NUMERICAL OPTIMIZATION METHODS FOR IMAGE RESTORATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Byunggyoo Kim
December 2002

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Michael A. Saunders
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Walter Murray

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yinyu Ye

Approved for the University Committee on Graduate Studies:

Abstract

Image restoration is the process of approximating an original image from an observed blurred and noisy image. It is an inverse problem whose goal is to use the observed image and any knowledge of the statistics of the noise and blur to produce an image that is better in a quantitative sense. When the blurs are linear and spatially invariant, they can be represented as a convolution in an integral equation. Discretizing the convolution produces a large-scale algebraic system.

In the absence of information about noise, we can model the image restoration problem as a nonnegatively constrained least squares problem (NNLS). We then find that enforcing nonnegativity is computationally nontrivial, and that direct methods for NNLS problems are not appropriate because of their high memory and work requirements. Expectation Maximization (EM) methods have been used widely as iterative methods because they require moderate memory, but they converge slowly. Thus, faster iterative algorithms are needed. We propose a primal-dual interior method for the NNLS problem, with an iterative solver computing each search direction approximately. Primal-dual methods need a moderate number of iterations regardless of the problem size. (There may be millions of variables for a moderate size image.)

Observing that a large percentage of pixels in the original image are essentially zero, we present some methods for taking advantage of sparsity in the solution of NNLS problems by interior methods. We also explore the use of simplex-like nonlinear methods such as the reduced-gradient method, which prove to work well for extremely sparse images (in applications such as astronomy and signal decomposition). Numerical examples include a satellite image and a star image obtained by the Hubble Space Telescope.

Acknowledgments

First of all, I would like to thank my advisor, Professor Michael A. Saunders, for his invaluable support, encouraging words and endurance for my inexpertness throughout this study. He has always been generous and a guiding light when I lost research direction. Without his help and support, I could not finish my education at Stanford. My reader, Professor Walter Murray, introduced me to my advisor for the first time and provided invaluable suggestions to improve the quality of this dissertation. Another reader, Professor Yinyu Ye, contributed his essential optimization expertise and ideas. He also gave me a chance to do more research on semidefinite optimization. I would also like to thank Professor B. Curtis Eaves, who was my academic advisor for several years, and Professor Eric Darve in Mechanical Engineering, for being on my oral committee.

I would like to thank many friends at SOL (the Systems Optimization Laboratory) for sharing our office so patiently. I will miss the companionship of Alexis Collomb, Maureen Doyle, Michael Friedlander, Kien-Ming Ng, Uday Shanbhag, and Che-Lin Su.

Very special thanks are due to Professor James Nagy at Emory University for providing the motivation and software and test problems needed for this research, and to Carolyn Sale for working hard to improve the readability of this dissertation.

Many thanks also to Jeonghoon Mo and his wife Mijung Park. Jeonghoon has been very supportive during my stay at Stanford. Having such a good friend is definitely a great fortune to me.

Finally, I dedicate this work to my wife, Yooshin Lee, for her unconditional love and support. Her tremendous help and encouragement are greatly appreciated. Sincerest thanks also to my parents and parents-in-law, who gave me so much motivation and support. Thanks are not enough for them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we discuss the image degradation process and the difference between local filters and global filters. We also discuss the organization of the thesis.

1.1 What is Image Restoration?

Image restoration (or reconstruction) finds an approximation as close as possible in some measure to the original image from a blurred and noisy image. The image restoration problem is known as an ill-posed problem, because small changes in the data can cause arbitrarily large changes in the results. The goal of our study is to provide a fast and numerically stable algorithm to solve the image restoration problem.

An image is a two-dimensional light-intensity function, denoted by $f(x, y)$, where the value (or amplitude) of f at spatial coordinates (x, y) gives the intensity (brightness) of the image at that point. The degradation process is modeled as a blurring operator H that, together with an additive noise term $\eta(x, y)$, operates on an input image $f(x, y)$ to produce a degraded image $g(x, y)$. Figure 1.1 shows the relationship between an input image $f(x, y)$ and the output image $g(x, y)$ with blurring operator H and η.

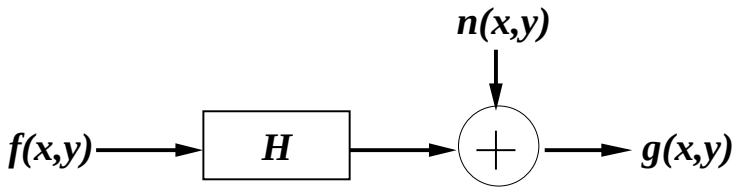

Figure 1.1: Image degradation process

The blurring operation, which is linear and spatially invariant, can be expressed as a convolution. Most image restoration methods are based on the convolution applied to the *whole* image. In other words, image restoration can be viewed as the process of obtaining an approximation of $f(x, y)$, given $g(x, y)$ and knowledge of the degradation in the form of H, and possibly some information about the noise. Noise in imaging is usually additive or multiplicative, but we deal only with additive noise. We can use one of the common noise models (Gaussian, Laplacian and uniform) if we know the nature of the noise. We do not assume any distribution of the noise in this thesis, but our least squares model is best suited to Gaussian noise.

The image restoration problem can be modeled algebraically in several different ways depending on the criteria used to measure the quality of the restored image and the availability of information about noise. When we express the image restoration problem as a mathematical model, the algebraic system is so huge that direct methods would require too much storage and time. Thus, previous studies have tried to develop efficient iterative algorithms to find approximations to the original image [24, 33, 32, 48, 18, 4]. One realistic mathematical model for image restoration is a nonnegatively constrained least squares problem (the NNLS problem). The least squares model is justified by the fact that the least squares estimator is unbiased and has minimum variance when the noises are uncorrelated and all have zero means and the same variance [2, 25]. The least squares formulation without nonnegativity constraints on the variables is much easier to solve compared to the one with nonnegative constraints, but its solution is poor. Enforcing nonnegativity constraints on solutions to least squares systems introduces a high computational load, although it is not difficult to describe the mathematics.

Sparsity is inherent in certain cases, such as telescope images of the universe. We observed that for some images, a huge number of pixel values in the image solution were essentially zero, which means the original images are very sparse. We develop special methods to take advantage of sparsity in images.

1.2 Local vs. Global Filter

Local filters, which includes the median filter, linear average filter, max filter, and min filter, use information from the local neighborhood of a pixel to restore that pixel [17]. In contrast, global filters use information from the entire image to restore each pixel. Most image degradations have global effects; the blurring caused by optical limitations within an imaging device has a point spread function (PSF), which is a blurred image of a single

point source of light: The PSF is infinite in extent, and the power spectrum of additive noise covers the entire frequency domain evenly. Therefore, local filters alone are not sufficient to restore an image. Two widely used global filters are the inverse and the Wiener filter, both of which can be described as an algebraic system that finds a solution using the whole image. These filters have a disadvantage; they are quite slow. It takes several hours for a moderate sized image on a reasonably fast machine to generate acceptable results. For this reason, our study explores the use of fast *iterative* algorithms for global filters.

1.3 Measure of Image Quality

To compare results, we need a measure for image quality. However, image quality is highly subjective—it depends on the person who is viewing the image. One widely used measure is a residual norm or a mean squared error (MSE). One weakness of this measure is that it depends strongly on the scaling of variables despite the fact that the image is invariant to scaling. If we know the true solution as we do for test problems, we can compute the relative error. The relative error avoids the sensitivity to scaling. We compare results for the same image only because one image with a larger relative error may look better than another image with a smaller relative error. We use the 2-norm of a residual vector, relative error, and the distribution of the solution as the performance measure.

1.4 Organization of the Thesis

In Chapter 2, we review the degradation model through a matrix representation of a 2-D convolution, where the blurring matrix H is Block Toeplitz with Toeplitz blocks (**BTTB**). We focus on the computation of matrix-vector products Hx and $H^T y$ because they are the essential part of iterative methods. We can do these matrix-vector operations very efficiently using matrices that are Block Circulant with Circulant Blocks (**BCCB**), the circulant extension of **BTTB**.

Chapter 3 covers the mathematical model for image restoration. First, we consider the unconstrained least squares problem, which is known as an inverse filter. The inverse filter works reasonably well if the noise is negligible. We briefly give some methods for establishing an inverse filter. Then we discuss constrained image restoration, including nonnegatively constrained image restoration and the Wiener filter.

In Chapter 4, we investigate different problems for nonnegatively constrained image restoration. Nonnegative least squares (NNLS), maximum likelihood (ML) and Chebyshev

problems are presented. We also study numerical methods for these problems.

Chapters 5, 6 and 7 discuss numerical methods for NNLS. First, Chapter 5 discusses the scaled steepest descent (SSD) method in detail. The SSD method works well in the early iterations, but the performance is very poor as iterations continue. By and large, SSD is not recommended, but we can take advantage of the intermediate solutions from SSD for the initial points for other methods that behave well near the solution.

In Chapter 6, we focus on primal-dual interior methods for problems with a separable convex objective function. Our experience shows that interior methods, which give solutions containing many small nonzero values, are especially good for gray-scale images, although they work reasonably well for color images as well. We also present some numerical results using PDSCO [41], which implements a primal-dual interior method for problems that have a separable convex objective function and linear constraints.

Chapter 7 describes simplex-like methods for problems with nonlinear objective functions. In particular, we applied the reduced-gradient method in MINOS [29, 30]. This should work well for images arising from fields like astronomy, where the image is extremely sparse and only a small number of nonzero values have an impact on the quality of the image.

Chapter 8 covers ML, ℓ_1, and Chebyshev problems. The ML model has been widely used in medical imaging, where Poisson noise is adequate. Chebyshev and ℓ_1 approximation are given as an alternative to least squares problems because they can be converted into linear programs, for which interior methods are already established.

Chapter 9 summarizes numerical results on test problems and discusses advantages and disadvantages of each method we cover in this dissertation.

Finally, Chapter 10 presents summaries and conclusions about our study, and a description of future work.

Chapter 2

Background

The background material in this chapter summarizes the approach described in the book by Gonzalez [17]. We express the degradation model as a 2-D convolution in an integral form first, then we discretize the 2-D convolution. The discretization boils down to a huge algebraic system with a special structure. To handle the large-scale system, we treat the coefficient matrix as an operator.

2.1 Degradation Model

To establish a model for the degradation of an image, we use the term *image* to refer to a two-dimensional light-intensity function, denoted by $f(x, y)$. As light is a form of energy, $f(x, y)$ should be nonnegative in reality. The degradation process is modeled as an operator H together with a noise $\eta(x, y)$, which operate on an input image $f(x, y)$ to produce a degraded image $g(x, y)$. The observed image, a blurred and noisy copy of f, can be represented as

$$g(x, y) = H[f(x, y)] + \eta(x, y).$$

The operator H is linear if

$$H[k_1 f_1(x, y) + k_2 f_2(x, y)] = k_1 H[f_1(x, y)] + k_2 H[f_2(x, y)],$$

where k_1 and k_2 are constants and $f_1(x, y)$ and $f_2(x, y)$ are any two input images. Linearity means both additivity and homogeneity. Additivity means that if H is a linear operator, the response to a sum of two inputs is equal to the sum of the two responses. Homogeneity indicates that the response to a constant multiple of any input is equal to the response to that input multiplied by the same constant. An operator H is said to be *space (or position)*

invariant if

$$H[f(x-\alpha, y-\beta)] = g(x-\alpha, y-\beta)$$

for any $f(x,y)$ and any α and β. This definition states that the response at any point in the image depends only on the value of the input at that point, not on the position of the input. Our discussion is limited to a linear and space invariant H. The advantage of linear, space invariant processes is that they make the extensive tools of linear system theory available for the solution of the image restoration problem. Even if nonlinear and space variant techniques are more general and usually more accurate, they introduce computational difficulties.

We can now express the degradation process algebraically. We can write $f(x,y)$ in the following form using the impulse function δ:

$$f(x,y) = \int\int f(\alpha,\beta)\ \delta(x-\alpha, y-\beta)\ d\alpha\ d\beta.$$

Then,

$$\begin{aligned} g(x,y) &= H\left[\int\int f(\alpha,\beta)\ \delta(x-\alpha, y-\beta)\ d\alpha\ d\beta\right] + \eta(x,y) \\ &= \int\int H[f(\alpha,\beta)\ \delta(x-\alpha, y-\beta)]\ d\alpha\ d\beta\ + \eta(x,y) \\ &= \int\int f(\alpha,\beta)\ H[\ \delta(x-\alpha, y-\beta)]\ d\alpha\ d\beta + \eta(x,y), \end{aligned} \tag{2.1}$$

since H is a linear operator and $f(\alpha,\beta)$ is independent of x and y. The term

$$h(x,\alpha,y,\beta) = H[\ \delta(x-\alpha, y-\beta)] \tag{2.2}$$

is called the *impulse response* of H. The impulse becomes a point of light, so $h(x,\alpha,y,\beta)$ is also called the *point spread function* (PSF). This name is based on the optical phenomenon that occurs when the impulse corresponds to a point of light and an optical system responds by blurring (spreading) the point, with the degree of blurring determined by the quality of the optical device. Substituting (2.2) into (2.1) yields

$$g(x,y) = \int\int f(\alpha,\beta)\ h(x,\alpha,y,\beta)\ d\alpha\ d\beta + \eta(x,y),$$

which is called a *superposition* (or *Fredholm*) *integral of the first kind.* It states that if the response of H to an impulse is known, the response to any input $f(\alpha,\beta)$ can be calculated by means of the point spread function. In other words, a linear system H is completely

characterized by its PSF. Since H is space invariant, we obtain

$$H[\delta(x-\alpha, y-\beta)] = h(x-\alpha, y-\beta).$$

This indicates that the impulse response simply shifts with movements of the impulse and does not change its shape. Thus, the linear degradation model becomes

$$g(x,y) = \iint f(\alpha,\beta)\, h(x-\alpha, y-\beta)\, d\alpha\, d\beta + \eta(x,y), \tag{2.3}$$

which is the convolution integral.

To be suitable for computer processing, an image function $f(x,y)$ must be digitized both spatially and in amplitude. Thus, we can approximate a continuous image $f(x,y)$ by an $n \times n$ matrix:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \cdots & f(0,n-1) \\ f(1,0) & f(1,1) & f(1,2) & \cdots & f(1,n-1) \\ \vdots & & & \ddots & \vdots \\ f(n-1,0) & f(n-1,1) & f(n-1,2) & \cdots & f(n-1,n-1) \end{bmatrix}. \tag{2.4}$$

Although the matrix can be rectangular depending on the sampling method, we use the square matrix for simplicity. To make the mathematics easier, we represent an image as a vector $f \in \mathbf{R}^{n^2}$ by stacking column by column:

$$f = \mathbf{vec}(f(x,y)),$$

where the **vec** operation makes a matrix into a vector by stacking the columns of the matrix. We discuss this process in detail in Section 2.3.1, where we treat x and y as either a discrete or a continuous variable depending on the context. For more details about the degradation process using the PSF, refer to Gonzalez [17] and Pratt [39].

