : \mathbb{R}^{n\times n} \mapsto \mathbb{R}$ is a function that computes the total variation of X in the $x-$ and $y-$ directions.

The authors solve (4.1) by recasting the problem into a Second Order Cone Problem (SOCP) and solve a new problem using the interior point method via the log barrier function based from [6]. Before doing so, we must define a few terms.

We define the TV function in (4.1) in order to use Newton's method. The Total Variation of a 2D image is defined as the sum of the finite differences of pixels with their vertical and horizontal neighboring pixels. Mathematically, given an two-dimensional image $X \in \mathbb{R}^{n\times n}$, whose ij^{th} pixel is denoted as X_{ij} and define the operators

$$\mathscr{D}_{h;ij}X = \begin{cases} X_{i,j+1} - X_{ij}, & i < n \\ 0, & i = n \end{cases} \qquad \mathscr{D}_{v;ij}X = \begin{cases} X_{i+1,j} - X_{ij}, & j < n \\ 0, & j = n \end{cases}$$

where $\mathscr{D}_{h;ij}$ and $\mathscr{D}_{v;ij}$ are the horizontal and vertical difference operators, respectively. Furthermore, define the vector $\mathscr{D}_{ijk}X \in \mathbb{R}^2$ as

$$\mathscr{D}_{ij}X = \begin{bmatrix} \mathscr{D}_{h;ij}X \\ \mathscr{D}_{v;ij}X \end{bmatrix}. \tag{4.2}$$

This vector, $\mathscr{D}_{ij}X$, is known as the **discrete gradient** of the image at the ij^{th} pixel, and hence the total variation in (4.1) is defined as follows

$$\mathrm{TV}(X) = \sum_{ij} \sqrt{(\mathscr{D}_{h;ij}X)^2 + (\mathscr{D}_{v;ij}X)^2} = \sum_{ij} \|\mathscr{D}_{ij}X\|_2. \tag{4.3}$$

4.1 Interior Point Method for SOCPs

We can rewrite (4.1) as the SOCP [3, 6, 14]

$$\min_{X,T} \sum_{ij} T_{ij} \text{ subject to } \|\mathscr{D}_{ij}X\|_2 \leq T_{ij} \; i,j = 1,\ldots,n, \tag{4.4}$$

$$\|\mathcal{A}(X) - B\|_2 \leq \epsilon.$$

where $T = [T_{ij}]$ is a $n \times n$ matrix. In order to use the log barrier method, we define the functions

$$f_{T_{ijk}} = \frac{1}{2}(\|\mathscr{D}_{ij}X\|_2^2 - T_{ij}^2) \; i,j, = 1,\ldots,n \tag{4.5a}$$

$$f_\epsilon = \frac{1}{2}(\|\mathcal{A}(X) - B\|_2^2 - \epsilon^2), \tag{4.5b}$$

and embedded them into the objective function via the log-barrier function.

Now applying the log-barrier function and making the functions (4.5a) and (4.5b) implicit into the objective function, we obtain the functional

$$F(s) = \langle c_0, s\rangle - \frac{1}{\tau}\left[\sum_{ijk} \log(f_{T_{ijk}}(s)) + \log(-f_\epsilon(s))\right], \tag{4.6}$$

where τ is an accuracy parameter, $c_0 = \begin{bmatrix} 0_{n^2} \\ 1_{n^2} \end{bmatrix}$ and $s = \begin{bmatrix} x \\ t \end{bmatrix}$, and

$$x := X(:) = \begin{bmatrix} X_{11} \\ \vdots \\ X_{n1} \\ \vdots \\ X_{1n} \\ \vdots \\ X_{nn} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{n^2} \end{bmatrix}, \text{and } t := T(:) = \begin{bmatrix} T_{11} \\ \vdots \\ T_{n1} \\ \vdots \\ T_{1n} \\ \vdots \\ T_{nn} \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ t_{n^2} \end{bmatrix}. \tag{4.7}$$

and the notation 0_{n^2} and 1_{n^2} are used to denote the column vectors of 1's and 0's of length n^2, respectively. The gradient and Hessian of $F(s)$ are computed as the following

$$g_s = c_0 + \frac{1}{\tau}\left[\sum_{ij} \frac{1}{-f_{T_{ijk}}(s)} \nabla f_{T_{ij}}(s) + \frac{1}{-f_\epsilon(s)} \nabla f_\epsilon(s)\right] \tag{4.8}$$

and

$$\begin{aligned} H_z =& \frac{1}{\tau}\left[\sum_{ij} \frac{1}{f_{T_{ij}}(s)^2} \nabla f_{T_{ij}}(s)[\nabla f_{T_{ij}}(s)]^T + \sum_{ij} \frac{1}{-f_{T_{ij}}(s)} \nabla^2 f_{T_{ij}}(s) \right. \\ & \left. + \frac{1}{f_\epsilon(s)^2} \nabla f_\epsilon(s)[\nabla f_\epsilon(s)]^T + \frac{1}{-f_\epsilon(s)} \nabla^2 f_\epsilon(s)\right] \end{aligned}. \tag{4.9}$$

Assuming that s is feasible, the direction Δs along which $F(s)$ is to be approximated is the solution to the following system of linear equations

$$H_s \Delta s = -g_s \tag{4.10}$$

This linear system is reduced and then solved by the linear CG method.

CHAPTER 5

Our Method

Now that we have discussed the ℓ_1-magic algorithm discussed in Chapter 4, we will extend the model and formulate a new method in order to do sparse reconstruction and deconvolution via an alternating minimization scheme with $\ell_1 - \ell_2$ regularization. Consider the following problem

$$\min_{Z,X} \mathrm{TV}(Z) \text{ subject to } \|\mathcal{A}(Z) - B\|_2 + \eta\|\mathcal{C}(X) - Z\|_2 \leq \epsilon, \tag{5.1}$$

where $Z, X \in \mathbb{R}^{n\times n}$ are the blurred and true images, respectively, $\mathcal{A} : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is a Fourier sparsifying linear transformation, $\mathcal{C} : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is the linear convolution transformation associated with the PSF, $B \in \mathbb{R}^{n\times n}$ is the measured observations of sampled pixels of a blurred image in the k-space, TV is a function that computes the 2D Total Variation of Z, and η is a non-negative penalty parameter.

Recall in Chapter 4 that we already defined the discrete gradient (4.2) and total variation (4.3) of 2D image X. Now consider the blurred image of Z, whose ij^{th} pixel is denoted as Z_{ij} and define the operators

$$\mathscr{D}_{h;ij}Z = \begin{cases} Z_{i,j+1} - Z_{ij}, & i < n \\ 0, & i = n \end{cases} \qquad \mathscr{D}_{v;ij}Z = \begin{cases} Z_{i+1,j} - Z_{ij}, & j < n \\ 0, & j = n \end{cases},$$

where $\mathscr{D}_{h;ij}$, $\mathscr{D}_{v;ij}$ are the horizontal and vertical difference operators, respectively. Furthermore, define the vector $\mathscr{D}_{ij}Z \in \mathbb{R}^2$ as

$$\mathscr{D}_{ij}Z = \begin{bmatrix} \mathscr{D}_{h;ij}Z \\ \mathscr{D}_{v;ij}Z \end{bmatrix}, \tag{5.2}$$

and it is the discrete gradient of blurred image Z at the ij^{th} pixel. Hence, the total variation of Z in (5.1) is defined as follows

$$\mathrm{TV}(Z) = \sum_{ij} \sqrt{(\mathscr{D}_{h;ij} Z)^2 + (\mathscr{D}_{v;ij} Z)^2} = \sum_{ij} \|\mathscr{D}_{ij} Z\|_2. \tag{5.3}$$

5.1 Compressed Sensing with Interior Point Method via Log-Barrier Function

We first minimize the variable Z by recasting (5.1) to a SOCP [3, 6, 14]

$$\begin{aligned} &\min_{Z,T} \sum_{ij} T_{ij} \\ &\text{subject to } \|\mathscr{D}_{ij} Z\|_2 \le T_{ij} \ i, j = 1, \ldots, n, \\ &\|\mathcal{A}(Z) - B\|_2 + \eta \|\mathcal{C}(X) - Z\|_2 \le \epsilon. \end{aligned} \tag{5.4}$$

where $T = [T_{ij}]$ is an $n \times n$ matrix. Note that we are now optimizing over two parameters, Z and T. The vectorization of Z and T are defined as

$$z := Z(:) = \begin{bmatrix} Z_{11} \\ \vdots \\ Z_{n1} \\ \vdots \\ Z_{1n} \\ \vdots \\ Z_{nn} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ z_{n^2} \end{bmatrix}, \text{and } t = T(:) = \begin{bmatrix} T_{11} \\ \vdots \\ T_{n1} \\ \vdots \\ T_{1n} \\ \vdots \\ T_{nn} \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ t_{n^2} \end{bmatrix} \tag{5.5}$$

which are the vectorization of Z and T, respectively.