2.2 2-D Discrete Convolution

Now we consider the discrete degradation model through discretization by formulating $f(x,y)$ and $h(x,y)$ as discrete arrays of size of $n \times n$ as shown in (2.4). The discrete version of the convolution

$$g(x,y) = \iint f(\alpha,\beta)\, h(x-\alpha, y-\beta)\, d\alpha\, d\beta$$

is

$$g(x,y) = \sum_{\alpha=0}^{n-1} \sum_{\beta=0}^{n-1} f(\alpha,\beta) h(x-\alpha, y-\beta),$$

where $h(x,y)$ is considered to be periodic with period $N = 2n-1$ in the x and y directions. That is,

$$h(x,y) = 0 \quad \text{for } n \le x \le N-1 \text{ or } n \le y \le N-1,$$

and

$$\begin{aligned} h(x,y) &= h(x',y) \quad \text{whenever} \quad x = x' \mod N, \\ h(x,y) &= h(x,y') \quad \text{whenever} \quad y = y' \mod N, \end{aligned}$$

where mod is the *modulo* operator. In matrix notation, $g = \mathbf{H}f$, where f and g are vectorized images and

$$\mathbf{H} = \begin{bmatrix} H_0 & H_{-1} & H_{-2} & \cdots & H_{-n+1} \\ H_1 & H_0 & H_{-1} & \cdots & H_{-n+2} \\ H_2 & H_1 & H_0 & \cdots & H_{-n+3} \\ \vdots & & & \ddots & \vdots \\ H_{n-1} & H_{n-2} & \cdots & H_1 & H_0 \end{bmatrix}. \tag{2.5}$$

Each block H_j is

$$H_j = \begin{bmatrix} h(0,j) & h(-1,j) & h(-2,j) & \cdots & h(-n+1,j) \\ h(1,j) & h(0,j) & h(-1,j) & \cdots & h(-n+2,j) \\ h(2,j) & h(1,j) & h(0,j) & \cdots & h(-n+3,j) \\ \vdots & & & \ddots & h(-1,j) \\ h(n-1,j) & h(n-1,j) & \cdots & h(j,1) & h(0,j) \end{bmatrix}.$$

The first column of H_j is the j-th column of $h(x,y)$ and H_j is Toeplitz, that is, H_j is constant along its diagonals. $\mathbf{H}$ is called **BTTB** (block Toeplitz with Toeplitz blocks) because it is Toeplitz blockwise and each block is Toeplitz. The Toeplitz property of $\mathbf{H}$ arises directly from the space invariance of the degradation operator H from Section 2.1. The structure of $\mathbf{H}$ is determined by the properties of the degradation operator H, and this structure plays an important role in computing the matrix-vector product. For example, the convolution

for 2×2 images can be written as follows:

$$\begin{bmatrix} g_{00} \\ g_{10} \\ g_{01} \\ g_{11} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{-1,0} & h_{0,-1} & h_{-1,-1} \\ h_{1,0} & h_{0,0} & h_{1,-1} & h_{0,-1} \\ h_{0,1} & h_{-1,1} & h_{0,0} & h_{-1,0} \\ h_{1,1} & h_{0,1} & h_{1,0} & h_{0,0} \end{bmatrix} \begin{bmatrix} f_{00} \\ f_{10} \\ f_{01} \\ f_{11} \end{bmatrix}.$$

Here $h_{0,0}$ is the center of the given PSF. If the given PSF is not in this order, it must be reordered so that $h_{0,0}$ is placed at the $(1,1)$ position. Figure 2.1 shows the general nonzero structure of $\mathbf{H}$ when the size of the image is 8×8. Each block is banded, as is the full BTTB matrix itself. Kamm and Nagy [21] elaborate on the structure of banded BTTB matrices from the PSF. We shall utilize the banded BTTB nonzero pattern in this study.

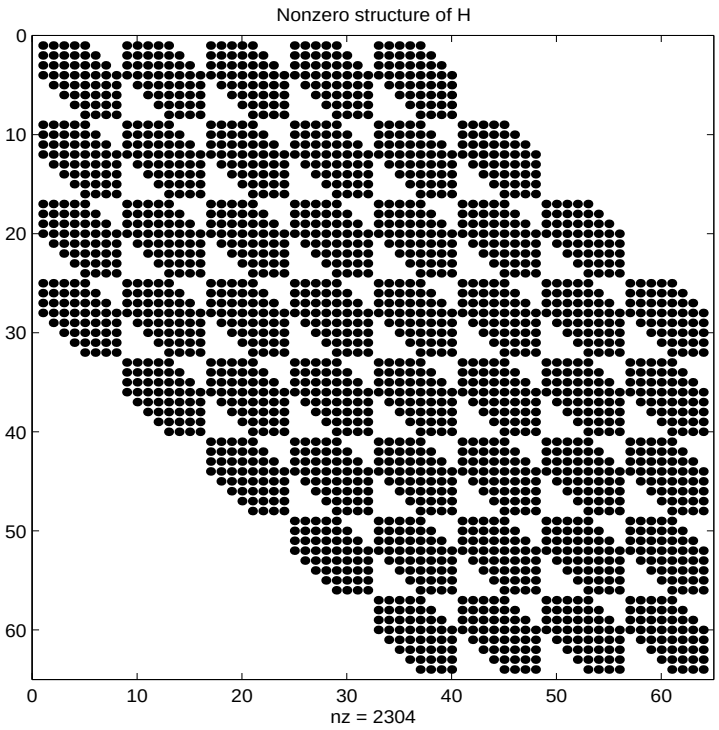

Figure 2.1: General nonzero structure of a **BTTB** matrix $\mathbf{H}$

Although the algebraic expression of the degradation model looks simple, a direct solution to get f would incur a tremendous computational cost for images of practical size. For example, if $n = 256$, $\mathbf{H}$ is of size $256^2 \times 256^2$, *i.e.* 65536×65536. Saving $\mathbf{H}$ explicitly would require about 35 gigabytes if we store data as double precision floating-point numbers. Thus, it is not practical to save $\mathbf{H}$ explicitly. Instead, we should rely on iterative

methods in which $\mathbf{H}$ is regarded as an *operator* for obtaining products $\mathbf{H}x$ and $\mathbf{H}^T y$ cheaply for arbitrary vectors x and y. However, the complexity of this problem can be reduced considerably by taking advantage of an extended form of $\mathbf{H}$ that has circulant properties. We can do the matrix-vector operations with the extended $\mathbf{H}$ quickly by using a discrete 2-D Fourier transform and its inverse. We extend the discrete functions $f(x,y)$ and $h(x,y)$ to be periodic with some period N. It is known that $N \geq n+n-1 = 2n-1$ to avoid interference from adjacent periods [17]. We set $N = 2n$ for convenience. For two digitized $n \times n$ images $f(x,y)$ and $h(x,y)$, we form $N \times N$ periodic extensions $f_e(x,y)$ and $h_e(x,y)$:

$$f_e(x,y) = \begin{cases} f(x,y) & 0 \leq x \leq n-1 \quad \text{and} \quad 0 \leq y \leq n-1 \\ 0 & n \leq x \leq N-1 \quad \text{or} \quad n \leq y \leq N-1 \end{cases}$$

and similarly for $h_e(x,y)$. In matrix notation,

$$f_e(x,y) = \begin{bmatrix} f(x,y) & 0_{n\times n} \\ 0_{n\times n} & 0_{n\times n} \end{bmatrix}, \quad h_e(x,y) = \begin{bmatrix} h(x,y) & 0_{n\times n} \\ 0_{n\times n} & 0_{n\times n} \end{bmatrix}.$$

Since $f_e(x,y)$ and $h_e(x,y)$ are periodic extensions, both $f_e(x,y)$ and $h_e(x,y)$ are periodic with period N in the x and y directions. That is,

$$\begin{aligned} f_e(x,y) &= f_e(x',y) \quad \text{whenever} \quad x = x' \mod N, \\ f_e(x,y) &= f_e(x,y') \quad \text{whenever} \quad y = y' \mod N. \end{aligned}$$

We find the discrete formulation of the degraded model by discretizing the equation:

$$g_e(x,y) = \sum_{\alpha=0}^{N-1} \sum_{\beta=0}^{N-1} f_e(\alpha,\beta) h_e(x-\alpha, y-\beta).$$

For notational convenience, f and g are overloaded and interpreted as either a vector or a matrix of appropriate size depending on the context. This discrete convolution can be expressed in the matrix-vector form $g_e = \mathbf{C} f_e$, where f, g, and η are $\mathbf{R}^{N^2}$ vectors and

$$\mathbf{C} = \begin{bmatrix} C_0 & C_{-1} & C_{-2} & \cdots & C_{-N+1} \\ C_1 & C_0 & C_{-1} & \cdots & C_{-N+2} \\ C_2 & C_1 & C_0 & \cdots & C_{-N+3} \\ \vdots & & & \ddots & \vdots \\ C_{N-1} & C_{N-2} & \cdots & C_1 & C_0 \end{bmatrix}.$$

Each block C_j is a Toeplitz matrix

$$C_j = \begin{bmatrix} h(0,j) & h(-1,j) & h(-2,j) & \cdots & h(-N+1,j) \\ h(1,j) & h(0,j) & h(-1,j) & \cdots & h(-N+2,j) \\ h(2,j) & h(1,j) & h(0,j) & \cdots & h(-N+3,j) \\ \vdots & & & \ddots & h(-1,j) \\ h(N-1,j) & h(N-1,j) & \cdots & h(j,1) & h(0,j) \end{bmatrix}. \tag{2.6}$$

The first column of C_j is the j-th column of $h_e(x,y)$. Each partition of $\mathbf{C}$ is a *Toeplitz* matrix C_j in (2.6). We can simplify the notation using the periodic property of the extended images. Thus, $\mathbf{C}$ becomes

$$\mathbf{C} = \begin{bmatrix} C_0 & C_{N-1} & C_{N-2} & \cdots & C_1 \\ C_1 & C_0 & C_{N-1} & \cdots & C_2 \\ C_2 & C_1 & C_0 & \cdots & C_3 \\ \vdots & & & \ddots & \vdots \\ C_{N-1} & C_{N-2} & \cdots & C_1 & C_0 \end{bmatrix}, \tag{2.7}$$

where each block C_j is

$$C_j = \begin{bmatrix} h(0,j) & h(N-1,j) & h(N-2,j) & \cdots & h(1,j) \\ h(1,j) & h(0,j) & h(N-1,j) & \cdots & h(2,j) \\ h(2,j) & h(1,j) & h(0,j) & \cdots & h(3,j) \\ \vdots & & & \ddots & \vdots \\ h(N-1,j) & h(N-2,j) & \cdots & h(1,j) & h(0,j) \end{bmatrix}. \tag{2.8}$$

For example, we consider 2×2 images:

$$f_e = \begin{bmatrix} f(0,0) & f(0,1) & 0 & 0 \\ f(1,0) & f(1,1) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad g_e = \begin{bmatrix} g(0,0) & g(0,1) & 0 & 0 \\ g(1,0) & g(1,1) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Using vector notation, they can be written as follows:

$$f_e = \begin{bmatrix} f_{00} \\ f_{10} \\ 0 \\ 0 \\ f_{01} \\ f_{11} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad g_e = \begin{bmatrix} g_{00} \\ g_{10} \\ 0 \\ 0 \\ g_{01} \\ g_{11} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

and

$$\mathbf{C} = \begin{bmatrix} h_{00} & 0 & 0 & h_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{01} & 0 & 0 & h_{11} \\ h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 \\ 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 \\ 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} \\ h_{01} & 0 & 0 & h_{11} & h_{00} & 0 & 0 & h_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{01} & 0 & 0 & h_{11} & h_{00} & 0 & 0 & h_{10} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{01} & 0 & 0 & h_{11} & h_{00} & 0 & 0 & h_{10} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{11} & h_{01} & 0 & 0 & h_{10} & h_{00} \end{bmatrix}.$$

The sparsity of the matrix $\mathbf{C}$ is $\frac{n^2 \times N^2}{N^2 \times N^2} = \frac{1}{4}$ assuming that every element of $h(x, y)$ is nonzero, which is true in practice. The elements of each column of $\mathbf{C}_j$ are identical to those of the previous column, but are shifted one position to the right and wrapped around. Such a matrix is called *circulant* and denoted by $\mathbf{C}_j = \mathbf{circ}(\text{first column of } C_j) = \mathbf{circ}(h_e(0, j), h_e(1, j), h_e(2, j), \ldots, h_e(N-1, j))$. The matrix $\mathbf{C}$ is called a *block circulant with circulant blocks*, abbreviated **BCCB**, which means that it is circulant blockwise and each block is a circulant matrix. **BCCB** is not necessarily circulant. The $h_{0,0}$ element should be the center of the PSF as in the BTTB case. If the center is not located in the $(1, 1)$

position, we shift the extended matrix circularly so that the center of the PSF is in the $(1,1)$ position. This must be done in order to get the spectrum correct. Figure 2.2 shows the general nonzero structure of **BCCB** when the size of the image is 8×8. A further, more extensive discussion of BTTB and BCCB matrices may be found in Vogel [47].

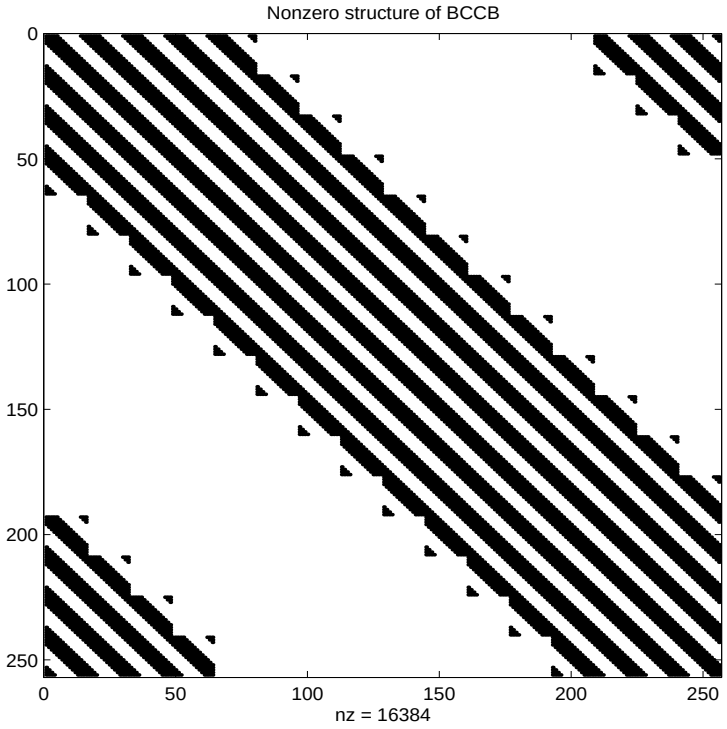

Figure 2.2: General nonzero structure of a **BCCB** matrix $\mathbf{C}$

2.3 Matrix-Vector Product

In the previous section, we modeled the degradation process as $g = \mathbf{H}f + \eta$. No matter what criteria we use to find the approximation of f from the observed g, the key computational operation is matrix-vector multiplication $\mathbf{H}x$ and $\mathbf{H}^T y$ for arbitrary vectors x and y. We provide an efficient way of computing the matrix-vector product using $\mathbf{C}x$ and $\mathbf{C}^T y$.

2.3.1 Kronecker Product and the vec Operator

Let A and B be $m \times n$ and $p \times q$ matrices, respectively. Then the *Kronecker* (or *tensor*) product of A and B is the $mp \times nq$ matrix defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix},$$

where a_{ij} is the (i,j) element of A. Some useful properties of the Kronecker product are in Appendix B. The **vec** operator maps $\mathbf{R}^{m\times n}$ to $\mathbf{R}^{mn}$, stacking the columns of a matrix in a vector:

$$\mathbf{vec}(A) = \begin{bmatrix} a_{11} & a_{21} & \ldots & a_{m1} & a_{12} & a_{22} & \ldots & a_{m2} & \ldots & a_{mn} \end{bmatrix}^T.$$

The **mat** operator defines the inverse of **vec**, and the **vec** (**mat**) operators reshape matrix (vector) into vector (matrix), respectively. From these definitions,

$$\begin{aligned} \mathbf{mat}(\mathbf{vec}(A)) &= A \quad \text{for matrix } A, \\ \mathbf{vec}(\mathbf{mat}(z)) &= z \quad \text{for vector } z, \\ (A \otimes B)\ \mathbf{vec}(X) &= \mathbf{vec}(BXA^T), \end{aligned}$$

for any matrix X of appropriate size.

2.3.2 Fourier Matrix

In this section we explain the Fourier matrix that represents the Fourier transform, and in the next section discuss the diagonalization of BCCB matrices through Fourier transform. Let w be the N-th root of 1:

$$w = \exp\left(-j\frac{2\pi}{N}\right) = \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right),$$

where $j = \sqrt{-1}$. We define the *Fourier* matrix of order N as

$$F_N(u,v) = \frac{e^{-j2\pi u\, v/N}}{\sqrt{N}}, \quad 0 \le u, v \le (N-1),$$

and using the definition of w, we can write F_N as

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & & & \ddots & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)(N-1)} \end{bmatrix}.$$

Since the sequence $w^k,\ k = 0, 1, \ldots,$ is periodic with period N, F_N reduces to

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{N-2} \\ \vdots & & & \ddots & \vdots \\ 1 & w^{N-1} & w^{N-2} & \cdots & w \end{bmatrix}. \tag{2.9}$$

A *discrete Fourier transform* (DFT), denoted by $\mathcal{F}$, of a vector $f \in \mathbf{R}^N$ is

$$[\mathcal{F}(f)]_i = \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} f_v \, e^{-j2\pi \, i \, v/N}, \quad i = 0, 1, 2, \ldots, (N-1).$$

This DFT can be written as a matrix-vector product using the Fourier matrix F_N:

$$\mathcal{F}(f) = F_N f.$$

We can then see that F_N and ${F_N}^*$ are symmetric:

$$F_N = {F_N}^T, \quad {F_N}^* = ({F_N}^*)^T = \overline{F_N}.$$

The important property is that F_N is unitary, *i.e.*

$${F_N}^* F_N = F_N {F_N}^* = I \quad \text{or} \quad {F_N}^{-1} = {F_N}^* = \overline{F}.$$

The superscript * denotes matrix conjugate transpose, ${F_N}^* = \overline{F_N}^T$. The DFT of a vector with N entries can be computed using a *fast Fourier transform* (FFT) algorithm with complexity $O(N \log_2 N)$, and similarly for the inverse FFT. This is a significant gain compared

with $O(N^2)$ for large N. The two-dimensional Fourier transform is represented by

$$[\mathcal{F}(f)]_{ik} = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} f_{uv}\, e^{-j2\pi\,(i\,u/N + kv/N)},\ i,k = 0,1,2,\ldots,(N-1).$$

The 2-D DFT is equivalent to applying the 1-D DFT to the columns and then the rows of the matrix. The Fourier transform of a 2-D image $f(x,y)$ can then be expressed using the Kronecker product of F_N:

$$(F_N \otimes F_N)\ \mathbf{vec}(f) = \mathbf{vec}(F_N\ f\ {F_N}^T) = \mathbf{vec}(F_N\ f\ F_N).$$

Since $F_N\ f\ F_N$ is the 2-D DFT of $f(x,y)$, it follows that $\mathbf{mat}((F_N \otimes F_N)\ \mathbf{vec}(f))$ is equivalent to the 2-D DFT of $f(x,y)$.

2.3.3 Diagonalization of Block Circulant Matrix

Any circulant matrix C, denoted by $\mathbf{circ}(c) = \mathbf{circ}(c_0, c_1, \ldots, c_{N-1})$, can be diagonalized by the Fourier matrix [10]:

$$C = {F_N}^* \,\mathrm{diag}(\lambda) F_N, \tag{2.10}$$

where F_N is the Fourier matrix of order N and

$$\lambda = \mathcal{F}\left(\sqrt{N}c\right) = \sqrt{N} F_N\ c.$$

The components of λ are the eigenvalues of C and the columns of ${F_N}^*$ are the corresponding eigenvectors.

From (2.10), each circulant block in (2.8) is decomposed using the Fourier matrix F_N:

$$C_k = {F_N}^*\, \Lambda_k\, F_N,$$

where Λ_k is a diagonal matrix composed of eigenvalues of each circulant block C_k:

$$\Lambda_k = \mathcal{F}\left(\sqrt{N} \cdot \text{first column of } C_k\right) = \mathcal{F}\left(\sqrt{N}\, C_k e_1\right).$$

Now we consider the eigendecomposition of BCCB $\mathbf{C}$. The matrix $\mathbf{C}$ is diagonalized as

follows [10]:

$$\mathbf{C} = (F_N \otimes F_N)^* \, \Lambda \, (F_N \otimes F_N), \tag{2.11}$$

where $\Lambda = \sum_{k=0}^{N-1} ({\Omega_N}^k \otimes \Lambda_k)$, $\Omega_N = \mathrm{diag}(1, w, w^2, \ldots, w^{N-1})$. The diagonal elements of diagonal matrix Λ can be expressed in terms of the elements of a 2-D Fourier transform of $h_e(x, y)$, $H(u, v)$ [17]:

$$\mathrm{diag}(\Lambda) = N\mathcal{F}(\mathbf{vec}(h_e)) = N(F_N \otimes F_N)\mathbf{vec}(h_e). \tag{2.12}$$

The first N elements are $H(0,0)$, $H(1,0)$, $\ldots$, $H(N-1,0)$; the next, $H(0,1)$, $H(1,1)$, $\ldots$,$H(N-1,1)$; and so on, with the last N diagonal elements being $H(0,N-1)$, $H(1,N-1)$, $\ldots$, $H(N-1,N-1)$ multiplied by N. The diagonal matrix containing the eigenvalues, Λ, is the 2-D DFT of $h_e(x, y)$ with a constant.

2.3.4 Cost of Matrix-Vector Computation

Now we show how we can compute $\mathbf{H}f$ efficiently by making use of

$$\mathbf{C}f_e = (F_N \otimes F_N)^* \Lambda (F_N \otimes F_N) f_e, \quad f_e = \mathbf{vec}(f_e(x, y))$$

in the following four steps.

(1) $\overline{f_e} = (F_N \otimes F_N) f_e$ is a vectorized version of the 2-D DFT of $f_e(x, y)$.

(2) Let $\overline{\lambda} = \Lambda \overline{f_e}$, The vector $\overline{\lambda}$ is computed as $\overline{\lambda} = \Lambda \overline{f_e} = \lambda \,.* \, \overline{f_e}$, where the operator $.*$ is the elementwise product of two vectors.

(3) $\hat{f}_e = (F_N \otimes F_N)^* \overline{\lambda}$ is the inverse 2-D DFT of $\overline{\lambda}$.

(4) Apply $\mathbf{mat}(\hat{f}_e)$ and extract the leading $n \times n$ sub-block from the extended output.

Similarly, we can compute

$$\mathbf{C}^T f = \mathbf{C}^* f = (F_N \otimes F_N)^* \Lambda^* (F_N \otimes F_N) f$$

efficiently as well. From the fact that $\mathbf{C}$ is a real matrix, $\mathbf{C}^T = \mathbf{C}^*$ holds.

To get the computational cost of direct computation $\mathbf{H}f$ for arbitrary f, we start with an example of 4×4 images. We denote the j-th column of f by f_j. Thus, $\mathbf{H}f$ becomes

$$\mathbf{H}f = \begin{bmatrix} H_0 & 0 & 0 & 0 \\ H_1 & H_0 & 0 & 0 \\ H_2 & H_1 & H_0 & 0 \\ H_3 & H_2 & H_1 & H_0 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} H_0 f_0 \\ H_1 f_0 + H_0 f_1 \\ H_2 f_0 + H_1 f_1 + H_0 f_2 \\ H_3 f_0 + H_2 f_1 + H_1 f_2 + H_0 f_3 \end{bmatrix}$$

and it requires $1+2+3+4 = (4 \cdot 5)/2 = 10$ block multiplications. Each block multiplication

$$H_j f_k = \begin{bmatrix} h_{0j} & 0 & 0 & 0 \\ h_{1j} & h_{0j} & 0 & 0 \\ h_{2j} & h_{1j} & h_{0j} & 0 \\ h_{3j} & h_{2j} & h_{1j} & h_{0j} \end{bmatrix} \begin{bmatrix} f_{0k} \\ f_{1k} \\ f_{2k} \\ f_{3k} \end{bmatrix} = \begin{bmatrix} h_{0j} f_{0k} \\ h_{1j} f_{0k} + h_{0k} f_{0k} \\ h_{2j} f_{0k} + h_{1k} f_{0k} + h_{0k} f_{0k} \\ h_{3j} f_{0k} + h_{2k} f_{0k} + h_{1k} f_{0k} + h_{0k} f_{0k} \end{bmatrix}$$

requires $1 + 2 + 3 + 4 = (4 \cdot 5)/2 = 10$ multiplications. In general, $\mathbf{H}f$ needs $1 + 2 + 3 + \cdots + n = n \cdot (n+1)/2$ block multiplications, and one block multiplication requires $1 + 2 + 3 + \cdots + n = n \cdot (n+1)/2$ multiplications. Therefore, the total work of the direct computation is $O(n^4/4)$ overall. On the other hand, the total amount of work using $\mathbf{C}f_e$ is $O(N^2 \log_2(N))$ because $O(N^2 \log_2(N))$ is needed for each 2-D DFT and we use one 2-D DFT and one inverse 2-D DFT. Computing the eigenvalue matrix Λ requires the 2-D DFT because eigenvalues are obtained from the 2-D DFT of h. However, since we compute this only once before starting the main algorithm, the computation is negligible.

2.3.5 Some Properties of BCCB

In addition to the block circulant property, $\mathbf{C}$ has some important properties.

(1) Each row (column) sum is $\sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} = 1$. Thus, $\mathbf{C}^T e = e$, $\mathbf{C}e = e$. Furthermore, each element is nonnegative; hence $\mathbf{C}$ is *doubly stochastic*.

(2) $\mathbf{C}^T = \mathbf{C}^*$ because $\mathbf{C}$ is a real matrix.

(3) $\mathbf{C}$ is a normal matrix, *i.e.*, $\mathbf{C}^T\mathbf{C} = \mathbf{C}\mathbf{C}^T$. Hence, $\mathbf{C}$ is unitarily diagonalizable: $\mathbf{C} = F^* \Lambda F, \quad F^* F = I$.

(4) $\mathbf{C}^T\mathbf{C} = \mathbf{C}\mathbf{C}^T = \mathbf{C}^*\mathbf{C} = (F^*\Lambda^* F)(F^*\Lambda F) = F^*(\Lambda^*\Lambda)F$. Hence, $\mathbf{C}^T\mathbf{C}$ is also a **BCCB** matrix.

(5) $\mathbf{C} = F^*\Lambda F$ is not symmetric. $\mathbf{C} = F^*\Lambda F \neq \mathbf{C}^T = \mathbf{C}^* = F^*\Lambda^* F$.

(6) $\mathbf{C}$ could be singular.

(7) $\mathbf{C}^{-1} = F^*\Lambda^{-1}F$ if $\mathbf{C}$ is nonsingular.

The first property derives from the fact that the PSF can be scaled such that the sum of all elements of the PSF is one. Furthermore, the sum of every row (column) in $\mathbf{C}^T\mathbf{C}$ is also one because $\mathbf{C}^T\mathbf{C}e = \mathbf{C}^T e = e$ and $\mathbf{C}\mathbf{C}^T e = \mathbf{C}e = e$. To prove that $\mathbf{C}$ is normal, we find

$$\begin{aligned} \mathbf{C}^T\,\mathbf{C} &= (F^*\Lambda^*F)(F^*\Lambda F) = F^*\Lambda\Lambda^*F, \\ \mathbf{C}\,\mathbf{C}^T &= (F^*\Lambda F)(F^*\Lambda^*F) = F^*\Lambda^*\Lambda F. \end{aligned}$$

It is obvious that $\Lambda^*\Lambda = \Lambda\Lambda^*$ holds because Λ and Λ^* are diagonal.

2.4 Test Problems

As test problems, we have two 256×256 color images provided by James Nagy. We refer to them as the star image and the satellite image. In addition, we have a smaller version of the satellite image of size 64×64. We use these three images throughout this dissertation to test and compare image restoration methods. The basic information about the test images is summarized in Table 2.1. In the next three sections, we present the true and degraded images. The blurred images are degraded by a spatially invariant point spread function, and also contain both Gaussian and Poisson noise.

	Star	Satellite	Small Satellite
# of nonzeros	470	6596	605
sparsity	0.0072	0.1006	0.1477

Table 2.1: The number of nonzeros and sparsity of images

2.4.1 Star Image

The star image is extremely sparse and each nonzero represents a star in our galaxy. The nonzeros are well separated and have no interdependency among them. Details are described in Nagy and website [32, 45]. The true and degraded images are shown in Figure 2.3.

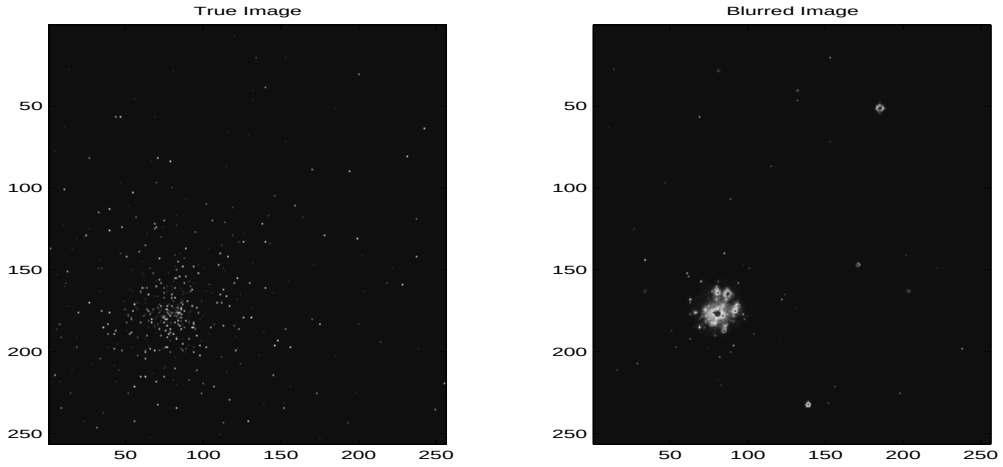

Figure 2.3: True and blurred image of star data

The distribution of pixels in the true and observed images is shown in Table 2.2.

	True Image	Observed Image
[1e+04 , 1e+05)	5	0
[1e+03 , 1e+04)	84	5
[100 , 1e+03)	381	695
[10 , 100)	0	6549
[1 , 10)	0	34080
[0.1 , 1)	0	8401
[0.01 , 0.1)	0	921
[0.001 , 0.01)	0	0
[0 , 1e−04)	65066	0

Table 2.2: Distribution of true and blurred image of star data

2.4.2 Satellite Image

Unlike the star image, the center of the satellite image is full of nonzeros representing the body of the satellite and axes. The pixels are clustered together. Details can be found in [33, 18]. The true and degraded images are shown in Figure 2.4. The distribution of pixel values for the true and observed images is shown in Table 2.3.