Now we must embedded our constraints into our objective function via the log-barrier function, so we define

$$f_{T_{ij}} = \frac{1}{2}(\|\mathscr{D}_{ij} Z\|_2^2 - T_{ij}^2) \ \ i, j = 1, \ldots, n \tag{5.6a}$$

$$f_\epsilon = \frac{1}{2}(\|\mathcal{A}(Z) - B\|_2^2 + \eta \|\mathcal{C}(X) - Z\|_2^2 - \epsilon^2) \tag{5.6b}$$

We now can use the log barrier function and make the inequality constraints implicit into the objective function using (5.6a) and (5.6b) as follows

$$F(s) = \langle c_0, s\rangle - \frac{1}{\tau}\left[\sum_{ijk} \log(f_{T_{ij}}(s)) + \log(-f_\epsilon(s))\right], \tag{5.7}$$

where τ is an accuracy parameters and $c_0 = \begin{bmatrix} 0_{n^2} \\ 1_{n^2} \end{bmatrix}$, $s = \begin{bmatrix} z \\ t \end{bmatrix}$. The notations 0_{n^2} and 1_{n^2} are used again to denote the column vectors of 1's and 0's of length n^2, respectively. The gradient of $F(s)$, denoted as g_s is expressed as

$$g_s = c_0 + \frac{1}{\tau}\left[\sum_{ij} \frac{1}{-f_{T_{ij}}(s)} \nabla f_{T_{ij}}(s) + \frac{1}{-f_\epsilon(s)} \nabla f_\epsilon(s)\right] \tag{5.8}$$

and the Hessian of $F(s)$,denoted by H_s, is expressed as

$$\begin{aligned} H_s = & \frac{1}{\tau}\left[\sum_{ij} \frac{1}{f_{T_{ij}}(s)^2} \nabla f_{T_{ij}}(s)[\nabla f_{T_{ij}}(s)]^T + \sum_{ij} \frac{1}{-f_{T_{ij}}(s)} \nabla^2 f_{T_{ij}}(s) \right. \\ & \left. + \frac{1}{f_\epsilon(s)^2} \nabla f_\epsilon(s)[\nabla f_\epsilon(s)]^T + \frac{1}{-f_\epsilon(s)} \nabla^2 f_\epsilon(s)\right] \end{aligned}. \tag{5.9}$$

Assuming for a feasible s, the direction Δs along which $F(s)$ is to be approximately minimized is the solution to the following systems of linear equations

$$H_s \Delta s = -g_s. \tag{5.10}$$

In the next two sections we compute the gradient and the Hessian explicitly in order to solve our method.

5.2 Computing g_s

In this section we compute the gradient in (5.8). Note that s is comprised of z and t. This means that we have to take partial derivatives when computing our gradients. In [3], the author computes the gradient in parts and we will do the

same here. We will compute the term $\sum_{ij} \frac{1}{f_{T_{ij}}(s)} \nabla f_{T_{ij}}(s)$ first. Define the row vectors $D_{h;ij}, D_{v;ij}, \in \mathbb{R}^{1\times n^2}$ such that

$$D_{h;ij} z = \mathscr{D}_{h;ij} Z \tag{5.11a}$$

$$D_{v;ij} z = \mathscr{D}_{v;ij} Z. \tag{5.11b}$$

We now can express $f_{T_{ij}}$ as

$$\begin{aligned} f_{T_{ij}} &= \frac{1}{2}\left(\left\| \begin{bmatrix} D_{h;ij} z \\ D_{v;ij} z \end{bmatrix} \right\|_2^2 - T_{ij}^2\right) \\ &= \frac{1}{2}\left(\|D_{ij} z\|_2^2 - T_{ij}^2\right) \;\; i,j = 1,\ldots,n, \end{aligned} \tag{5.12}$$

where $D_{ij} = \begin{bmatrix} D_{h;ij} Z \\ D_{v;ij} Z \end{bmatrix}$. By expanding (3.14) we obtain the expression

$$\begin{aligned} f_{T_{ij}} &= \frac{1}{2}\left(\|D_{ij} z\|_2^2 - T_{ij}^2\right) \\ &= \frac{1}{2}\left((D_{ij} z)^T (D_{ij} z) - T_{ij}^2\right) \\ &= \frac{1}{2}\left(z^T D_{ij}^T D_{ij} z - T_{ij}^2\right) \;\; i,j = 1,\ldots,n. \end{aligned} \tag{5.13}$$

Thus, taking the gradient of $f_{T_{ij}}$ and noting that it is of quadratic form in z we can write

$$\begin{aligned} \sum_{ij} \frac{1}{f_{T_{ij}}(s)} \nabla f_{T_{ij}}(s) &= \sum_{ij} \frac{1}{f_{T_{ij}}(s)} \begin{bmatrix} D_{ij}^T D_{ij} z \\ -T_{ij}\delta_{ij} \end{bmatrix} \\ &= \begin{bmatrix} D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z \\ -F_t^{-1} t \end{bmatrix} \end{aligned} \tag{5.14}$$

where $\delta_{ij} \in \mathbb{R}^{n^2}$ is a vector that is 1 in the ij^{th} entry and 0 elsewhere, $F_t^{-1} = \texttt{diag}(1 \oslash f_T)$, where $f_T \in \mathbb{R}^{n^2}$ contains the $f_{T_{ij}}$ elements listed in lexicographical order according to their ij^{th} indices. The $\oslash$ notation is used to denote element-wise division, i.e. given vectors x and y, $x \oslash y = \frac{x_i}{y_i}$. Finally, the $D_h, D_v, \in \mathbb{R}^{n^2\times n^2}$ are comprised of the row vectors $D_{h;ij}, D_{v;ij} \in \mathbb{R}^{1\times n^2}$, for $i,j = 1,\ldots,n$, that have been vectorized. Based on lexicographical ordering, we can express

$$D_h = B \otimes I_n$$

$$D_v = I_n \otimes B,$$

where

$$B = \begin{bmatrix} -1 & 1 & & & \\ & \ddots & \ddots & & \\ & & -1 & 1 & \\ & & & 0 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

We now compute the terms $\frac{1}{f_\epsilon(s)} \nabla f_\epsilon(s)$. For implementation, we first note that, $\mathcal{A}(Z)$ and $\mathcal{C}(X)$ are equivalent to matrix-vector multiplication, Az and Cx, by formally writing $Az := (\mathcal{A}(Z))(:)$ and $Cx := (\mathcal{C}(X))(:)$. Similarly, we compare the resulting vectors to the vectors $b := B(:)$ and $z := Z(:)$. Now we can expand the equivalent f_ϵ expression as

$$\begin{aligned} f_\epsilon &= \frac{1}{2}(\|Az - b\|_2^2 + \eta \|Cx - z\|_2^2 - \epsilon^2) \\ &= \frac{1}{2}((Az-b)^T(Az-b) + \eta((Cx-b)^T(Cx-b)) - \epsilon^2) \\ &= \frac{1}{2}(z^T A^T A z - 2A^T b z + b^T b + \eta(x^T C^T C x - 2z^T C x + z^T z) - \epsilon^2). \end{aligned} \tag{5.15}$$

Taking the gradient of (5.15), we get

$$\begin{aligned} \nabla f_\epsilon(s) &= \begin{bmatrix} A^T A z - A^T b + \frac{\eta}{2}(-2Cx + 2z) \\ 0_{n^2} \end{bmatrix} \\ &= \begin{bmatrix} A^T(Az - b) - \eta(Cx - z) \\ 0_{n^2} \end{bmatrix} \\ &= \begin{bmatrix} A^T r_1 - \eta r_2 \\ 0_{n^2} \end{bmatrix}, \end{aligned} \tag{5.16}$$

where $r_1 = Az - b$ and $r_2 = Hx - z$ are residual terms and $0_{n^2} \in \mathbb{R}^{n^2}$. Note that A^T is equivalent to applying the tranpose operation of $\mathcal{A}$ that can be denoted as

$$A^T r_1 := (\mathcal{A}^T(R_1))(:),$$

where $\mathcal{A}^T : \mathbb{R}^{n \times n} \mapsto \mathbb{R}^{n \times n}$ and $R_1 = A(Z) - B$. Thus,

$$\frac{1}{f_\epsilon(s)} \nabla f_\epsilon(s) = \frac{1}{f_\epsilon(z)} \begin{bmatrix} A^T r_1 - \eta r_2 \\ 0_{n^2} \end{bmatrix}. \tag{5.17}$$