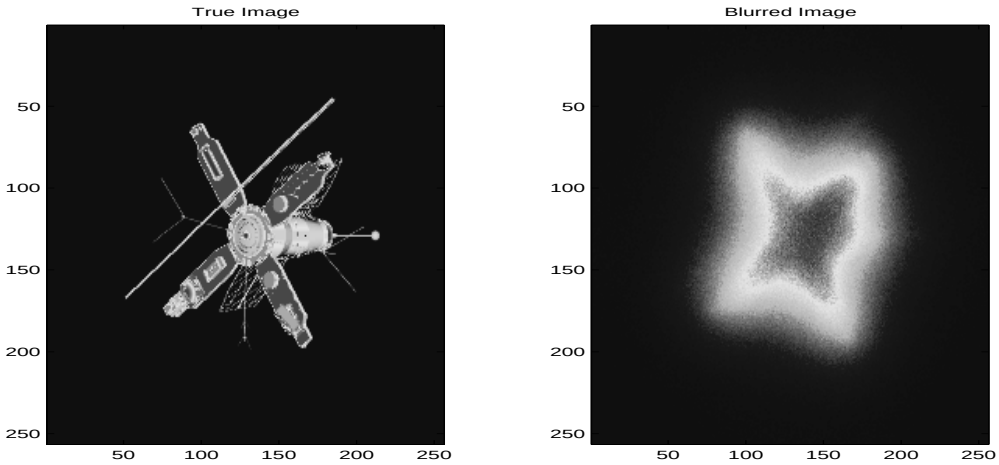

Figure 2.4: True and blurred image of satellite data

	True Image	Observed Image
[0.001 , 0.01)	0	0
[0.0001 , 0.001)	4881	254
[1e−05 , 0.0001)	1701	21018
[1e−06 , 1e−05)	67	37562
[1e−07 , 1e−06)	29	6700
[1e−08 , 1e−07)	0	2
[1e−09 , 1e−08)	0	0
[0 , 1e−09)	58858	0

Table 2.3: Distribution of true and blurred image of satellite data

2.4.3 Small Satellite Image

The smaller version of the satellite image is obtained by extracting a 64×64 subimage from the center of the satellite image in Section 2.4.2. This image is mainly used for testing and comparing restoration methods before we give extensive computational results in Chapter 9. The true and degraded images are shown in Figure 2.5. The distribution of pixels is shown in Table 2.4.

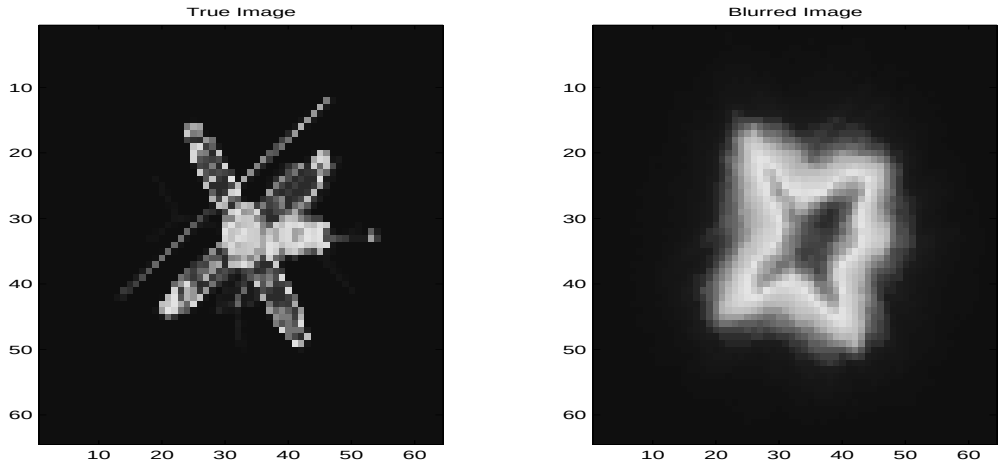

Figure 2.5: True and blurred image of small satellite data

			True Image	Observed Image
[	1e+04	1e+05)	264	0
[	1e+03	1e+04)	265	1162
[	100	1e+03)	69	1963
[	10	100)	5	971
[	1	10)	2	0
[	0.1	1)	0	0
[	0	1e−01)	0	0

Table 2.4: Distribution of true and blurred image of small satellite

Chapter 3

Mathematical Model

In this chapter, we examine the algebraic approach to image restoration using optimization techniques. The idea is to find an f that optimizes a criterion of performance with or without some constraints. A criterion of performance is hard to define because image quality is not only highly subjective, but also strongly dependent on the given application. We use the 2-norm of a residual vector and the distribution of the solution as the performance measure. To describe the algorithm, we interchangeably use A, x, and b for H, f, and g respectively since A, x, and b are more familiar notation.

3.1 Unconstrained Image Restoration

When we have no *a priori* information about noise, a meaningful criterion function is to seek an approximation f such that $\mathbf{H}f$ approximates g in a *least squares sense*. Thus, we want to find f such that

$$\|\eta\|^2 = \|g - \mathbf{H}f\|^2 \tag{3.1}$$

is minimized. This is an *unconstrained large-scale least squares problem* and the coefficient matrix $\mathbf{H}$ is BTTB as shown in (2.5).

3.1.1 Inverse Filter

Now we introduce the inverse filter from the algebraic viewpoint. If $\mathbf{H}$ has full column rank, the least-squares solution of (3.1) gives the estimate

$$\hat{f} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T g. \tag{3.2}$$

If $\mathbf{H}$ is square and nonsingular, $\hat{f} = \mathbf{H}^{-1}g$. If we assume that $f(x,y)$ and $g(x,y)$ are periodic with n in addition to space invariance, then BTTB in (2.5) becomes BCCB of size $n^2 \times n^2$. As noted in (2.11) of section 2.3.3, $\mathbf{H}$ is decomposed using a Fourier matrix:

$$\mathbf{H} = F^* \Lambda F,$$

where $F = (F_n \otimes F_n)$ and F_n is the $n \times n$ Fourier matrix. Hence, we have $\hat{f} = F^* \Lambda^{-1} F g$. Premultiplying both sides by F yields

$$F\hat{f} = \Lambda^{-1} F g. \tag{3.3}$$

Using the Fourier transform,

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)},$$

where Fg corresponds to $G(u,v)$ and Λ^{-1} corresponds to $H(u,v)$. This approach is called an *inverse filter* because $H(u,v)$ is considered a "filter" function that multiplies $F(u,v)$ to produce the transform of the degraded image $g(x,y)$. Computational difficulties arise if $H(u,v)$ vanishes or becomes very small in any region of interest in the u-v plane. The more serious difficulty is encountered in the presence of noise. From the fact that $G(u,v) = H(u,v)F(u,v) + N(u,v)$,

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)}.$$

This indicates that if $H(u,v)$ is small compared to $N(u,v)$, the term $N(u,v)/H(u,v)$ dominates the restoration result. In practice, $H(u,v)$ drops off rapidly as a function of distance from the origin of the u-v plane. However, the noise term usually becomes smaller at a much slower rate. The quality of the restored image is usually poor unless noise is negligible. Using either (3.2) and (3.3) requires an $\mathbf{H}$ that has full rank. If $\mathbf{H}$ is rank deficient, we have to resort to iterative methods where we do not assume full rank.

3.1.2 Conjugate Gradient Method

One of the main iterative methods for large-scale least squares problem is a form of the *conjugate gradient* (CG) method. This method was originally developed for solving symmetric positive definite linear systems (Hestenes and Stiefel [19]). Each iteration requires only one matrix-vector multiplication, plus a small number of inner products. The storage requirements are also very modest, since the vectors are reused. Hestenes and Stiefel gave a

reliable form of CG that is applicable to least squares problems (Freund, Golub, and Nachtigal [14]). We refer to this method as CGLS. It is mathematically equivalent to applying CG to the normal equations $\mathbf{H}^T\mathbf{H}f = \mathbf{H}^T g$, but without forming $\mathbf{H}^T\mathbf{H}$ explicitly and without forming products $\mathbf{H}^T(\mathbf{H}y)$. For a detailed formulation of CG for the least squares problem, we refer the reader to Björck [2]. For the least squares problem $\min \|Ax - b\|_2$, CGLS is as follows:

$r_0 = b - Ax_0$
$p_0 = s_0 = A^T r_0$
$\gamma_0 = \|s_0\|^2$
for $k = 0, 1, 2, \ldots$ while $\gamma_k > tol$ compute
    $q_k = Ap_k$
    $\alpha_k = \gamma_k / \|q_k\|^2$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = r_k - \alpha_k q_k$
    $s_{k+1} = A^T r_{k+1}$
    $\gamma_{k+1} = \|s_{k+1}\|^2$
    $\beta_k = \gamma_{k+1} / \gamma_k$
    $p_{k+1} = s_{k+1} + \beta_k p_k$
end

The convergence rate of CGLS depends on the distribution of singular values of A (eigenvalues of $A^T A$). When these values are in a small number of clusters, then CG converges quickly to a good approximate solution.

In the context of image restoration, the crucial drawback of the pure CG approach is that it does not always produce a nonnegative solution. Early iterations may result in solutions with large negative values and they remain as the iterations proceed [34]. In some applications, the quality of the image is severely damaged by negative elements.

3.1.3 Preconditioned Conjugate Gradient Method

To make CGLS more useful, some form of *preconditioning* may be used. A preconditioned conjugate gradient method (PCGLS) solves

$$\underset{y}{\text{minimize}} \ \|g - (\mathbf{H}C^{-1})y\|^2, \quad Cf = y, \tag{3.4}$$

where C should be chosen so that $\mathbf{H}C^{-1}$ has a more favorable spectrum than $\mathbf{H}$, and $Cf = y$ and $C^T g = z$ are easily solved for arbitrary y and z. The description of the PCGLS method for $\min \|Ax - b\|_2$ is as follows:

$r_0 = b - Ax_0$
$p_0 = s_0 = C^{-1}(A^T r_0)$
$\gamma_0 = \|s_0\|^2$
for $k = 0, 1, 2, \ldots$ while $\gamma_k > tol$ compute
    $t_k = C^{-1} p_k$
    $q_k = A t_k$
    $\alpha_k = \gamma_k / \|q_k\|^2$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = r_k - \alpha_k q_k$
    $s_{k+1} = C^{-T}(A^T r_{k+1})$
    $\gamma_{k+1} = \|s_{k+1}\|^2$
    $\beta_k = \gamma_{k+1} / \gamma_k$
    $p_{k+1} = s_{k+1} + \beta_k p_k$
end

The choice of an appropriate preconditioner depends on the trade-off between the gain in the convergence rate and the increased cost that results from applying the preconditioner. Many different choices have been proposed [2] for the general case, and some preconditioners based on the block Toeplitz structure of $\mathbf{H}$ have been developed. Chan [6] introduced the optimal circulant preconditioner, which is the closest circulant matrix in the Frobenius norm. Chan and Olkin [7] extend this to the block case, that is, they compute a BCCB matrix C minimizing $\|\mathbf{H} - C\|_F$. Systems involving C and C^T are easily solved using the block circulant property of C. The BCCB approximation works well when the unknown true solution is almost periodic. If this is not the case, the performance of BCCB can degrade.

The approximate inverse preconditioner using the properties of $\mathbf{H}$ has been proposed [33]. One simple preconditioner is the diagonal preconditioner. As we will see in Chapter 6, the diagonal elements of $A^T A$ are easy to compute. The optimal Kronecker product approximation was proposed by Kamm and Nagy [21]. The optimal Kronecker approximation finds $n \times n$ matrices A_k and B_k that minimize $\|\mathbf{H} - \sum A_k \otimes B_k\|_F$. The optimal A_k and B_k are banded Toeplitz matrices as well.

3.2 Constrained Image Restoration

As a natural extension from the unconstrained restoration, we consider some constrained least squares problems. First, if we have some knowledge about the noise, we can use this information to find a good estimate for f by solving

$$\boxed{\begin{array}{ll} \underset{f}{\text{minimize}} & \|Qf\|^2 \\ \text{subject to} & \|g - \mathbf{H}f\|^2 = \|\eta\|^2, \end{array}} \tag{3.5}$$

where Q is a linear operator on f defining a performance criterion. This formulation provides considerable flexibility in the restoration process because it yields different solutions for different choices of Q. The solution of (3.5) is given by

$$\hat{f} = (\mathbf{H}^T\mathbf{H} + \gamma Q^T Q)^{-1}\mathbf{H}^T g, \tag{3.6}$$

where γ is a Lagrange multiplier. The proof is given in [17].

Problem (3.5) is equivalent to the least squares problem

$$\min \left\| \begin{pmatrix} \mathbf{H} \\ \gamma^{\frac{1}{2}} Q \end{pmatrix} f - \begin{pmatrix} g \\ 0 \end{pmatrix} \right\|_2 . \tag{3.7}$$

The approach is known as Tikhonov regularization [47].

3.2.1 Wiener Filter

The Wiener filter is a solution to the dilemma of the inverse filter discussed in Section 3.1.1. It provides a least squares estimate of an object from its blurred image when the noise is additive and independent of the image. We define

$$Q^T Q = {R_f}^{-1} R_\eta, \tag{3.8}$$

where R_f and R_η are the correlation matrices of f and η defined by

$$R_f = \mathrm{E}(ff^T), \quad R_\eta = \mathrm{E}(\eta\eta^T).$$

Since the correlation is defined by expectation, it follows that R_f and R_η are real symmetric matrices. The Fourier transform of these correlations is the power spectrum (or spectral density) of $f(x, y)$ and $\eta(x, y)$, denoted by $S_f(u, v)$ and $S_\eta(u, v)$. We can choose R_f and R_η

to be approximate block-circulant matrices, and then R_f and R_η can be diagonalized by $F = F_n \otimes F_n$ as we do with $\mathbf{H}$. From (3.6) and (3.8),

$$\hat{f} = (\mathbf{H}^T\mathbf{H} + \gamma {R_f}^{-1} R_\eta)^{-1} \mathbf{H}^T g.$$

If we assume that $f(x,y)$ and $g(x,y)$ are periodic with n with $R_f = F^*\Lambda_f F$ and $R_\eta = F^*\Lambda_n F$,

$$\hat{f} = (F^*\Lambda^*\Lambda F + \gamma F^* {\Lambda_f}^{-1}\Lambda_\eta F)^{-1} F^*\Lambda^* F g.$$

Multiplying both sides by F and performing some manipulations gives

$$F\hat{f} = (\Lambda^*\Lambda + \gamma {\Lambda_f}^{-1}\Lambda_\eta)^{-1}\Lambda^* F g.$$

When we rewrite this in terms of 2-D Fourier transforms,

$$\begin{aligned} \hat{F}(u,v) &= \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma[S_\eta(u,v)/S_f(u,v)]}\right] G(u,v) \\ &= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + \gamma[S_\eta(u,v)/S_f(u,v)]}\right] G(u,v), \end{aligned}$$

where $|H(u,v)|^2 = H^*(u,v)H(u,v)$. When $\gamma = 1$, it is called the *Wiener filter*. If γ is variable, this expression is called a *parametric Wiener filter*. In the absence of noise, $S_\eta(u,v) = 0$, this reduces to the ideal inverse filter. Thus, the Wiener filter is expressed as

$$W(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + \gamma[S_\eta(u,v)/S_f(u,v)]}\right]. \tag{3.9}$$

From (3.9), it follows that

(1) $W(u,v) \approx \dfrac{1}{H(u,v)}$ if $H(u,v) \neq 0$ and $\dfrac{S_f}{S_\eta} \gg |H(u,v)|^{-2}$; that is, the Wiener filter becomes an inverse filter when the signal to noise ratio is much greater than $|H(u,v)|^{-2}$.

(2) $W(u,v) = 0$ if $|H(u,v)| = 0$ or $\dfrac{S_f}{S_\eta} \ll |H(u,v)|^{-2}$.

Clearly, the Wiener filter will give a better approximation than the inverse filter because we use the noise information. However this does not enforce nonnegativity. When noise information is not available, we consider the nonnegatively constrained problem.

3.2.2 Nonnegatively Constrained Image Restoration

In the unconstrained formulation, we can obtain negative values for some elements of the solution f, which diminishes the quality of an image. It is reasonable to formulate the degradation model as a nonnegatively constrained image restoration because the image is nonnegative from the fact that it is one form of energy. We establish an attractive model, a nonnegatively constrained least squares (NNLS) problem, by adding nonnegativity constraints to the least squares formulation:

$$\begin{aligned} &\underset{f}{\text{minimize}} \quad \|g - \mathbf{H}f\|^2 \\ &\text{subject to} \quad f \geq 0. \end{aligned}$$

We discuss NNLS and other formulations in detail in Chapter 4.

Chapter 4

Nonnegatively Constrained Image Restoration

Although the inverse filter is a meaningful criterion of performance when we have no information about the noise, the quality of the output is susceptible to the noise and to negative elements. As we have seen, the Wiener filter provides the solution to the problems posed by the inverse filter, but it requires some information about noise (such as correlation) and it does not produce nonnegative solutions. As a remedy, we formulate the image restoration problem as a nonnegatively constrained least squares problem. In this chapter, we examine the possibilities of problem formulation and the numerical methods for each problem.

4.1 Choice of Problem

4.1.1 NNLS

The unconstrained least squares estimate has a sound statistical justification behind our intuition. In fact, the least squares estimate is optimal without any assumptions that the random noise follows a particular distribution [2]. We only assume that the random variables η_i are uncorrelated and all have zero means and the same variance, that is, $\mathrm{E}(\eta) = 0, \Sigma = \sigma^2 I$, where Σ is the covariance matrix. If the random noises η_i follow a normal distribution, we have the following property.

Theorem 4.1.1 *The least squares estimate is the maximum likelihood estimate when the random noise variables η_i follow a normal distribution.*

Proof. Under the assumption that the noise variables η_i follow a normal distribution, the likelihood function is

$$\begin{aligned} f(\eta) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\eta-0)\Sigma^{-1}(\eta-0)} \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(b-Ax)\Sigma^{-1}(b-Ax)} \end{aligned}$$

where $|\Sigma|$ is the determinant of Σ. We use $\mathrm{E}(\eta) = 0$ in the first equation. Since $\Sigma = \sigma^2 I$, maximizing the likelihood function $f(\eta)$ is equivalent to minimizing the 2-norm of the residual vector, $\|b - Ax\|^2$. Therefore, computing a least squares estimate is identical to obtaining a maximum likelihood estimate when the noise variables η_i are independent and identically distributed random variables having normal distribution. ■

However, the least squares estimate must be constrained because images are nonnegative. Therefore, we use the NNLS problem:

$$\boxed{\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) = \frac{1}{2}\|b - Ax\|^2 \\ \text{subject to} & x \geq 0. \end{array}} \tag{4.1}$$

The gradient of $f(x)$ is $\nabla f(x) = A^T(Ax - b)$ and the KKT optimality conditions for (4.1) are

$$\begin{aligned} x &\geq 0 \\ \nabla f(x) &\geq 0 \\ \nabla f(x)^T x &= 0. \end{aligned} \tag{4.2}$$

The problem reduces to finding a *nonnegative* x that satisfies $(Ax - b)^T Ax = 0$. Handling nonnegativity constraints is computationally nontrivial because we are dealing with expensive nonlinear equations. An equivalent, but more tractable, formulation of NNLS using the residual variable $p = b - Ax$ is as follows:

$$\boxed{\begin{array}{ll} \underset{x,\,p}{\text{minimize}} & \frac{1}{2}p^T p \\ \text{subject to} & Ax + p = b, \quad x \geq 0. \end{array}} \tag{4.3}$$

The advantage of this formulation is that we have a simple and separable objective function with linear and nonnegativity constraints. We delve into this formulation in Chapters 5–7.

4.1.2 ML

The maximum likelihood (ML) model assumes the noise has a Poisson distribution. It has been widely used in image reconstruction arising from medical imaging such as positron emission tomography [34]. The problem formulation using the *maximum likelihood function* $\mathcal{J}(x)$ is as follows:

$$\boxed{\begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{J}(x) = \sum (Ax)_i - \sum b_i \log (Ax)_i \\ \text{subject to} & x \geq 0. \end{array}} \tag{4.4}$$

Again we can introduce variables $s = Ax$ to obtain a linearly constrained formulation:

$$\boxed{\begin{array}{ll} \underset{x,\, s}{\text{minimize}} & e^T s - \sum b_i \log s_i \\ \text{subject to} & Ax - s = 0, \quad x,\, s \geq 0 \end{array}} \tag{4.5}$$

The justification for using a maximum likelihood function comes from the statistical properties of the random noise, which are related to the Poisson noise. If we assume that $\mathrm{E}(b) = Ax$ and b_i are independently Poisson distributed, then the likelihood is written as

$$\mathrm{Prob}(b|Ax) = \prod_{i=1}^{n} \frac{(Ax)_i{}^{b_i} e^{-(Ax)_i}}{b_i!}.$$

The ML criterion is formed by taking the log of the likelihood:

$$\log \mathrm{Prob}(b|Ax) = \sum_{i=1}^{n} \Big(b_i \log (Ax)_i - (Ax)_i - \log(b_i!) \Big).$$

Ignoring the constant term and putting a minus sign to convert to minimization, we obtain $\mathcal{J}(x) = \sum (Ax)_i - \sum b_i \log (Ax)_i$ as an objective function. Another interpretation of the Poisson model for the ML objective function is found in [20]. Chapter 8 covers the ML formulation in detail.

4.1.3 Chebyshev and ℓ_1 Approximation

In solving a least squares problem, we minimize the 2-norm of a residual vector as a performance measure. As alternative measures, we can use the ℓ_1 and $\ell_i nfty$ norm. The ℓ_1

approximation formulation is

$$\begin{array}{l} \underset{x}{\text{minimize}} \quad \|b - Ax\|_1 \\ \text{subject to } \; x \geq 0, \end{array}$$

and the ℓ_∞ formulation is

$$\begin{array}{l} \underset{x}{\text{minimize}} \quad \|b - Ax\|_\infty \\ \text{subject to } \; x \geq 0. \end{array}$$

Both problems can be transformed into a linear program (LP) by introducing additional variables. Since A is an operator, we must apply iterative methods that do not require an explicit A. For both the NNLS and LP formulations, we make use of an interior-point method, in which each search direction is computed by an iterative least squares solver similar to CGLS. The drawback of these ℓ_1 and Chebyshev formulations is that they do not have the statistical justification that the least squares formulation does. We discuss both formulations in Chapter 8.

4.2 Choice of Method

4.2.1 SSD (Scaled Steepest Descent) Method

One popular algorithm for handling the ML criterion is the expectation maximization (EM) method, which solves the *maximum likelihood function* formulation (4.4). The EM method was originally developed by Dempster *et al.* [12] and proposed for PET (Positron Emission Tomography) by Shepp and Vardi [44] and Lange and Carson [5]. Kaufman [22, 23] applied EM algorithms to least squares, and the penalized least squares problem. To derive a numerical algorithm for (4.4), Hanke, Nagy, and Vogel [18] use an approach that transforms the constrained minimization problem into an unconstrained problem using the parameterization $x = e^z$ elementwise [18]. Using the chain rule, the gradient of $\mathcal{J}(x)$ with respect to z:

$$\nabla_z \mathcal{J}(x) = X \nabla_x \mathcal{J}(x) = X A^T Y^{-1}(Ax - b) = X A^T (e - Y^{-1} b),$$

where $y = Ax$, $Y = \text{diag}(y)$, and $X = \text{diag}(x)$. Using the energy conservation relation, $A^T e = e$, we obtain

$$\nabla_z \mathcal{J}(x) = x - X A^T Y^{-1} b.$$

By setting $\nabla_z \mathcal{J}(x) = 0$, we obtain the fixed point iteration

$$x_{k+1} = X_k A^T {Y_k}^{-1} b, \quad X_k = \mathrm{diag}(x_k),\ Y_k = \mathrm{diag}(Ax_k).$$

The EM algorithm is a scaled steepest descent (SSD) approach that elegantly handles the nonnegativity constraints of the problem. Indeed, we can think of this method as the steepest descent method with the distance of each element to the nonnegativity constraint used as the scale factor. Many variants of the EM algorithm are called EM-like or EM-type algorithms [23]. One advantage of the EM method is that it is very easy to implement from the fixed point relationship, but it converges rather slowly. One critical disadvantage is that some elements of $y = Ax$ could be very small, resulting in the failure of the EM in some instances. Another weakness comes from the assumption $A^T e = e$. To be precise, the PSF is scaled so that A has, approximately, a conservation of energy property, but $A^T e = e$ holds only for pixels that are away from the boundary (that is, close to the center of the image). If some important pixels are located around the boundary, this method would not generate a high quality image.

We can also apply this technique ($x = e^z$) to the least squares problem

$$f(x) = \tfrac{1}{2}\|b - Ax\|^2$$

to enforce the nonnegativity of x. Then we have

$$\nabla_z f(x) = X \nabla_x f(x) = X A^T (Ax - b), \quad X = \mathrm{diag}(x).$$

Setting $\nabla_z f(x) = 0$ gives us the KKT conditions (4.2). This is discussed further in Chapter 5.

4.2.2 Methods for Quadratic Programs

By definition of the 2-norm, we may rewrite NNLS (4.1) as the following quadratic program (QP):

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \tfrac{1}{2} x^T A^T A x - (A^T b)^T x \\ \text{subject to} & x \geq 0. \end{array} \tag{4.6}$$

In other words, a nonnegatively constrained least squares problem is a QP whose Hessian is a positive semidefinite matrix $A^T A$. The dual active set method for QP with a positive definite

matrix was proposed by Goldfarb and Idnani [15]. This method uses the QR factorization of A or the Cholesky factorization of A^TA, which requires an explicit A. Boland proposed an extension for the positive semidefinite case, which also needs A explicitly [3]. In our study, A is an operator and is not given as an explicit form. These methods are therefore not applicable.

4.2.3 Methods for LCP

Since A^TA is positive semidefinite, the program (4.6) is equivalent to the linear complementarity problem (LCP):

$$\begin{aligned} x &\geq 0 \\ q + Mx &\geq 0 \\ x^T(q + Mx) &= 0, \end{aligned} \tag{4.7}$$

where $M = A^TA$ and $q = -A^Tb$. Note that the KKT conditions (4.2) are the same as (4.7). The NNLS problem can therefore be solved by any one of the large number of algorithms designed for LCP. In particular, active-set methods [9, 31] can be used for this purpose. Portugal, Júdice and Vicente [38] have proposed block principal pivoting methods for the solution of large-scale strictly convex quadratic programs with nonnegativity constraints. An active-set method can be regarded as a single principal pivoting algorithm.

Separately, various interior methods have been proposed in [37] and elsewhere. However, most implementations of active-set methods and interior methods assume A or A^TA is available explicitly, whereas in our case it is just an operator. Thus, many algorithms for LCP are not applicable.

4.2.4 Primal-Dual Interior Approach

Primal-dual interior point methods have been widely used to solve nonlinear problems [50]. We develop a primal-dual interior point method to tackle NNLS problem (4.3), where the objective function is separable and the constraints are linear with nonnegative bounds on x. We call this method PDSCO (Primal-Dual Barrier Method for Separable Convex Objectives). It is especially suitable when the output is a gray-scale image, where a particular value being exactly "zero" or close to zero is immaterial. The approach is also suitable because slight inaccuracies below the gray-scale threshold are inconsequential, but obtaining an image rapidly is a concern. For color images, the quality of the output is not as good as for gray-scale images, but images are reconstructed in an excellent way within a moderate

number of iterations. PDSCO [41] is implemented in MATLAB and solves more general problems of the form

$$\begin{array}{ll} \underset{x,p}{\text{minimize}} & \varphi(x) + \frac{1}{2}\|\gamma x\|^2 + \frac{1}{2}\|p\|^2 \\ \text{subject to} & Ax + \delta p = b, \quad x \geq 0, \end{array} \tag{4.8}$$

where $\varphi(x)$ is a smooth separable function. From the separability of the objective function, the Hessian is a diagonal matrix that is easy to handle and store. In addition to this advantage, there is no computation of $A^T A$. PDSCO can also be applied to the ML formulation (4.5) as described in Chapter 8.

4.2.5 Reduced-Gradient Method

The reduced-gradient method is designed for nonlinear programs with linear constraints and bounds:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & F(x) \\ \text{subject to} & Gx = h \\ & l \leq x \leq u, \end{array}$$

where G is $m \times n$ matrix, $m \leq n$. The implementation in MINOS [29, 30] is designed for problems in which G is a general sparse matrix. A sparse basis package is used to generate search directions in the null space of the active-constraint matrix, and the associated reduced Hessian is approximated by a dense quasi-Newton method.

MINOS would be applicable to formulation (4.3) if A were a sparse matrix, but instead we apply it to the bound-constrained formulation (4.6), for which objective and its gradient can be evaluated efficiently even if A is an operator.

The main motivation for using MINOS comes from the fact that for certain applications such as astronomy, the true images are very sparse (only a few hundred pixels have nonzero values). Since the performance of this algorithm is based on the number of nonzeros in the solution, the sparser the image the better. Also, if the image size is moderate (say 64×64 or less), the reduced gradient method should be efficient regardless of how sparse the image is.

MINOS can also be applied to the ML formulation (4.4). We investigate this further in Chapter 8.

Chapter 5

NNLS using SSD

The scaled steepest descent (SSD) method is one of methods that can enforce nonnegativity. In this chapter, we apply SSD method to problem NNLS. Kaufman [23] introduced the same algorithm for NNLS and called it EM-LS, which has a bounded line search. Saad [40] and Nagy [34] called the same algorithm the modified residual norm steepest descent method (MRNSD). The SSD method for the maximum likelihood function is discussed in Chapter 8.

5.1 SSD Method

As in Nagy [34], we can enforce the nonnegativity of x by applying the transformation $x = e^z (x_j = e^{z_j} \text{ for all } j)$. Thus we have the constrained problem

$$\boxed{\begin{array}{ll} \underset{x}{\text{minimize}} & \|Ax - b\|_2{}^2 \\ \text{subject to} & x = e^z, \end{array}} \tag{5.1}$$

which becomes the unconstrained problem

$$\min_z \ \overline{f}(z) = \|Ae^z - b\|_2{}^2.$$

We can express the gradient of $\overline{f}(z)$ in terms of $x = e^z$:

$$g(z) = \nabla_z \overline{f}(z) = X \nabla_x f(x) = X A^T (Ax - b),$$

where $X = \text{diag}(x)$.