We are ready to compute g_s for our reconstruction problem. Now substitute the results of (5.14) and (5.17) into (5.8), we get that the gradient of our objective function is

$$\begin{aligned} g_s &= c_0 + \frac{1}{\tau}\left[\sum_{ij} \frac{1}{-f_{T_{ij}}(s)}\nabla f_{T_{ij}}(s) + \frac{1}{-f_\epsilon(s)}\nabla f_\epsilon(s)\right] \\ &= \begin{bmatrix} 0_{n^3} \\ 1_{n^3} \end{bmatrix} + \frac{-1}{\tau}\Big(\begin{bmatrix} D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z \\ -F_t^{-1} t \end{bmatrix} + \frac{1}{f_\epsilon(s)} \begin{bmatrix} A^T r_1 - \eta r_2 \\ 0_{n^2} \end{bmatrix} \Big) \\ &= \frac{-1}{\tau} \begin{bmatrix} D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z + \frac{1}{f_\epsilon(s)}(A^T r_1 - \eta r_2) \\ -\tau 1_{n^2} - F_t^{-1} t \end{bmatrix}. \end{aligned} \tag{5.18}$$

5.3 Computing H_s

We now need to compute the Hessian in (5.9), H_s, but first we need to compute four quantities:

1. $\sum_{ij} \frac{1}{f_{T_{ij}}(s)^2}\nabla f_{T_{ij}}(s)[\nabla f_{T_{ij}}(s)]^T$
2. $\sum_{ij} \frac{1}{-f_{T_{ij}}(s)}\nabla^2 f_{T_{ij}}(s)$
3. $\frac{1}{f_\epsilon(s)^2}\nabla f_\epsilon(s)[\nabla f_\epsilon(s)]^T$
4. $\frac{1}{-f_\epsilon(s)}\nabla^2 f_\epsilon(s)$

We will begin by find quantities 1 and 3 since we already have expressions for the $\sum_{ij} \frac{1}{-f_{T_{ij}}(s)}\nabla f_{T_{ij}}(s)$ and $\frac{1}{-f_\epsilon(s)}\nabla f_\epsilon(s)$. By using expression (5.14), we get

$$\sum_{ij} \frac{1}{f_{T_{ij}}(s)^2}\nabla f_{T_{ij}}(s)[\nabla f_{T_{ij}}(s)]^T = \begin{bmatrix} \hat{z} \\ -F_t^{-1} t \end{bmatrix} \begin{bmatrix} \hat{z}^T & -F_t^{-1} t^T \end{bmatrix}, \tag{5.19}$$

where $\hat{z} = D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z$. Computing the left top block matrix of the resulting matrix is given by

$$\begin{aligned} \hat{z}\hat{z}^T &= (D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z)(D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z)^T \\ &= (D_h^T D_h z + D_v^T D_v z) F_t^{-1} F_t^{-1} (D_h^T D_h z + D_v^T D_v)^T \\ &= B F_t^{-2} B^T, \end{aligned} \tag{5.20}$$

where $B = D_h^T \Sigma_h + D_v^T \Sigma_v$, with $\Sigma_h = \texttt{diag}(D_h z)$ and $\Sigma_v = \texttt{diag}(D_v z)$. Computing the top right entry gives us

$$\begin{aligned} -\hat{z} F_t^{-1} &= -(D_h^T F_t^{-1} D_h z + D_v^T F_t^{-1} D_v z) F_t^{-1} t^T \\ &= -B \tilde{T} F_t^{-2} \end{aligned} \tag{5.21}$$

where $\tilde{T} = \texttt{diag}(t)$ and $F_t^{-2} = \texttt{diag}(1 \oslash f_t^2)$. The remaining blocks are obvious, thus we get

$$\sum_{ij} \frac{1}{f_{T_{ij}}(s)^2} \nabla f_{T_{ij}}(s) [\nabla f_{T_{ij}}(s)]^T = \begin{bmatrix} B F_t^{-2} B^T & -B \tilde{T} F_t^{-2} \\ -F_t^{-2} \tilde{T} B^T & F_t^{-2} \tilde{T}^2 \end{bmatrix}. \tag{5.22}$$

Now we examine how to compute the term $\frac{1}{f_\epsilon(s)^2} \nabla f_\epsilon(s) [\nabla f_\epsilon(s)]^T$. Recall the expression we obtained from (5.17), and plug it in to get

$$\begin{aligned} \frac{1}{f_\epsilon(s)^2} \nabla f_\epsilon(s) [\nabla f_\epsilon(s)]^T &= \frac{1}{f_\epsilon(s)^2} \begin{bmatrix} A^T r_1 - \eta r_2 \\ 0 \end{bmatrix} \begin{bmatrix} (A^T r_1 - \eta r_2)^T & 0 \end{bmatrix} \\ &= \frac{1}{f_\epsilon(s)^2} \begin{bmatrix} (A^T r_1 - \eta r_2)(r_1^T A - \eta r_2^T) & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned} \tag{5.23}$$

We now calculate the final terms of our Hessian: $\sum_{ij} \frac{1}{-f_{T_{ij}}(s)} \nabla^2 f_{T_{ij}}(s)$ and $\frac{1}{-f_\epsilon(s)} \nabla^2 f_\epsilon(s)$. Noting that the Hessian is the Jacobian of the gradient, we get that

$$\sum_{ij} \frac{1}{-f_{T_{ij}}(s)} \nabla^2 f_{T_{ij}}(s) = \begin{bmatrix} D_h^T (-F_t^{-1}) D_h + D_v^T (-F_t^{-1}) D_v & 0 \\ 0 & F_t^{-1} \end{bmatrix}, \tag{5.24}$$

and

$$\frac{1}{-f_\epsilon(s)} \nabla^2 f_\epsilon(s) = \frac{1}{-f_\epsilon(s)} \begin{bmatrix} A^T A + \eta I_n & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.25}$$

Note that $A^T A z + \eta := ((A^T (A(Z))) + \eta I_n)(:)$ but for consistency we use $A^T A + \eta I_n$ in writing for the Hessian. We now have all the expression to compute H_s

$$\begin{aligned} H_s = &\frac{1}{\tau} \left[\sum_{ij} \frac{1}{f_{T_{ij}}(s)^2} \nabla f_{T_{ij}}(s) [\nabla f_{T_{ij}}(s)]^T + \sum_{ij} \frac{1}{-f_{T_{ij}}(s)} \nabla^2 f_{T_{ij}}(s) \right. \\ &\left. + \frac{1}{f_\epsilon(s)^2} \nabla f_\epsilon(s) [\nabla f_\epsilon(s)]^T + \frac{1}{-f_\epsilon(s)} \nabla^2 f_\epsilon(s) \right] \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{\tau}\Big(\begin{bmatrix} BF_t^{-2}B^T & -B\tilde{T}F_T^{-2} \\ -F_t^{-2}\tilde{T}B^T & F_t^{-2}\tilde{T}^2 \end{bmatrix} + \begin{bmatrix} D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v & 0 \\ 0 & F_t^{-1} \end{bmatrix} \\
&+ \begin{bmatrix} f_\epsilon(z)^{-2}(A^Tr_1 - \eta r_2)(r_1^TA - \eta r_2^T) & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -f_\epsilon(s)^{-1}(A^TA + \eta I_n) & 0 \\ 0 & 0 \end{bmatrix} \Big) \\
&= \frac{1}{\tau} \begin{bmatrix} H_{11} & -B\tilde{T}F_t^{-2} \\ -F_t^{-2}\tilde{T}B^T & F_t^{-2}\tilde{T}^2 + F_t^{-1} \end{bmatrix}, \qquad (5.26)
\end{aligned}
$$

where

$$
\begin{aligned}
H_{11} &= BF_t^{-2}B^T + D_h^T(-F_t^{-1})D_h + D_v^T(-F_t^{-1})D_v + f_\epsilon(z)^{-2}(A^Tr_1 - \eta r_2)(r_1^TA - \eta r_2^T) \\
&- f_\epsilon(z)^{-1}(A^TA + \eta I_n) \qquad (5.27)
\end{aligned}
$$

As in [3], we will simplify the notation here. So define

$$\Sigma_{12} = -\tilde{T}F_t^{-2}, \qquad (5.28a)$$

$$\Sigma_{22} = F_t^{-2}\tilde{T} + F_t^{-2}, \qquad (5.28b)$$

and thus

$$H_s = \frac{1}{\tau} \begin{bmatrix} H_{11} & B\Sigma_{12} \\ \Sigma B^T & \Sigma_{22} \end{bmatrix}. \qquad (5.29)$$

5.4 Reduce the System

In the previous section, we found the expressions for g_s and H_s stated in (5.18) and (5.29). We now use Newton's method to solve a large scale system, so substituting (5.18) and (5.29) in (5.10) yields

$$
\begin{aligned}
\begin{bmatrix} H_{11} & B\Sigma_{12} \\ \Sigma B^T & \Sigma_{22} \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta t \end{bmatrix} &= \begin{bmatrix} D_h^TF_t^{-1}D_hz + D_v^TF_t^{-1}D_vz + \frac{1}{f_\epsilon(s)}(A^Tr_1 - \eta r_2) \\ -\tau - F_t^{-1}t \end{bmatrix} \\
&:= \begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}. \qquad (5.30)
\end{aligned}
$$

We begin to reduce the system of equation by eliminating Δt in equation (5.30)

$$\Sigma_{12}B^T\Delta z + \Sigma_{22}\Delta t = \nu_2 \implies \Delta t = \Sigma_{22}^{-1}(\nu_2 - \Sigma_{12}B^T\Delta z). \qquad (5.31)$$

Using this expression results in system reductions as the following

$$\begin{aligned}
& H_{11}\Delta z + B\Sigma_{12}\Delta t = \nu_1 \\
\implies & H_{11}\Delta z + B\Sigma_{12}[\Sigma_{22}^{-1}(\nu_2 - \Sigma_{12}B^T\Delta z)] = \nu_1 \\
\implies & H_{11}\Delta z + B\Sigma_{12}\Sigma_{22}^{-1}\nu_2 - B\Sigma_{12}\Sigma_{22}B^T\Delta z = \nu_1 \\
\implies & (H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T)\Delta z = \nu_1 - B\Sigma_{12}\Sigma_{22}^{-1}\nu_2 \\
\implies & \hat{H}_{11}\Delta z = \hat{\nu_1}, \qquad (5.32)
\end{aligned}$$

where $\hat{H}_{11} = H_{11} - B\Sigma_{12}^2\Sigma_{22}^{-1}B^T$ and $\hat{\nu_1} = \nu_1 - B\Sigma_{12}\Sigma_{22}^{-1}\nu_2$. In order to make the coding of the problem easier to implement we get a more simplified expression for $\hat{H}$; refer to [3].

5.5 Deblurring Step via Regularization

After each log-barrier iteration that has been solved by Newton's method, the approximation of $n \times n$ compressed and blurred image Z is updated. We then use this updated Z to solve the least squares problem using the singular value decomposition

$$\min_x \|Cx - z\|_2, \qquad (5.33)$$

to update the approximation of the $n \times n$ ground truth image X. Recall that the matrix $C \in \mathbb{R}^{N\times N}$ is the blurring matrix associated with the unwanted PSF and its structure is dependent on the boundary conditions imposed with $N = n^2$. It is a known fact that the blurring matrix is highly ill-conditioned [23, 30]. That is, the naive solution we get by the singular value decomposition

$$x_{naive} = C^{-1}z = V\Sigma^{-1}U^Tz \qquad (5.34)$$

maybe contaminated by unwanted noise due to division of small singular values, $\frac{1}{\sigma_1}$.

Filtering or regularization methods are used to reduce the effects of inverted noise by enforcing regularity conditions on the solution. The two regularization methods we focus on are

- TSVD, and
- Tikhonov regularization.

The degree of regularization enforced by these two techniques are governed by a regularization parameter as discussed in Chapter 3. This regularization parameter must be chosen carefully and we will discuss potential ways to find an appropriate regularization parameter later on.

The singular value decomposition can be used for general problems or for problems that have a Kronecker structure (i.e., separable blurs) [36, 56] with the filtered solution of the form

$$x_{filt} = \sum_{i=1}^{N} \phi_i \frac{u_i^T z}{\sigma_i} v_i, \tag{5.35}$$

while the unitary spectral decomposition can be used when we can use FFT algorithms, such as the case for BCCB matrices:

$$x_{filt} = \sum_{i=1}^{N} \phi_i \frac{\tilde{u}_i^T z}{\lambda_i} \tilde{u}_i, \tag{5.36}$$

with filter factor ϕ_i of the forms

$$\phi_i = \begin{cases} 1, & i = 1, \ldots, k \\ 0, & i = k+1 \ldots N \end{cases} \quad \text{and} \quad \phi_i = \begin{cases} \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2}, & \sigma_i > 0 \\ 0, & \sigma_i = 0, \end{cases}$$

for TSVD and Tikhonov regularization, respectively, for the SVD, and

$$\phi_i = \begin{cases} 1, & i = 1, \ldots, k \\ 0, & i = k+1 \ldots N \end{cases} \quad \text{and} \quad \phi_i = \begin{cases} \frac{|\lambda_i|^2}{|\lambda_i|^2 + \alpha^2}, & |\lambda_i| > 0 \\ 0, & |\lambda_i| = 0 \end{cases}$$

for the unitary spectral decomposition.

For the remainder of the discussion, we discuss the SVD to simplify things but can also be applied to the unitary spectral decomposition. So now we rewrite the filtered solution as

$$x_{filt} = V\Phi\Sigma^{-1}z, \tag{5.37}$$

where $\Phi \in \mathbb{R}^{N\times N}$ is a diagonal matrix whose entries are the filter factors ϕ_i for the particular regularization methods. That is, the diagonal entries are of Φ are of the form

$$\Phi_{ii} = \begin{cases} 1, & i = 1, \ldots, k \\ 0, & i = k+1 \ldots N \end{cases} \tag{5.38}$$

for TSVD and

$$\Phi_{ii} = \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \tag{5.39}$$

for Tikhonov regularization. Observe that

$$\begin{aligned} x_{filt} &= V\Phi\Sigma^{-1}U^T z \\ &= V\Phi\Sigma^{-1}U^T(z_{exact} + e) \\ &= V\Phi\Sigma^{-1}U^T z_{exact} + V\Phi\Sigma^{-1}U^T e \\ &= V\Phi\Sigma^{-1}U^T C x_{exact} + V\Phi\Sigma^{-1}U^T e \\ &= V\Phi\Sigma^{-1}U^T(U\Sigma V^T)x_{exact} + V\Phi\Sigma^{-1}U^T e \\ &= V\Phi V^T x_{exact} + V\Phi\Sigma^{-1}U^T e, \end{aligned}$$

and therefore the error of the filtered solution is given by

$$x_{exact} - x_{filt} = (I_N - V\Phi V^T)x_{exact} - V\Phi\Sigma^{-1}U^T e. \tag{5.40}$$

Here, the error consist of two types of contributions, the **regularization error** and **perturbation error** [30].

- The regularization error, represented by the first component, $(I_N - V\Phi V^T)x_{exact}$, is caused by the using a regularized inverse matrix $V\Phi\Sigma^{-1}U^T$ (rather than that of the original inverse matrix ($C^{-1} = V\Sigma^{-1}U^T$). The matrix $V\Phi V^T$ describes the mapping between x_{exact} and x_{filt}. If the diagonal entries of Φ are relatively small, then the regularization error becomes large. On the contrary, if Φ approaches the identity matrix I_N, then the regularization error becomes small.
- The perturbation error, $V\Phi\Sigma^{-1}U^T$, consists of the inverted noise and filtered noise. If $\Phi = 0$, then the perturbation error is zero. When the most of the entries of Φ are small, then the inverted noise is heavily dampened and perturbation error is small.

The change of Φ and the two types of errors are dependent on the changes in the regularization parameter. The relationship can be summed up as follows

1. When many filter factors ϕ_i are close to 1, then the regularization error is small, while the perturbation error is large. Thus, our reconstructed image is **undersmoothed**.
2. When too few filters factors ϕ_i are close 1, then the perturbation error is small, while the regularization error is large. Thus, our reconstructed image will be **oversmoothed**.

A proper choice of the truncation parameter k or Tikhonov regularization parameter α is needed to balance these two types of error.

5.5.1 Parameter of Choice Method

We present two different parameter choice methods: **user experimentation** and the **generalized cross validation (GCV)** [26, 30, 41]. In user experimentation, the user chooses the regularization parameter that they feel results in the best

reconstruction. However, this is a tedious process as one is selecting the parameter ad hocly.