The steepest descent method is

$$\begin{array}{rrcccl} & z_{k+1} & = & z_k - \alpha g(z_k) & & \\ \Rightarrow & x_{k+1} & = & e^{z_k - \alpha g(z_k)} & = & x_k \cdot e^{-\alpha g(x_k)} \\ \Rightarrow & x_{k+1} & \approx & x_k(1 - \alpha g(x_k)) & = & x_k - \alpha X_k g(x_k), \end{array} \tag{5.2}$$

where α is a positive steplength chosen by a line search to give a reduction in the objective function. Now we develop an iterative method of the form

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)},$$

where $p^{(k)} = -X_k A^T(Ax^{(k)} - b)$ and the steplength α_k is chosen to maintain nonnegativity as well as reduce the objective. Basically, this is the same as the steepest descent approach applied to the unconstrained least squares problem, $\min \|Ax - b\|^2$, but we scale the steepest descent direction by X_k and have an upper bound on α_k in the line search. We describe the SSD method as follows:

Scale b such that $\|b\|_\infty = 1$
$x = x^{(0)} \geq 0$
$g = A^T(Ax - b)$
for $k = 0, 1, 2, \ldots$
    $X = \text{diag}(x)$
    $p = -Xg$
    $\gamma = -g^T p$
    $u = Ap$
    $\theta = \gamma / u^T u$
    $\alpha = \min(\theta, \texttt{StepTol} * \min_{p_j < 0}(-x_j/p_j))$, $\texttt{Steptol} = 0.99$
    $x = x + \alpha p$
    $z = A^T u$
    $g = g + \alpha z$
end

where θ is the minimum of f along the direction p (from the fact that $\partial f(x + \alpha p)/\partial \alpha = p^T \nabla f(x + \alpha p) = 0$), and α maintains the nonnegativity of x.

The convergence rate of the steepest descent method for minimizing a function $f(x)$ depends on the condition of the Hessian of f. Luenberger [26] gives an analysis of steepest descent for a quadratic function without constraints and this illustrates all the important

convergence characteristics of steepest descent. Let

$$f(x) = \frac{1}{2}x^T Q x - x^T b,$$

where Q is a symmetric positive definite matrix. The unique minimizer x^* satisfies $Qx^* = b$. Introducing the function

$$E(x) = \frac{1}{2}(x - x^*)^T Q(x - x^*)$$

for computational convenience, we have $E(x) = f(x) + \frac{1}{2}x^{*T} Q x^*$, which differs from $f(x)$ by a constant. We wish to analyze the *diagonally scaled* steepest descent method,

$$x_{k+1} = x_k - \alpha_k D g_k,$$

where $g_k = Qx_k - b$ and D is a diagonal positive definite scaling matrix. The optimal steplength α_k can be found by differentiating $f(x_k - \alpha D^{-1} g_k)$ with respect to α, leading to

$$\alpha_k = \frac{g_k^T D g_k}{g_k^T DQD g_k}.$$

We obtain the relationship between $E(x_{k+1})$ and $E(x_k)$ from the following theorem.

Theorem 5.1.1

$$E(x_{k+1}) = \left\{1 - \frac{g_k^T D g_k}{(g_k^T DQD g_k)(g_k^T Q^{-1} g_k)}\right\} E(x_k). \tag{5.3}$$

Proof. The proof is by direct computation and similar to the proof without diagonal scaling [26]. By setting $y_k = x_k - x^*$, we have

$$\frac{E(x_k) - E(x_{k+1})}{E(x_k)} = \frac{2\alpha_k {g_k}^T DQ y_k - \alpha_k^2 g_k^T DQD g_k}{y_k^T Q y_k}.$$

Using $g_k = Qy_k$ we have

$$\frac{E(x_{k+1}) - E(x_k)}{E(x_k)} = \frac{g_k^T D g_k}{(g_k^T DQD g_k)(g_k^T Q^{-1} g_k)}$$

and the result follows. ■

Theorem 5.1.2 *The convergence rate of the diagonally scaled steepest descent method for a quadratic function is linear.*

Proof. In order to obtain a bound on the rate of convergence, we let $z_k = D^{1/2} g_k$ in (5.3). Now we have

$$\frac{g_k^T D g_k}{(g_k^T DQDg_k)(g_k^T Q^{-1} g_k)} = \frac{z_k^T z_k}{(z_k^T R z_k)(z_k^T R^{-1} z_k)},$$

where $R = D^{1/2} Q D^{1/2}$ is symmetric positive definite and the diagonals of D are positive. From the Kantorovich inequality [26],

$$\frac{g_k^T D g_k}{(g_k^T DQDg_k)(g_k^T Q^{-1} g_k)} = \frac{z_k^T z_k}{(z_k^T R z_k)(z_k^T R^{-1} z_k)} \geq \frac{4Bb}{(B+b)^2}$$

where b and B are the smallest and the largest eigenvalues of R. Then, we have

$$E(x_{k+1}) \leq \left\{1 - \frac{4Bb}{(B+b)^2}\right\} E(x_k) = \left(\frac{B-b}{B+b}\right)^2 E(x_k).$$

We denote the smallest and largest eigenvalues of Q by a and A, respectively. Then we have the following relationship between b, B and a, A:

$$\begin{array}{rcccccccl} B & = & \|R\|_2 & = & \|DQ\|_2 & \leq & \|D\|_2 \|Q\|_2 & = & q_1 A \\ b & = & \frac{1}{\|R^{-1}\|_2} & = & \frac{1}{\|D^{-1}Q^{-1}\|} & \geq & \frac{1}{\|D^{-1}\|\|Q^{-1}\|} & = & q_n\, a, \end{array} \tag{5.4}$$

where q_1 and q_n are the largest and the smallest eigenvalues of D. From this, we have the following relationship:

$$\frac{B-b}{B+b} = \frac{\kappa_B - 1}{\kappa_B + 1} \leq \frac{\kappa_A/\kappa_D - 1}{\kappa_A/\kappa_D + 1} \equiv K,$$

where $\kappa_A = A/a$, $\kappa_B = B/b$, $\kappa_D = q_1/q_n > 1$ are condition numbers of A, B, and D, respectively. Thus, $E(x_{k+1}) \leq K^2 E(x_k)$ as required. ■

We can generalize the above analysis of quadratic problems to the nonquadratic case. For details, see [26]. In practice the method of steepest descent usually progresses quite well during the early iterations, but becomes very slow as the solution is approached. The SSD method also suffers in this way, as shown in Table 5.1, Table 5.2, and Figure 5.1.

Error	x^*	x_{20}	x_{100}	x_{500}	x_{1000}	x_{2000}
$\|b - Ax\|_2$	18283	8062.64	3164.17	1872.26	1751.19	1700.38
$\frac{\|x - x^*\|_2}{\|x^*\|_2}$	0	0.43651	0.436345	0.479107	0.492015	0.505763

Table 5.1: Absolute and relative error of SSD restored images

	x^*	Observed	x_{20}	x_{100}	x_{500}	x_{1000}	x_{2000}
[1e+04 , 1e+05)	264	0	214	245	252	248	247
[1e+03 , 1e+04)	265	1162	512	339	282	274	274
[100 , 1e+03)	69	1963	1286	634	375	281	237
[10 , 100)	5	971	2084	2877	2015	1659	1314
[1 , 10)	2	0	0	1	1167	1344	1067
[0.1 , 1)	0	0	0	0	5	284	649
[0.01 , 0.1)	0	0	0	0	0	6	257
[0.001 , 0.01)	0	0	0	0	0	0	42
[0.0001 , 0.001)	0	0	0	0	0	0	7
[1e−05 , 0.0001)	0	0	0	0	0	0	2
[0 , 1e−05)	3491	0	0	0	0	0	0

Table 5.2: Distribution of true image and SSD restored images

The slow convergence is verified by the decrease in the 2-norm of the residual vector in Table 5.1. One unexpected fact is that the relative error worsens as iterations proceed. This is perhaps because x^* is not the true least squares solution and the more iterations we go, the more pixels having small values we have. The slow convergence is also verified by the distribution of solutions in Table 5.2, where x_j is the solution after j iterations. In these solutions, the number of pixels in the first two layers determine the quality of the image as we see in the true case. Comparing the distributions of x_{100}, x_{500}, x_{1000} and x_{2000}, we observe that there is no noticeable difference among them and that the pixels having small values are growing as iterations proceed.

The SSD restored images are shown in Figure 5.1. The image resolution does not improve significantly after 100 iterations. In x_{2000}, we can see the axes around the satellite relatively clearly, but the satellite in the center does not look as good as the satellite in x_{100}. More iterations do not necessarily improve the quality of the solution.

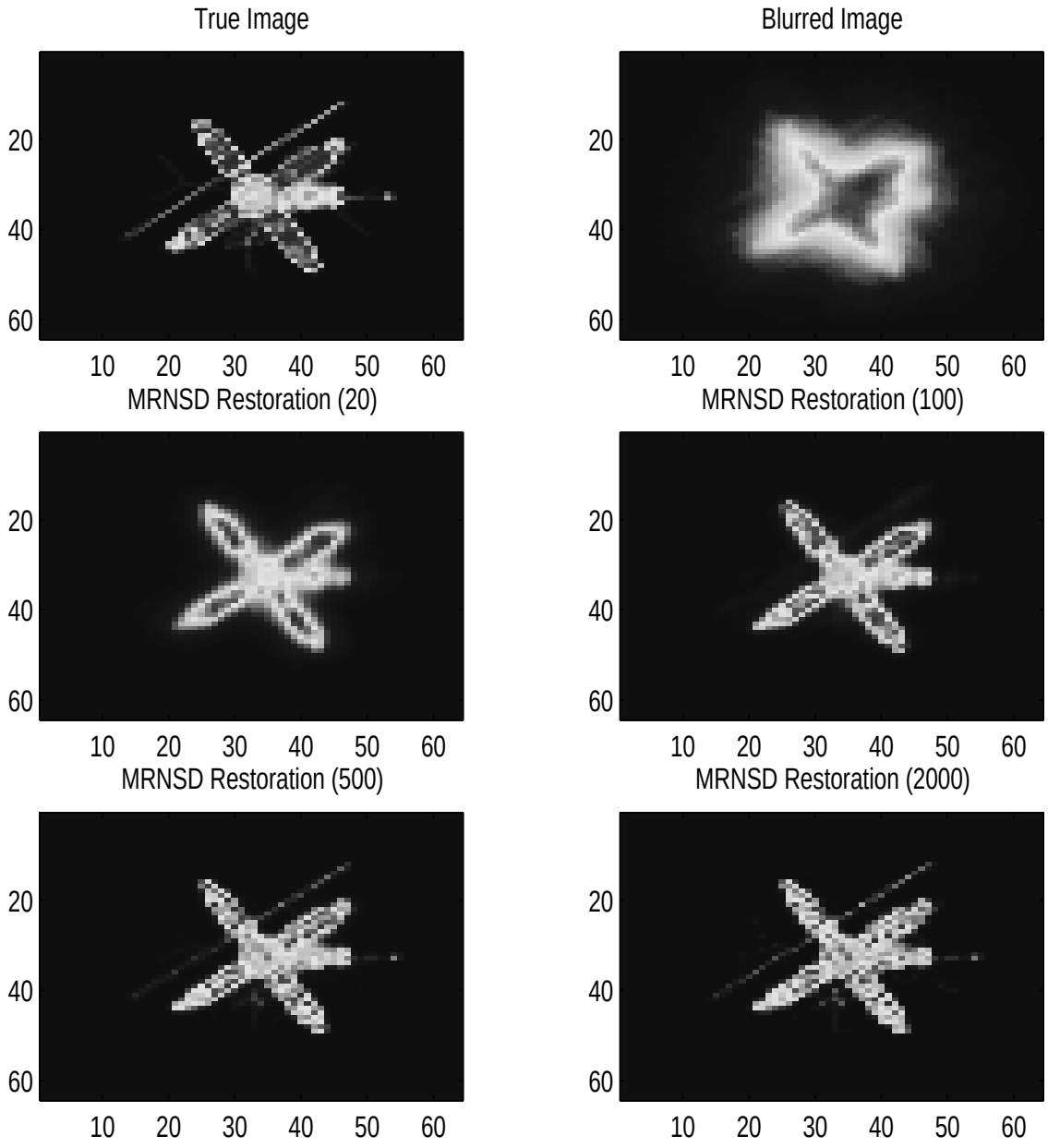

Figure 5.1: Solutions from SSD method after various numbers of iterations

Chapter 6

NNLS using PDSCO

In this chapter, we discuss a numerical method for solving the nonnegative least squares problem NNLS. We state the problem in terms of the residual vector p:

$$\begin{array}{l} \underset{x,\,p}{\text{minimize}} \quad \frac{1}{2}p^T p \\ \text{subject to} \quad Ax + p = b, \quad x \geq 0. \end{array} \tag{6.1}$$

Here we have a *separable convex* objective function and linear constraints. We now explain the PDSCO algorithm (Primal-Dual barrier method, Separable Convex Objective) to solve a perturbed form of (6.1).

6.1 PDSCO

Most of our discussion applies to regularized problems of the form

$$\begin{array}{l} \underset{x,\,p}{\text{minimize}} \quad \frac{1}{2}\|\gamma x\|^2 + \frac{1}{2}\|p\|^2 \\ \text{subject to} \quad Ax + \delta p = b, \quad x \geq 0, \end{array} \tag{6.2}$$

where $\delta = 1$ for the NNLS problem and the specified scalar γ is typically small. (For regularized linear programs, δ would also be small. Section 6.6 provides a further discussion of regularization.) This problem is a convex quadratic program. The dual problem is

$$\begin{array}{l} \underset{x,\,y,\,z}{\text{maximize}} \quad b^T y - \frac{1}{2}\|\delta y\|^2 - \frac{1}{2}\|\gamma x\|^2 \\ \text{subject to} \quad A^T y + z = \gamma^2 x, \quad z \geq 0. \end{array} \tag{6.3}$$

PDSCO uses many of the techniques developed earlier for linear programs. In particular, it uses a primal-dual interior method (Megiddo [28]). The distinguishing feature is that the search directions are converted to linear least-squares problems and solved by a conjugate gradient (CG) type iterative method (LSQR), with early termination at all stages. Introducing the barrier function $-\mu \sum_j \log x_j$ into the objective of (6.2) leads to the nonlinear equations defining the central trajectory:

$$\begin{aligned} Ax + \delta^2 y &= b \\ A^T y + z &= \gamma^2 x \\ XZe &= \mu e, \end{aligned} \tag{6.4}$$

where μ is a barrier parameter ($\mu > 0$), $X = \text{diag}(x)$, $Z = \text{diag}(z)$, $x, z > 0$, and e is a vector of ones. If $x^*(\mu)$, $y^*(\mu)$, and $z^*(\mu)$ are the solution of (6.4), then $x^*(\mu) \to x^*$, $y^*(\mu) \to y^*, z^*(\mu) \to z^*$ as $\mu \to 0$ [13]. In practice, μ is reduced to a certain level (for example, 10^{-8}), below which it does not decrease. The primal-dual algorithm uses Newton's method to obtain search directions for (6.4) from equations of the form

$$\begin{aligned} A\Delta x + \delta^2 \Delta y &= r \equiv b - Ax - \delta^2 y \\ -\gamma^2 \Delta x + A^T \Delta y + \Delta z &= t \equiv \gamma^2 x - A^T y - z \\ Z\Delta x + X\Delta z &= v \equiv \mu e - XZe. \end{aligned} \tag{6.5}$$

In matrix notation,

$$\begin{bmatrix} A & \delta^2 I & 0 \\ -\gamma^2 I & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r \\ t \\ v \end{bmatrix}. \tag{6.6}$$

Eliminating Δz gives us

$$\begin{bmatrix} -H & A^T \\ A & \delta^2 I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} w \\ r \end{bmatrix}, \tag{6.7}$$

where $H = (X^{-1}Z + \gamma^2 I)$ and $w = t - X^{-1}v$. We may now eliminate Δx from (6.7) and solve for Δy, or vice versa.