The idea of the GCV is that, if one omits a data value, then a good value of the parameter α should be able to predict the missing data point. According to the authors of [26], this can be achieved by choosing the parameter that minimizes the GCV function

$$G(\alpha) = \frac{\|(I_N - CV\Phi\Sigma^{-1}U^T)z\|_2^2}{(\text{trace}(I_N - CV\Phi\Sigma^{-1}U^T))^2}. \tag{5.41}$$

Note that we abuse notation here for parameter α as we are using it as the regularization parameter for both TSVD and Tikhonov, with $\alpha = \frac{1}{k}$ where k is the TSVD cutoff. The author of [30] gives a formulation of the numerator

$$\|I_N - CV\Phi\Sigma^{-1}U^T)z\|_2^2 = \|b - Cx_{filt}\|_2^2 = \sum_{i=1}^{N}((1-\phi_i)u_i^T z)^2 \tag{5.42}$$

and that the denominator can be evaluated easily as the trace of matrix is the sum of its main diagonal elements

$$\begin{aligned}
\text{trace}(I_N - CV\Phi\Sigma^{-1}U^T) &= \text{trace}(I_N - U\Sigma V^T V\Phi\Sigma^{-1}U^T) \\
&= \text{trace}(U(I_N - \Phi)U^T) \\
&= \text{trace}(I_N - \Phi) \\
&= N - \sum_{i=1}^{N}\phi_i.
\end{aligned} \tag{5.43}$$

Hence,

$$G(\alpha) = \frac{\|(I_{N\times N} - CV\Phi\Sigma^{-1}U^T)z\|_2^2}{(\text{trace}(I_{N\times N} - CV\Phi\Sigma^{-1}U^T))^2} = \frac{\sum_{i=1}^{N}((1-\phi_i)u_i^T z)^2}{N - \sum_{i=1}^{N}\phi_i}. \tag{5.44}$$

In particular, we want to find the minimum of the functions,

$$G(k) = \frac{1}{N-k}\sum_{i=k+1}^{N}(u_i^T z)^2, \tag{5.45}$$

for $k = 1, 2, \ldots, N-1$, and

$$G(\alpha) = \frac{\sum_{i=1}^{N} \left(\frac{u_i^T z}{\sigma_i^2 + \alpha^2}\right)^2}{\sum_{i=1}^{N} \left(\frac{1}{\sigma_i^2 + \alpha^2}\right)^2}, \tag{5.46}$$

for TSVD and Tikhonov regularization, respectively. The built-in MATLAB function `fminbnd` is used to find the optimal parameter (k or α).

The proposed algorithm can be summarized in Figure 5.1 and Algorithm 5.1 below.

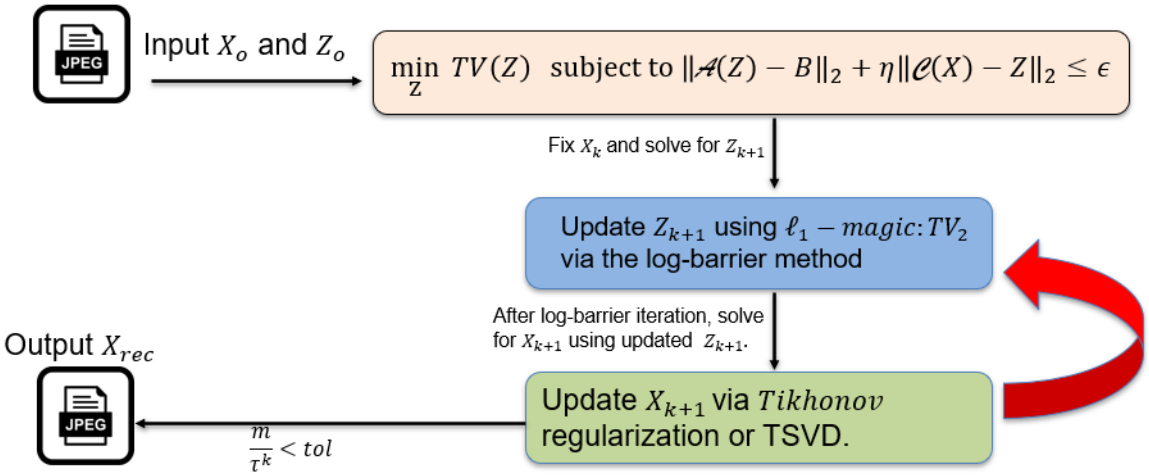

Figure 5.1: The workflow diagram of our alternating minimization scheme.

Algorithm 5.1 Compressive Deconvolution via $\ell_1 - \ell_2$ regularization

Input: feasible starting point s_0, τ_0, $\mu > 0, \theta > 0, \eta > 0, n_{max} > 0, x_0$.

$k = 0$, $\tau = \tau_0$, $\eta_0 = \eta$

while $\frac{m}{\tau} > \epsilon$ **do** ▷ $\frac{m}{\tau}$ is the dual gap of (5.1)

    Step 1: Update Compressed Blurry Image z_{k+1} using interior point method via log-barrier function.

    Find $s_k = \min_s F(s)$ by Newton's Method, where $F(s)$ is defined in (5.7):

    $n_{stop} = 0, k = 0$

    while $n_{stop} = 0$ **do**

        Compute g_s and H_s

        Find Newton Step Δs by using CG to solve $H_s \Delta s = -g_s$.

        $\lambda^2 = g_s^T H_s^{-1} g_s$

        BLS: Find appropriate β

        $s_k = s_k + \beta \Delta s$

        $k = k + 1$

        $n_{stop} = (\frac{\lambda^2}{2} < \epsilon)$ or $(k > n_{max})$

    end while

    Extract z_{k+1} from s_{k+1}.

    Step 2: Update Deblurred Image x_{k+1} using z_{k+1}.

    Solve $x_{k+1} = \min_x \|Cx - z_{k+1}\|_2$ using Tikhonov regularization or TSVD.

    Parameter Choice: Implement GCV implementation or user experimentation to find appropriate α.

    $\tau = \mu\tau$

    $\eta = \theta\eta$

end while

Output: Approximation to ground truth image $\tilde{X}$.

CHAPTER 6

Numerical Results

This chapter discusses some results of the recovery of a 2D brain image from subsampled and blurred data in $k-space$. To evaluate the quality of the reconstructed images, we used three imaging metrics: (1) the structural similarity (SSIM) index [51]; (2) the peak signal-to-noise ratio; and (3) the human perception.

6.1 Image Reconstruction Metrics

The **peak signal-to-noise ratio (PSNR)** is the ratio between the maximum possible power (value) of a signal and the power of the noise that affects the quality of its representation. Due to signals having a wide dynamic range, the PSNR is typically expressed in terms of logarithmic decibel scale (dB). A higher PSNR typically indicates a better image reconstruction. However, there have been cases where images with higher PSNR values look worse than those with a lower PSNR value, and ultimately, human perception is the deciding factoring [3, 35].

Given two $m \times n$ 2D images X and Y, we first compute the mean square error (MSE) as

$$MSE = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} ((X(i,j) - Y(i,j))^2. \tag{6.1}$$

The PSNR is defined as

$$PSNR = 10 \log_{10} \Big(\frac{I_{max}^2}{MSE} \Big), \tag{6.2}$$

where I_{max}^2 is the maximum possible pixel value of the image on Y. Implementation of the PSNR can be done in MATLAB using the built-in function `psnr`.

The other metric used to evaluate our reconstruction is the **structural similarity (SSIM) index**, and it improves upon the deficiencies of PSNR as the PSNR does not take into account any biological factors of the human vision system [51, 52]. The SSIM index is used to evaluate the similarity of the intensity, contrast, and structure between the ground-truth image (fully sampled) and the image reconstructed by our algorithm using compressed and blurred samples. The SSIM is defined as follows

$$SSIM(X,Y) = \frac{(2\mu_X\mu_Y + c_1)(2\sigma_{XY} + c_2)}{(\mu_X^2 + \mu_Y^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)}, \tag{6.3}$$

where X and Y denote the ground truth and reconstructed images, respectively, μ's are the averages of the respective image intensities, σ's are the variances, and $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are two variables that stabilizes the division by the denominator. L is the dynamic range of the pixel values with $k_1 = 0.01$ and $k_2 = 0.03$. Two identical grayscale images will have a maximum SSIM value of 1, while a value 0 indicates they are two completely different images. The code for SSIM can be download from `www.mathworks.com/matlabcentral/answers/9217-need-ssim-m-code`.

The last metric is the human perception. No matter what the PSNR or SSIM tell us, we get the final say on how the reconstruction compares to the ground truth. This will become more evident as we at look some of the images we recovered from our method.

6.2 Performance of Reconstruction

The imaging data that is presented in this section comes from the Radiology 229: MRI Signals and Sequences course at Standford University provided Drs. Daniel Ennis and Brian Hargreaves [1]. It is a 2D brain image that was collected in 2016 and contains 512×512 complex data that was fully sampled in the k-space.

[1] Their dataset can be found at `https://github.com/mribri999/MRSignalsSeqs`

We shall blur this imaging data with a 17×17 Gaussian PSF with standard deviation 7. Next, we will sample the measurements in k-space by implementing a pratical 2D sampling scheme [24, 28, 49]. This scheme is known as the **radial line sampling scheme** which mainly focus on sampling the center frequencies and some other adjacent frequencies [20]. The scheme was used in the algorithms presented in [3] and [14]. These compressed measurements will be used as the initial starting image for our algorithm. Figure 6.1 gives a representation of the process.

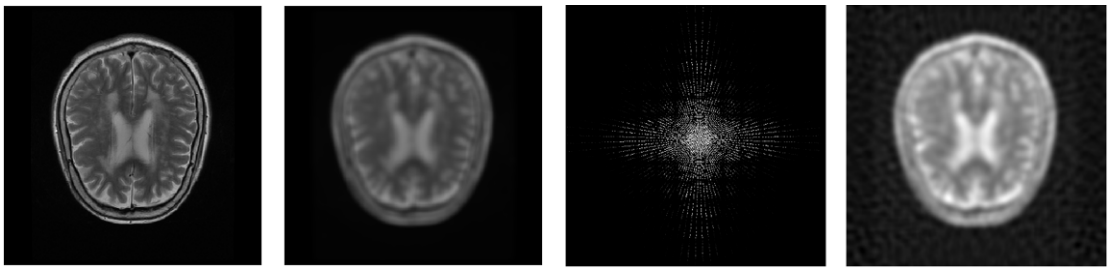

Figure 6.1: Left: Reference Image of Brian. Mid-Left: Reference Image: Brain Image blurred by a 17×17 Gaussian PSF with standard deviation 7. Mid-Right: Radial line sampling in k-space. Right: Compressed Blurred Observations.

The image deblurring step will consist of solving the least-squares approximation using TSVD and Tikhonov regularization. The regularization parameter will chosen via the GCV implementation and through user experimentation.

6.2.1 Reconstruction Results using Zero Boundary Conditions

Here we take a look at the reconstruction results of our algorithm when we assume that the Gaussian PSF has blurred the brain image under the zero boundary condition. Additionally, the blurred image has been sampled in the k-space at 22%. The reconstruction results of sampling at 22% rate are shown in Figure 6.2.

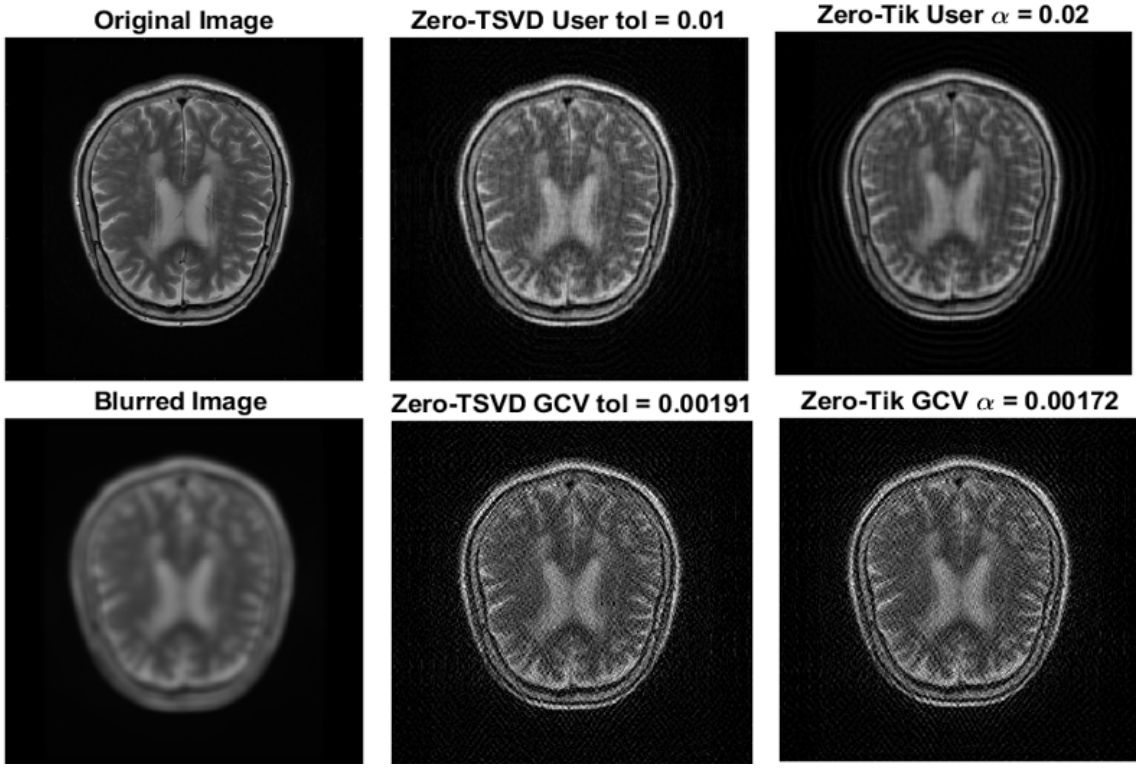

Figure 6.2: Reconstruction of brain image using TSVD and Tikhonov regularization under zero boundary condition at a 22% sampling rate. First Column: Original Image and blurred image. Second column: Image recovery using TSVD with singular value tolerance chosen by the user and through GCV implementation. Third Column: Image recovery using Tikhonov regularization with α chosen by the user and through GCV implementation.

Figure 6.2 shows a better reconstruction under the chose of the user input for the regularization parameters rather than what GCV provides. This is backed up by SSIM and PSNR values computed in Table 6.1. However, we do note that the user's input is not efficient as it can be daunting and time-consuming to find an appropriate regularization parameter to get a good reconstruction.

Regularization	Input	SSIM	PSNR
TSVD	User = .01	.71	30.52
	GCV = .00191	.22	21.25
Tikhonov	User = .02	.81	30.96
	GCV = .00172	.26	21.72

Table 6.1: Reconstruction Evaluation under Zero Boundary Conditions at 22% Sampling.

6.2.2 Reconstruction Regularization using Periodic Boundary Conditions

We now assume that the brain image has been blurred by the Gaussian PSF under the periodic boundary conditions at the same sample rating as in the zero boundary case. The results of reconstruction are shown by Table 6.2 and Figure 6.3. Here again, Tikhonov regularization under user choice produces the best results visually. However, there is a slight decrease in the PSNR value compared to that in the truncated SVD.

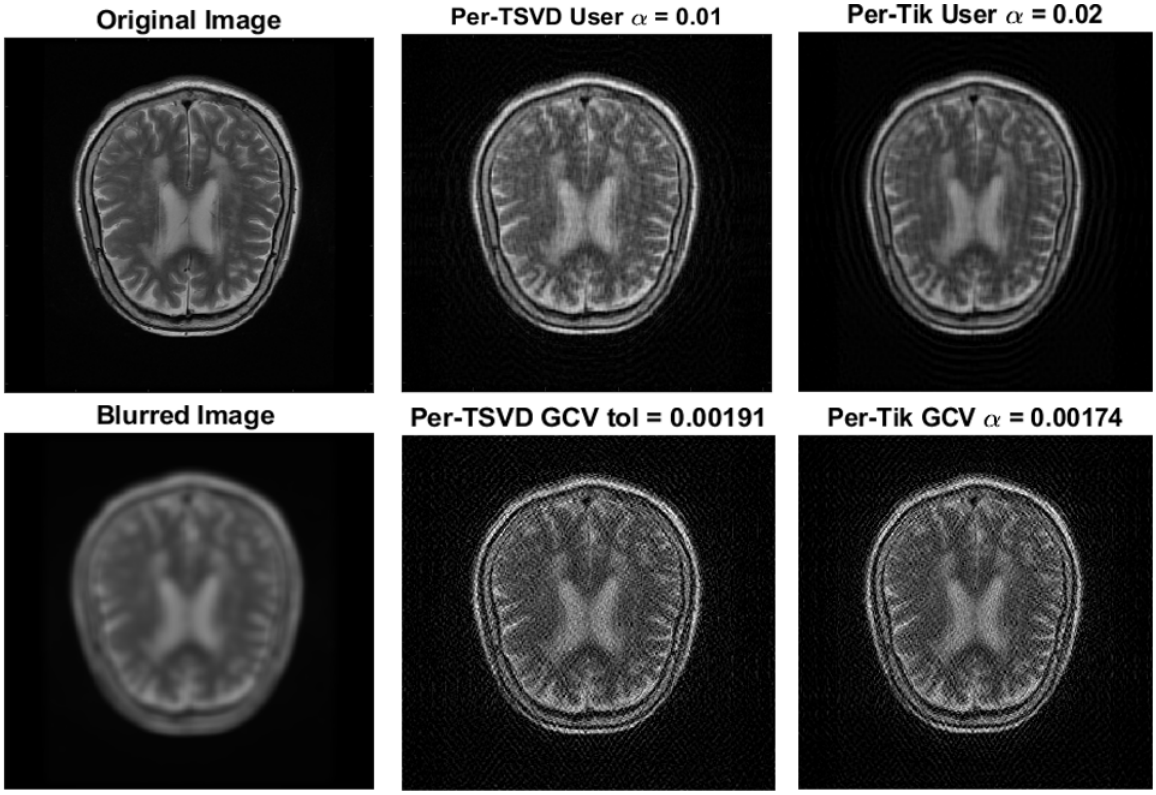

Figure 6.3: Reconstruction of brain image using TSVD and Tikhonov regularization under periodic boundary condition at a 22% sampling rate. First Column: Original Image and blurred image. Second column: Image recovery using TSVD with singular value tolerance chosen by the user and through GCV implementation. Third Column: Image recovery using Tikhonov regularization with α chosen by the user and through GCV implementation.