6.1.1 Solve for Δy

By letting $D = H^{-1/2}$ and $\Delta x = \delta D\overline{s}$, we have an equivalent system:

$$\begin{bmatrix} -\delta I & DA^T \\ AD & \delta I \end{bmatrix} \begin{bmatrix} \overline{s} \\ \Delta y \end{bmatrix} = \begin{bmatrix} Dw \\ r/\delta \end{bmatrix}. \tag{6.8}$$

This is also equivalent to the least squares problem

$$\min \left\| \begin{pmatrix} DA^T \\ \delta I \end{pmatrix} \Delta y - \begin{pmatrix} Dw \\ r/\delta \end{pmatrix} \right\|. \tag{6.9}$$

To solve (6.9) for Δy, we apply the iterative solver LSQR [36, 35]. Then we obtain Δx and Δz from

$$\begin{aligned} q &= A^T \Delta y \\ \Delta x &= D^2(q - w) \\ X\Delta z &= v - Z\Delta x. \end{aligned}$$

When an explicit A is available, this method is attractive because computing a diagonal preconditioner for (6.9) is easy. It is also preferred in cases where the dimension of Δy is much less than that of Δx, that is, A is fat ($m \ll n$). However, when A is available only as an operator, it is too expensive to find the norms of the columns of DA^T when D changes at each iteration. Instead, we recommend solving for Δx because we can obtain a diagonal preconditioner without difficulty.

6.1.2 Solve for Δx

When δ is not too small, we may eliminate Δy from (6.7) to obtain the least squares problem

$$\min \left\| \begin{pmatrix} AD \\ \delta I \end{pmatrix} s - \begin{pmatrix} r \\ -\delta Dw \end{pmatrix} \right\|, \tag{6.10}$$

where $\Delta x = Ds$. LSQR is used to compute s, and we obtain Δx, Δy and Δz from

$$\begin{aligned} \Delta x &= D\, s \\ \Delta y &= (r - A\Delta x)/\delta^2 \\ X\Delta z &= v - Z\Delta x. \end{aligned}$$

As shown in Section 6.5, diagonal preconditioning is easy if $\|A_{.j}\|$ can be obtained efficiently for all columns j.

The associated normal equations are equivalent to $(A^TA + \delta^2 H)\Delta x = A^T r - \delta^2 w$. When $\delta = 1$, this is the Newton equation for

$$\min \tfrac{1}{2}\|b - Ax\|^2 + \tfrac{1}{2}\|\gamma x\|^2 - \mu \sum \log x_j$$

with Z satisfying $Xz = \mu e$. Johnson and Sofer deal with this system to compute Δx [20], applying the symmetric conjugate gradient (CG) method with the analogous diagonal preconditioner.

We prefer solving the least squares problem (6.10) because LSQR tends to be more reliable than CG on the normal equations, and because it returns useful information about the accuracy of approximate solutions.

6.1.3 Algorithm

The main steps of PDSCO are as follows:

1. Set parameters: the feasibility tolerance `FeaTol`, the complementary gap tolerance `OptTol`, the regularization parameter γ.
2. Initialize $x > 0$, y, $z > 0$, and $\mu > 0$. Set $k = 0$.
3. (a) With $X = \text{diag}(x)$ and $Z = \text{diag}(z)$, set

$$\begin{aligned} r &= b - Ax - \delta^2 y, \\ t &= \gamma^2 x - A^T y - z, \\ v &= \mu e - Xz, \\ D &= (X^{-1}Z + \gamma^2 I)^{-1/2}, \\ w &= t - X^{-1}v. \end{aligned}$$

(b) Obtain Δx, Δy, Δz from

$$\begin{aligned} \min \left\| \begin{pmatrix} AD \\ \delta I \end{pmatrix} s - \begin{pmatrix} r \\ -\delta D w \end{pmatrix} \right\|, \\ \Delta x &= D\, s, \\ \Delta y &= (r - A\Delta x)/\delta^2, \\ \Delta z &= X^{-1}(v - Z\Delta x). \end{aligned}$$

(c) Calculate the primal and dual steplengths α_p, α_d and update the variables:

$$\begin{aligned}
\alpha_p &= \texttt{StepTol} \times \max\{\alpha : x + \alpha\Delta x \geq 0\} \\
\alpha_d &= \texttt{StepTol} \times \max\{\alpha : z + \alpha\Delta z \geq 0\} \\
\alpha &= \min(\alpha_p, \alpha_d) \\
x &= x + \alpha\Delta x, \quad y = y + \alpha\Delta y, \quad z = z + \alpha\Delta z \\
\mu &= \max((1 - \min(\alpha, \texttt{StepTol}))\mu, \mu_{\text{last}}),
\end{aligned}$$

where $\mu_{\text{last}} = 0.1 * \texttt{OptTol}$.

4. Terminate if the following three conditions are satisfied:

   (a) primal feasibility = $\|r\| <$ `FeaTol`.

   (b) dual feasibility = $\|t\| <$ `FeaTol`.

   (c) complementarity gap = $\|v\| <$ `OptTol`.

5. $k = k + 1$. Repeat from Step 3.

For fuller discussion of PDSCO, see [41].

6.2 Starting Point

Some methods for image restoration converge rapidly during early iterations even if they require thousands of iterations overall because they converge very slowly near the solution. We use these methods to obtain a good initial estimate for other methods. These points are called *warm* starting points, and they result in many fewer iterations than with the *cold* starting points that we use by default. According to computational experience, the SSD method shows fast convergence in the first few iterations. We use the output from SSD as the starting point for other methods such as PDSCO. Since SSD gives x only, we need to provide initial estimates for the dual variables y and z for PDSCO. The dual variable for (6.2) satisfies $\delta y = p$ with $\delta = 1$ here. We therefore initialize y to be the initial residual vector. The initial z comes from (6.4). Thus, we set

$$\begin{aligned}
x_0 &= x_{\text{ssd}} \\
y_0 &= b - Ax_0 \\
z_0 &= \gamma^2 x_0 - A^T y_0.
\end{aligned} \tag{6.11}$$

In interior methods, initial estimates near boundary values result in poor performance. We set $x_0 \leftarrow \max(x_0, \theta)$ and $z_0 \leftarrow \max(z_0, \theta)$, $\theta = 0.1$ in the experiments. For complementarity, we then set $[z_0]_j = 1$ for j such that $[x_0]_j \leq 0.05$. Our experience shows that a cold start works as well as a warm start for interior methods, but a warm start is better for other methods such as the reduced-gradient method.

6.3 Inexact Newton Methods

The Newton directions are defined by the system of equations (6.6). Although Newton's method converges rapidly from a sufficiently good starting point, solving a huge system of linear equations at each iteration is very expensive. Therefore, we consider an *inexact Newton method* [11], which solves the Newton equations approximately. For the nonlinear equation $F(x) = 0$, the k-th step of an inexact Newton method is $x_{k+1} = x_k + \alpha s_k$, where

$$F'(x_k)s_k = -F(x_k) + r_k, \quad \rho \equiv \|r_k\|/\|F(x_k)\| \leq \eta_k, \tag{6.12}$$

where $\{\eta_k\}$ is a sequence of scalars. Taking $\eta_k = 0$ gives an exact Newton method. As long as the sequence $\{\eta_k\}$ is uniformly less than one, it can be shown that inexact methods are locally convergent (Dembo, Eisenstat, and Steihaug [11]).

For problem (6.4), if we solve (6.9) inexactly for an approximate Δy, we may solve

$$\begin{bmatrix} -\gamma^2 I & I \\ Z & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta z \end{bmatrix} = \begin{bmatrix} t - A^T \Delta y \\ v \end{bmatrix} \equiv \begin{bmatrix} \overline{t} \\ v \end{bmatrix} \tag{6.13}$$

to obtain $\Delta x = (Z + \gamma^2 X)^{-1}(v - X\overline{t})$ and $\Delta z = \overline{t} + \gamma^2 \Delta x$. We now have an approximate $(\Delta x, \Delta y, \Delta z)$ that gives small residuals for two of the three equations in the Newton system (6.6). The important ratio in (6.12) is

$$\rho = \frac{\|r - \delta^2 \Delta y - A\Delta x\|}{\|(r, t, v)\|}. \tag{6.14}$$

We must ensure that this ratio is less than 1 for every major iteration. The residual vector

in the numerator is

$$\begin{aligned} r - \delta^2\Delta y - A\Delta x &= r - \delta^2\Delta y - A(Z+\gamma^2 X)^{-1}(v - X(t - A^T\Delta y)) \\ &= r - \delta^2\Delta y - A(X^{-1}Z+\gamma^2 I)^{-1}X^{-1}(v - X(t - A^T\Delta y)) \\ &= r - \delta^2\Delta y - AD^2X^{-1}(v - X(t - A^T\Delta y)) \\ &= r - AD^2(X^{-1}v - t) - (AD^2A^T + \delta^2 I)\Delta y \\ &= (r + AD^2w) - (AD^2A^T + \delta^2 I)\Delta y \\ &= \begin{pmatrix} AD & \delta I \end{pmatrix}\left(\begin{pmatrix} Dw \\ r/\delta \end{pmatrix} - \begin{pmatrix} DA^T \\ \delta I \end{pmatrix}\Delta y\right) \end{aligned}$$

and a reliable estimate of $\|r - \delta^2\Delta y - A\Delta x\|$ is returned as an output parameter `arnorm` when LSQR is applied to problem (6.9).

Likewise, if we solve (6.10) to get an approximate Δx, we may solve

$$\begin{bmatrix} \delta^2 I & 0 \\ 0 & X \end{bmatrix}\begin{bmatrix} \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r - A\Delta x \\ v - Z\Delta x \end{bmatrix} \tag{6.15}$$

to get a (different) approximate $(\Delta x, \Delta y, \Delta z)$ that also gives small residuals for two of the equations in (6.6). The important ratio is now

$$\rho = \frac{\|t + \gamma^2\Delta x - A^T\Delta y - \Delta z\|}{\|(r,t,v)\|}, \tag{6.16}$$

where

$$\begin{aligned} t + \gamma^2\Delta x - A^T\Delta y - \Delta z &= t + \gamma^2\Delta x - \frac{1}{\delta^2}A^T(r - A\Delta x) - X^{-1}(v - Z\Delta x) \\ &= (\frac{1}{\delta^2}A^TA + X^{-1}Z + \gamma^2 I)\Delta x + (t - X^{-1}v) - A^Tr/\delta^2 \\ &= (1/\delta^2)D^{-1}\left((D^{-1} + DA^TA)\Delta x + \delta^2 Dw - DA^Tr\right) \\ &= (1/\delta^2)D^{-1}\left((DA^TAD + \delta^2 I)s + \delta^2 Dw - DA^Tr\right) \\ &= (1/\delta^2)D^{-1}\left(\begin{pmatrix} DA^T & \delta I \end{pmatrix}\left(\begin{pmatrix} AD \\ \delta I \end{pmatrix}s - \begin{pmatrix} r \\ -\delta Dw \end{pmatrix}\right)\right). \end{aligned}$$

In this case, the parameter `arnorm` returns a reliable estimate of

$$\left\|\begin{pmatrix} DA^T & \delta I \end{pmatrix}\left(\begin{pmatrix} r \\ -\delta Dw \end{pmatrix} - \begin{pmatrix} AD \\ \delta I \end{pmatrix}s\right)\right\|$$

when LSQR is applied to problem (6.10). Unfortunately this does not include the effect of

D^{-1}. However, diagonal preconditioning has a separate (canceling) effect of its own. The implemented test has performed as expected.

When Δy is computed by LSQR, we want ρ in (6.14) to be less than a certain level (for example, 0.01) throughout the iterations for rapid convergence. If it is greater than the pre-determined level but acceptable, we solve LSQR more accurately next iteration by decreasing `atol`. If it is excessively large, then we continue computing the direction more accurately with a decreased `atol`. In summary, we control this quantity as follows:

$$\begin{array}{rcl} \rho \leq 0.1 & : & \text{accept the direction.} \\ 0.1 < \rho \leq 0.5 & : & \text{accept the direction and decrease } \texttt{atol}. \\ \rho > 0.5 & : & \text{recompute the direction with a decreased } \texttt{atol}. \end{array}$$

Similar tests are used when Δx is computed by LSQR.

6.4 Scaling

Scaling means changing the units of measurement to improve the numerical stability of an algorithm. The variables are transformed as $\overline{x} = Sx$, where $S = \mathrm{diag}(s)$. The diagonal elements are the scale values, which are positive: $s_1, \ldots, s_n > 0$. Here we consider scaling the following convex optimization problem:

$$\begin{array}{ll} \underset{x,\,y}{\text{minimize}} & \varphi(x,y) = f(x) + \frac{1}{2}\|\gamma x\|^2 + \frac{1}{2}\|\delta y\|^2 \\ \text{subject to} & Ax + \delta^2 y = b, \\ & x \geq 0, \end{array}$$

so that the scaled primal vector and dual vector are balanced in magnitude. Since we wish to obtain $\|\overline{x}\|_\infty \approx \|\overline{z}\|_\infty \approx 1$, we estimate $\beta \approx \|x\|_\infty, \zeta \approx \|z\|_\infty$ and define

$$x = \beta\overline{x}, \quad b = \beta\overline{b}, \quad y = \zeta\overline{y}, \quad z = \zeta\overline{z}, \quad \varphi = \theta\overline{\varphi}$$

for some θ to be chosen. Then the scaled objective $\overline{\varphi}$ becomes

$$\begin{aligned} \overline{\varphi} &= \frac{1}{\theta} f(\beta\overline{x}) + \frac{1}{2\theta}\|\gamma\beta\overline{x}\|^2 + \frac{1}{2\theta}\|\delta\zeta\overline{y}\|^2 \\ &= \frac{1}{\theta} f(\beta\,\overline{x}) + \frac{1}{2}\|\overline{\gamma}\,\overline{x}\|^2 + \frac{1}{2}\|\overline{\delta}\,\overline{y}\|^2, \end{aligned}$$

where $\overline{\gamma} = \dfrac{\gamma\beta}{\sqrt{\theta}}$, $\overline{\delta} = \dfrac{\delta\zeta}{\sqrt{\theta}}$. The constraint $Ax + \delta^2 y = b$ becomes $A\overline{x} + \overline{\delta}^2\overline{y} = \overline{b}$. We need to choose θ so that $\dfrac{\delta^2\zeta}{\beta} = \overline{\delta}^2 = \dfrac{\delta^2\zeta^2}{\theta}$. Hence we have

$$\theta = \beta\zeta, \quad \overline{f} = \frac{1}{\theta} f(\beta\overline{x}), \quad \nabla\overline{f} = \frac{\beta}{\theta}\nabla f, \quad \nabla^2\overline{f} = \frac{\beta^2}{\theta}\nabla^2 f. \tag{6.17}$$

We show the effectiveness of the scaling in Table 6.1 and 6.2, where the smaller version of the satellite problem is being solved by PDSCO. The quantities shown are `Pinf` $= \log_{10}(r)$, `Dinf` $= \log_{10}(t)$, `Cinf` $= \log_{10}(v)$, `center` $= \frac{\max(x_j z_j)}{\min(x_j z_j)}$, `atol` = tolerance for LSQR, `LSQR` = the number of LSQR iterations, and `Inexact` is the ρ in (6.16). As we see, the scaled problem stays more well-centered, and the total number of LSQR iterations for the unscaled problem is 1152 compared with 538 iterations for the scaled problem.