Regularization	Input	SSIM	PSNR
TSVD	User = .01	.78	31.21
	GCV = .00191	.21	21.16
Tikhonov	User = .02	.82	31.02
	GCV = .00174	.27	21.66

Table 6.2: Reconstruction Evaluation under Periodic Boundary Conditions at 22% Sampling.

Overall Observations: We have shown that it is possible to recover the brain image blurred by the Gaussian PSF using our algorithm with a low sampling

rate of 22%. Based on our imaging metrics, we can see that we can produce better SSIM values using periodic boundary conditions than that of the zero boundary conditions, while the PSNR values are split yet negligibly close. We make note that the implementation of GCV was producing regularization parameters that gave poor quality results physical and numerically. This may be due to the behavior of the GCV function is very flat to the minimizer, thus this could cause potential problem for numerical method to find an opitmal regularization parameter [30]. The user choice method produces great reconstruction result, however it is a tedious method and not ideal.

6.2.3 Extra: Motion Blur Recovery

This is an extra reconstruction of the brain that has been affected by motion blur of length 20 and with a 45° angle of motion in the counter clockwise direction. We assume periodic boundary conditions at 22% sampling and using truncated unitary spectral decomposition and Tikhonov regularization in the Fourier domain. The results are shown in Table 6.3 and Figure 6.4.

Regularization	Input	SSIM	PSNR
TSVD	User = .029	.69	31.78
	GCV = .00289	.69	31.81
Tikhonov	User = .02	.74	32.05
	GCV = .0162	.73	31.8

Table 6.3: Reconstruction Evaluation under Periodic Boundary Conditions at 22% Sampling.

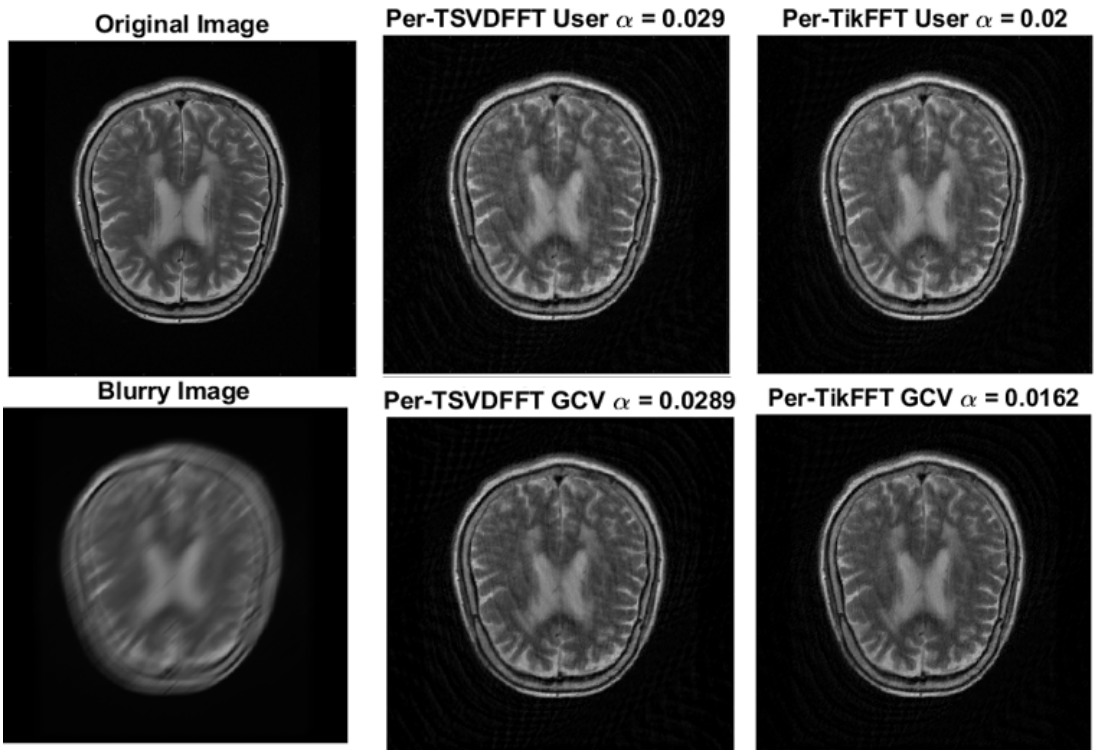

Figure 6.4: Reconstruction of brain image using truncated unitary eigendecomposition and Tikhonov regularization under periodic boundary condition at a 22% sampling rate. First Column: Original Image and blurred image. Second column: Image recovery using TSVD with singular value tolerance chosen by the user and through GCV implementation. Third Column: Image recovery using Tikhonov regularization with α chosen by the user and through GCV implementation.

CHAPTER 7

Conclusion and Future Works

We have presented a novel non-blind compressive deconvolution alternating minimization scheme that joints ℓ_1-magic with spectral filtering via truncated singular value decomposition and Tikhonov regularization. This work shows that this method is effective on blurred and highly compressed (sampling at a rate of 22%) MRI imaging data. Thus, this should represent a reduction in time for data acquisition by almost a factor of 5 when patients come in for an MRI scan. Additionally, we can reduce the effects of blurring that are an inherent characteristic in MRI caused by involuntary patient movement due to discomfort, inherent physical artifacts that are a response of implants or medical devices to the magnetic field, or by noise caused by hardware and software.

When implementing the method, we saw that implementing Tikhonov regularization was superior in deblurring the image by presenting higher SSIM and PSNR values than the truncated SVD. To find a value for the regularization parameter for both Tikhonov and the truncated SVD, we implemented two parameter of choice methods: user experimentation and the GCV. However, there are flaws to both methods: (1) the GCV parameter choice method is faster but often led to lower SSIM and PSNR values, while (2) user experimentation can produce better results than the GCV, it suffers from being too cumbersome and time-consuming to search for an appropriate regularization parameter ad hocly.

As it stands now, the method we implemented assumes that we know what PSF caused the blurring of the image. However, in most MRI imaging systems, the

PSF is typically unknown. In the future, we want to extend this method to perform blind compressive deconvolution, and we plan to refine this method for 3D images. Additionally, we plan to incorporate machine learning and create an AI method that performs auto compressive deconvolution for respiratory motion related blurring in 3T MRI imaging that affects extraction and evaluation of metabolic information about fat infiltration and muscle atrophy in the shoulder.

REFERENCES

[1] Amizic, B., Spinoulas, L., Molina, R., and Katsaggelos, A., Compressive Blind Image Deconvolution. *IEEE Transactions on Image Processing 22*, 2013: 3994-4006.

[2] Apostol, M., *Mathematical Analysis*. Addison-Welsey, 2nd edition, 1974.

[3] Au, M., Three dimensional image reconstruction (3DIRECT) of sparse signal with MRI application. *Ph.D. thesis, The University of Texas at Arlington*, 2016.

[4] Axler, S., *Linear Algebra Done Right*. Undergraduate Texts in Mathematics, 2014.

[5] Blackledge, J., *Digital Image processing*. Horwood Publishing, vol: ISBN:1-898563-49-7, 2005.

[6] Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge University Press, 2004.

[7] Burden, R. L. and Faires, J. D., *Numerical Analysis*. Cengage Learning, 2011.

[8] Campisi, P., and Egiazarian, K., *Blind Image Deconvolution: Theory and Applications*. CRC Press, 2007. https://doi.org/10.1201/9781420007299

[9] Elvidge, J., *Applications of Nuclear Magnetic Resonance Spectroscopy in Organic Chemistry*, 1970.

[10] Candès, E., The restricted isometry property and its implications for compressed sensing, *Comptes Rendus Mathematique*, vol. 346, no. 2, pp. 589–592, 2008.

[11] Candès, E.,Romberg, J., and Tao, T., Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[12] Candès, E., Tao, T., and Romberg, J., Stable signal recovery from incomplete and inaccurate measurements, *Comm. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[13] Candès, E. and Tao, T., Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[14] Candès, E. and Romberg, J., *l1*-Magic: Recovery of sparse signals via convex programming. Available online[1]. 2005.

[15] Candès, E. and Wakin, M., An Introduction To Compressive Sampling, *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21-30, 2008, doi: 10.1109/MSP.2007.914731.

[16] Chen, S., Basis pursuit, *Ph.D. dissertation*, CA, 1995.

[17] Chen, Z., Basarab, A. and Kouamé, D., Compressive Deconvolution in Medical Ultrasound Imaging, *IEEE Transactions on Medical Imaging*, vol. 35, no. 3, pp. 728-737, 2016, doi: 10.1109/TMI.2015.2493241

[18] Cheney, E. W., and Kincaid, D., *Numerical Mathematics and Computing*. Brooks/Cole, 2020.

[19] Chowdhury, M., Qin, J., and Lou, Y., Non-blind and Blind Deconvolution under Poisson Noise using Fractional-order Total Variation, *Journal of Mathematical Imaging and Vision*, 2020.

[20] Dale, B. M., Lewin, J. S., and Duerk, J. L., Optimal design of k-space trajectories using a multi-objective genetic algorithm. *Magn. Reson. Med.*, 52(4):831–841, 2004.

[21] Egiazarian, K., Foi, A., and Katkovnik, V., Compressed sensing image reconstruction via recursive spatially adaptive filtering, *IEEE International Conference on Image Processing*, vol. 1, pp. 549–552, 2007.

[1]https://statweb.stanford.edu/ candes/software/l1magic/downloads/l1magic.pdf

[22] Engl, H. W., Hanke, M. and Neubauer, M., *Regularization of inverse problems*. Kluwer, 1996.

[23] Freeman, T., *The Mathematics of Medical Imaging: A Beginner's Guide*. Springer, 2nd Edition, 2015.

[24] Glover, G. H. and Pauly, J. M., Projection reconstruction technique for reduction of motion effects in MRI. *Magn. Reson. Med.*, 28:275–289, 1992.

[25] Golub, G. E. and Von Loan, C. F., *Matrix Computations*. The Johns Hopkins University Press, 1989.

[26] Golub, G. H., Heath, M., and Wahba, G., Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

[27] Gray, R. M., Toeplitz and Circulant Matrices: A Review.*Foundations and Trends in Communications and Information Theory*: Vol. 2: No. 3, pp 155-239, 2006.

[28] Greiser, A. and von Kienlin, M., Efficient k-space sampling by density-weighted phase-encoding. *Magn. Reson. Med.*, 50(6):1266–1275, 2003.

[29] Gonzalez, R. and Woods, R., *Digital Image Processing*. 2nd Edition. Prentice Hall, 2004.

[30] Hansen, P.C., Nagy, J., and O'Leary, D., Deblurring Images: Matrices, Spectra, and Filtering. *Philadelphia: SIAM, Society for Industrial and Applied Mathematics*. Available online [2]. 2006.

[31] Hauser, K., Multivariate Newton's Method and Quasi-Newton methods. Algorithms for Optimization and Learning CSB553. Retrieved from Duke University. Delivered January 25, 2012.

[32] Havsteen, I., Ohlhues, A., Madsen, K. H., Nybing, J. D., Christensen, H., and Christensen, A. Are Movement Artifacts in Magnetic Resonance Imag-

[2]https://epubs.siam.org/doi/book/10.1137/1.9780898718874?mobileUi=0

ing a Real Problem?-A Narrative Review. *Frontiers in neurology*, 8, 232. 2017. https://doi.org/10.3389/fneur.2017.00232

[33] Heiland, S., From A as in aliasing to Z as in zipper: artifacts in MRI. *Clin Neuroradiol*, 2008; 1:25-36.

[34] Hestenes, M. R., and Stiefel, E., Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, Vol. 49, p. 409, 2006.

[35] Huynh-Thu, Q. and Ghanbari, M., Scope of validity of PSNR in image/video quality assessment. *Electronics Letters*, 44(13):800 – 801, 2008.

[36] Kamm, J. and Nagy, J.M., Kronecker Product and SVD Approximations in Image Restoration. *Linear Algebra and Its Applications*, vol. 284, no. 1–3, 1998, pp. 177–92. Crossref, doi:10.1016/s0024-3795(98)10024-1.

[37] Leon, J. S., *Linear Algebra with Applications*.6th Edition. 2002.

[38] Lustig, M., Donoho, D., and Pauly, J. M., Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magn Reson Med*, vol. 58,pp. 1182–1195. 2007.

[39] Moratal, D., Vallés-Luch, A., Martí-Bonmatí, L., and Brummer, M., k-space tutorial: an MRI educational tool for a better understanding of k-space. *Biomedical imaging and intervention journal*, 2008, 4(1), e15. https://doi.org/10.2349/biij.4.1.e15

[40] Nagy, J.G. and O'Leary, D.P., Fast iterative image restoration with a spatially varying PSF, Proc.*SPIE 3162, Advanced Signal Processing: Algorithms, Architectures, and Implementations VII*, 1997; doi: 10.1117/12.279513

[41] Nguyen, T. and Chung, J., Learning regularization parameters for general-form Tikhonov. *Inverse Problems 33:7*, 074004. 2007.

[42] Nocedal, J. and Wright, S. J., *Numerical Optimization*. Springer. 2nd edition. 2006.

[43] Pruessmann, K. P., Weiger, M., Scheidegger, M. B., and Boesiger, P., Sense: sensitivity encoding for fast mri., *Magnetic Resonance in Medicine*, vol. 42,no. 5, pp. 952–962, 1999.

[44] Richardson, W. H., Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55–59, 1972.

[45] Roggemann, M. C. and Welsh, B., *Imaging Through Turbulence*. CRC Press, Boca Raton, FL, 1996.

[46] Smith, S. W., The Scientist and Engineer's Guide to Digital Signal Processing. *California Technical Pub*, 1997.

[47] Sun, W. and Yuan, Y-X., *Optimization Theory and Methods: Nonlinear Programming*. Springer. 2006.

[48] Tibshirani, R., Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, 1996, pp. 267–88. Crossref, doi:10.1111/j.2517-6161.1996.tb02080.x.

[49] Tsai, C. M. and Nishimura, D. G., Reduced aliasing artifacts using variable density k-space sampling trajectories. *Magn. Reson. Med.*, 43(3):452–458, 2000.

[50] Varah, J. M., Pitfalls in the numerical solution of linear ill-posed problems. *SIAM J. Sci. Stat. Comput.*, 4:164–176, 1983.

[51] Wang, Z., Bovik, A.C., Sheikh, H.R., and Simoncelli, E.P., Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process. 13*, 600–612, 2004.

[52] Wang, Z., Simoncelli, E. P., and Bovik, A. C., Multiscale structural similarity for image quality assessment,*The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003, pp. 1398-1402 Vol.2, doi: 10.1109/ACSSC.2003.1292216.

[53] Wiener, N., *Extrapolation, interpolation, and smoothing of stationary time series: With engineering applications*. MIT Press, 113(21):1043–54, 1949.

[54] Yang, Jungang, Tian Jin, Chao Xiao, and Xiaotao Huang., Compressed Sensing Radar Imaging: Fundamentals, Challenges, and Advances" *Sensors* 2019, no. 14: 3100. https://doi.org/10.3390/s19143100

[55] Ye, J.C., Compressed sensing MRI: a review from signal processing perspective. *BMC biomed eng*1, 8, 2019. https://doi.org/10.1186/s42490-019-0006-z

[56] Zhang, Y., Li, Y., Wang, S., and Wang, C. -X., Image Restoration Based on the Convolution Kernel Matrix Reconstructed by Kronecker Product,*IEEE 5th International Conference on Signal and Image Processing (ICSIP)*, 2020, pp. 185-189, doi: 10.1109/ICSIP49896.2020.9339404.

BIOGRAPHICAL STATEMENT

Talon Johnson was born in Dallas, Texas, in 1994. He earned a Bachelor of Science in Mathematics at Morehouse College in Atlanta, GA of May 2016. As an undergraduate, he participated in research projects through the National Science Foundation sponsored Math Summer Program In Research And Learning (Math SPIRAL) and the National Institute for Mathematical and Biological Synthesis (NIMBioS). He served as the vice president of Pi Mu Epsilon Delta Chapter and has received a Best Presentation Award at the Joint Mathematics Meeting in 2015. After receiving his Bachelors degree, Talon entered the mathematical doctoral program at the University of Texas at Arlington (UTA), where he worked under the supervision of Jianzhong Su on projects related to math biology and computational neurosciences. Talon was awarded with the Michael B. and Wanda G. Ray Graduate Fellowship and the Graduate Assistance in Areas of National Need (GAANN) Fellowship. During his graduate studies, he became a member of the UTA American Mathematical Society and the Society of Industrial and Applied Mathematics, co-president of the College of Science Black Graduate Student Association, and was a mentor for the Army Educational Outreach Program.