Itn	mu	stepx	stepz	Pinf	Dinf	Cinf	Objective	nf	center	atol	LSQR	Inexact
0				-99.0	-16.4	-0.1	1.1628407e+03		6.1			
1	-1.0	1.000	1.000	-99.0	-16.4	-0.6	3.7294237e+02	1	2.9	-3.0	4	0.000
2	-2.0	1.000	0.772	-1.2	-5.0	-1.1	1.4544815e+02	1	7.4	-3.0	4	0.001
3	-2.6	1.000	0.623	-1.2	-4.5	-1.5	6.4472480e+01	1	14.7	-3.0	5	0.002
4	-3.1	1.000	0.524	-1.2	-4.4	-1.8	3.1500182e+01	1	10.2	-3.0	7	0.001
5	-3.4	1.000	0.481	-1.2	-4.2	-2.1	1.5751985e+01	1	10.0	-3.0	7	0.002
6	-3.7	1.000	0.548	-1.4	-4.3	-2.4	6.7685852e+00	1	22.4	-3.0	9	0.002
7	-4.0	1.000	0.478	-1.4	-4.3	-2.7	3.3361734e+00	1	17.7	-3.0	12	0.002
8	-4.3	0.268	0.268	-1.5	-4.4	-2.8	2.3002228e+00	2	123.5	-3.0	12	0.002
9	-4.4	1.000	0.423	-1.5	-4.2	-3.0	1.2764030e+00	1	28.7	-3.0	13	0.003
10	-4.7	0.203	0.203	-1.6	-4.3	-3.1	9.7557734e−01	2	223.7	-3.0	14	0.003
11	-4.8	0.254	0.254	-1.8	-4.4	-3.2	7.0448354e−01	2	55.9	-3.0	15	0.002
12	-4.9	0.195	0.195	-1.8	-4.5	-3.3	5.5977188e−01	2	115.8	-3.0	17	0.003
13	-5.0	0.040	0.040	-1.9	-4.6	-3.3	5.3606613e−01	2	700.3	-3.0	18	0.002
14	-5.0	0.012	0.012	-1.9	-4.6	-3.4	5.2941648e−01	2	2298.5	-3.0	16	0.003
15	-5.0	0.002	0.002	-1.9	-4.6	-3.4	5.2823828e−01	2	22945.0	-3.0	17	0.004
16	-5.0	1.000	0.523	-1.9	-4.4	-3.7	2.8350268e−01	1	57.9	-4.0	35	0.000
17	-5.3	0.300	0.300	-2.1	-4.6	-3.8	2.0787815e−01	2	201.6	-4.0	35	0.000
18	-5.5	0.303	0.303	-2.2	-4.7	-4.0	1.5824250e−01	2	100.0	-4.0	39	0.000
19	-5.6	0.127	0.127	-2.3	-4.8	-4.0	1.4360172e−01	2	879.3	-4.0	42	0.000
20	-5.7	0.019	0.019	-2.3	-4.8	-4.0	1.4165271e−01	2	8629.3	-4.0	43	0.001
21	-5.7	0.002	0.002	-2.3	-4.8	-4.0	1.4140521e−01	2	86085.9	-4.0	44	0.001
22	-5.7	0.300	0.300	-2.5	-4.9	-4.2	1.1416731e−01	2	114.5	-5.0	70	0.000
23	-5.9	0.436	0.436	-2.7	-5.2	-4.4	8.8369869e−02	2	39.1	-5.0	72	0.000
24	-6.0	0.296	0.296	-2.9	-5.3	-4.5	7.8042598e−02	2	35.2	-5.0	76	0.000
25	-6.0	0.451	0.451	-3.1	-5.6	-4.7	6.7655212e−02	2	46.8	-5.0	79	0.000
26	-6.0	0.393	0.393	-3.3	-5.8	-4.9	6.2502872e−02	2	18.2	-5.0	87	0.000
27	-6.0	0.436	0.436	-3.6	-6.1	-5.1	5.9002983e−02	2	16.8	-5.0	86	0.000
28	-6.0	0.432	0.432	-3.8	-6.3	-5.3	5.6951603e−02	2	11.9	-5.0	88	0.000
29	-6.0	0.562	0.562	-4.2	-6.7	-5.5	5.5383367e−02	2	12.4	-5.0	89	0.000
30	-6.0	1.000	1.000	-18.1	-19.3	-5.8	5.4072302e−02	1	5.5	-5.2	97	0.000
Total											**1152**	

Table 6.1: Run *without* scaling (but with diagonal preconditioning)

Itn	mu	stepx	stepz	Pinf	Dinf	Cinf	Objective	nf	center	atol	LSQR	Inexact
0				-0.8	-0.8	0.0	8.2674621e+02		6.1			
1	-1.0	1.000	1.000	-16.0	-16.2	-0.5	3.5682884e+02	1	3.2	-3.0	4	0.000
2	-2.0	1.000	0.741	-1.2	-4.9	-1.0	1.4511449e+02	1	8.1	-3.0	4	0.002
3	-2.6	1.000	0.625	-1.2	-4.5	-1.4	6.3273184e+01	1	15.7	-3.0	5	0.002
4	-3.0	1.000	0.517	-1.2	-4.3	-1.7	3.1201876e+01	1	10.9	-3.0	7	0.002
5	-3.3	1.000	0.484	-1.2	-4.1	-2.0	1.5472046e+01	1	10.1	-3.0	7	0.002
6	-3.6	1.000	0.550	-1.4	-4.2	-2.3	6.6149694e+00	1	17.2	-3.0	9	0.002
7	-4.0	1.000	0.487	-1.4	-4.2	-2.6	3.2095337e+00	1	16.6	-3.0	12	0.002
8	-4.3	1.000	0.287	-1.3	-4.0	-2.7	2.1518600e+00	1	131.3	-3.0	12	0.003
9	-4.4	0.137	0.137	-1.4	-4.1	-2.8	1.7747360e+00	2	230.3	-3.0	12	0.002
10	-4.5	1.000	0.353	-1.5	-4.0	-2.9	1.0579179e+00	1	1659.1	-3.0	13	0.002
11	-4.5	0.009	0.009	-1.5	-4.0	-2.9	1.0457302e+00	2	16459.2	-3.0	12	0.002
12	-4.5	1.000	0.608	-1.8	-4.2	-3.4	4.0730746e−01	1	49.6	-4.0	29	0.000
13	-4.9	1.000	0.478	-1.9	-4.4	-3.6	2.3352634e−01	1	23.5	-4.0	32	0.000
14	-5.2	1.000	0.366	-2.0	-4.3	-3.8	1.6254847e−01	1	17.6	-4.0	38	0.000
15	-5.4	1.000	0.335	-2.1	-4.4	-3.9	1.2124483e−01	1	21.9	-4.0	36	0.000
16	-5.5	1.000	0.545	-2.3	-4.6	-4.3	8.1647542e−02	1	12.0	-4.0	36	0.000
17	-5.9	1.000	0.418	-2.4	-4.8	-4.5	6.8841884e−02	1	10.1	-4.0	44	0.000
18	-6.0	1.000	0.473	-2.5	-4.9	-4.7	6.0947055e−02	1	12.7	-4.0	43	0.000
19	-6.0	1.000	0.455	-2.7	-5.1	-5.0	5.7341905e−02	1	12.7	-4.0	45	0.000
20	-6.0	1.000	0.457	-2.9	-5.3	-5.2	5.5540011e−02	1	10.6	-4.0	43	0.000
21	-6.0	1.000	0.612	-3.2	-5.5	-5.5	5.4360280e−02	1	23.1	-4.0	45	0.000
22	-6.0	1.000	1.000	-17.5	-19.3	-5.9	5.3686115e−02	1	2.4	-4.2	50	0.000
Total											**538**	

Table 6.2: Run *with* scaling (and diagonal preconditioning)

6.5 Diagonal Preconditioners

In this section we seek preconditioners for systems (6.9)–(6.10) to accelerate the computation of Δy and Δx respectively. The simplest diagonal preconditioning requires the 2-norm of each column of the matrices

$$\begin{pmatrix} DA^T \\ \delta I \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} AD \\ \delta I \end{pmatrix}$$

(where D changes at every iteration of the interior method). For the first system, the norms can be found efficiently if A is available explicitly as a sparse matrix, but they are impractical when A is an operator. However, for column j of the second system we have

$$\left\| \begin{pmatrix} AD \\ \delta I \end{pmatrix} e_j \right\|^2 = \|ADe_j\|^2 + \delta^2 = d_j^2 \|Ae_j\|^2 + \delta^2, \tag{6.18}$$

and from the structure of A and the PSF, $\|Ae_j\|$ can be easily computed. Thus, (6.10) is preferable to (6.9) because we can compute the preconditioner easily. Table 6.3 shows the run without preconditioner using the smaller satellite problem. Compared to Table 6.2 with the diagonal preconditioner, the number of LSQR iterations increases from 538 to 763.

Itn	mu	stepx	stepz	Pinf	Dinf	Cinf	Objective	nf	center	atol	LSQR	Inexact
0				-0.8	-0.8	0.0	8.2674621e+02		6.1			
1	-1.0	1.000	1.000	-16.0	-16.2	-0.5	3.5682878e+02	1	3.2	-3.0	4	0.000
2	-2.0	1.000	0.741	-1.2	-4.9	-1.0	1.4504864e+02	1	8.1	-3.0	4	0.003
3	-2.6	1.000	0.635	-1.2	-4.5	-1.4	6.2266331e+01	1	15.5	-3.0	6	0.006
4	-3.0	1.000	0.514	-1.2	-4.3	-1.7	3.0934104e+01	1	11.6	-3.0	7	0.027
5	-3.3	1.000	0.516	-1.2	-4.1	-2.0	1.4513118e+01	1	11.6	-3.0	10	0.030
6	-3.7	1.000	0.554	-1.4	-4.1	-2.4	6.2262601e+00	1	24.8	-3.0	12	0.041
7	-4.0	1.000	0.360	-1.3	-4.1	-2.6	3.7562774e+00	1	142.4	-3.0	15	0.070
8	-4.2	1.000	0.455	-1.4	-4.1	-2.8	1.8557507e+00	1	35.2	-3.0	17	0.064
9	-4.5	0.193	0.193	-1.5	-4.2	-2.9	1.4219929e+00	2	289.6	-3.0	17	0.116
10	-4.6	1.000	0.487	-1.6	-4.0	-3.2	7.0066193e−01	1	31.6	-4.0	30	0.018
11	-4.8	1.000	0.485	-1.8	-4.2	-3.5	3.6399419e−01	1	19.6	-4.0	32	0.020
12	-5.1	1.000	0.430	-1.9	-4.3	-3.7	2.1896896e−01	1	8.4	-4.0	35	0.026
13	-5.4	1.000	0.323	-2.0	-4.3	-3.9	1.5785171e−01	1	8.0	-4.0	38	0.027
14	-5.5	1.000	0.386	-2.1	-4.4	-4.1	1.1103480e−01	1	13.4	-4.0	38	0.027
15	-5.8	1.000	0.480	-2.3	-4.6	-4.3	7.9964253e−02	1	45.8	-4.0	40	0.031
16	-6.0	1.000	0.437	-2.4	-4.8	-4.6	6.6948928e−02	1	48.9	-4.0	44	0.039
17	-6.0	1.000	0.423	-2.5	-4.9	-4.8	6.0623295e−02	1	43.6	-4.0	44	0.042
18	-6.0	0.102	0.102	-2.6	-5.0	-4.8	5.9754672e−02	2	396.8	-4.0	44	0.047
19	-6.0	0.116	0.116	-2.6	-5.0	-4.9	5.8910537e−02	2	444.0	-4.0	47	0.038
20	-6.0	1.000	0.464	-2.8	-5.2	-5.1	5.6315761e−02	1	40.1	-4.0	47	0.045
21	-6.0	1.000	0.649	-3.2	-5.5	-5.4	5.4568375e−02	1	8.8	-4.0	50	0.043
22	-6.0	1.000	0.985	-4.9	-7.2	-5.8	5.3712679e−02	1	7.9	-4.2	55	0.035
23	-6.0	1.000	1.000	-17.8	-19.3	-6.0	5.3676186e−02	1	1.1	-5.9	127	0.003
Total											**763**	

Table 6.3: Run with scaling but *without* preconditioner

6.6 Regularization

Regularization is a process that perturbs a problem a little to make it easier to solve. Benefits include uniqueness of the solution and algorithmic simplicity. In practice, moderate regularization of any problem produces most of the benefits. All regularization methods for computing stable solutions to ill-posed problems involve a trade-off between the *size* of the regularized solution and the quality of the fit that it provides to the given data.

Regularization is necessary with ill-posed problems because the (unregularized) *naive* solution is completely dominated by contributions from data errors and rounding errors. If too much regularization or damping is imposed on the solution, then it will not fit the given data b properly, and the residual $\|Ax - b\|$ will be too large. Thus, the reconstruction is too smooth. On the other hand, if too little regularization is imposed, reconstructions have highly oscillatory artifacts that are due to noise amplification.

Several regularizations include:

- Tikhonov regularization: $\frac{1}{2}\alpha\|x\|^2$.
- Entropy-type penalty term: $\alpha \sum_{i=1}^{n} x_j \log x_j, \quad x > 0.$

- The 1-norm: $\|x\|_1$ ($= e^T x$ because of the nonnegativity of x).

The first two are discussed in Hanke [18] and the third is explained in Chen [8].

We use $\frac{1}{2}\|\gamma x\|^2$ for our constrained least squares problem, which is basically the same as Tikhonov regularization. The scalar γ is called a regularization parameter. Choosing the appropriate γ for each problem is not trivial. Thus we determine the value of γ by computational experiments. In any case, γ should be chosen so that the system (6.7) can be solved more easily. In particular, we pick a value of γ such that $H = (X^{-1}Z + \gamma^2 I)$ is sufficiently removed from singularity. The typical value of γ for an image restoration problem is 10^{-3} or 10^{-2}. Regularization parameter selection methods are discussed extensively in [47].

We present the effect of regularization by changing the value of γ in Tables 6.4 and 6.5. Despite the fact that the number of LSQR iterations increases drastically as γ decreases, the quality of the restored image does not improve. In fact, Table 6.5 shows that the quality of the restored image degenerates if γ is too small.

	x^*	$\gamma = 10^{-1}$	$\gamma = 10^{-2}$	$\gamma = 10^{-3}$	$\gamma = 10^{-4}$
Iterations		199	645	2827	9258
$\|b - Ax\|_2$	18283.02	8242.89	1707.24	1646.65	1645.47
$\max x_j$	19426.66	15494.93	21928.85	34054.35	39597.76
$\min x_j$	0.0	0.072	0.22	0.20	0.20

Table 6.4: Statistics for PDSCO restored images for each γ

	x^*	$\gamma = 10^{-1}$	$\gamma = 10^{-2}$	$\gamma = 10^{-3}$	$\gamma = 10^{-4}$
[1e+04 , 1e+05)	264	222	257	228	223
[1e+03 , 1e+04)	265	538	270	248	236
[100 , 1e+03)	69	331	238	216	218
[10 , 100)	5	754	434	430	451
[1 , 10)	2	1059	1880	1973	1966
[0.1 , 1)	0	1187	1017	1001	1002
[0.01 , 0.1)	0	0	0	0	0
[0.001 , 0.01)	0	0	0	0	0
[0 , 1e-03)	3491	0	0	0	0

Table 6.5: Distribution of x^* and restored images for different γ

The restored images for each γ are shown in Figure 6.1. The image with $\gamma = 10^{-2}$ looks the best among them. The proper value of γ is usually determined by experiment.

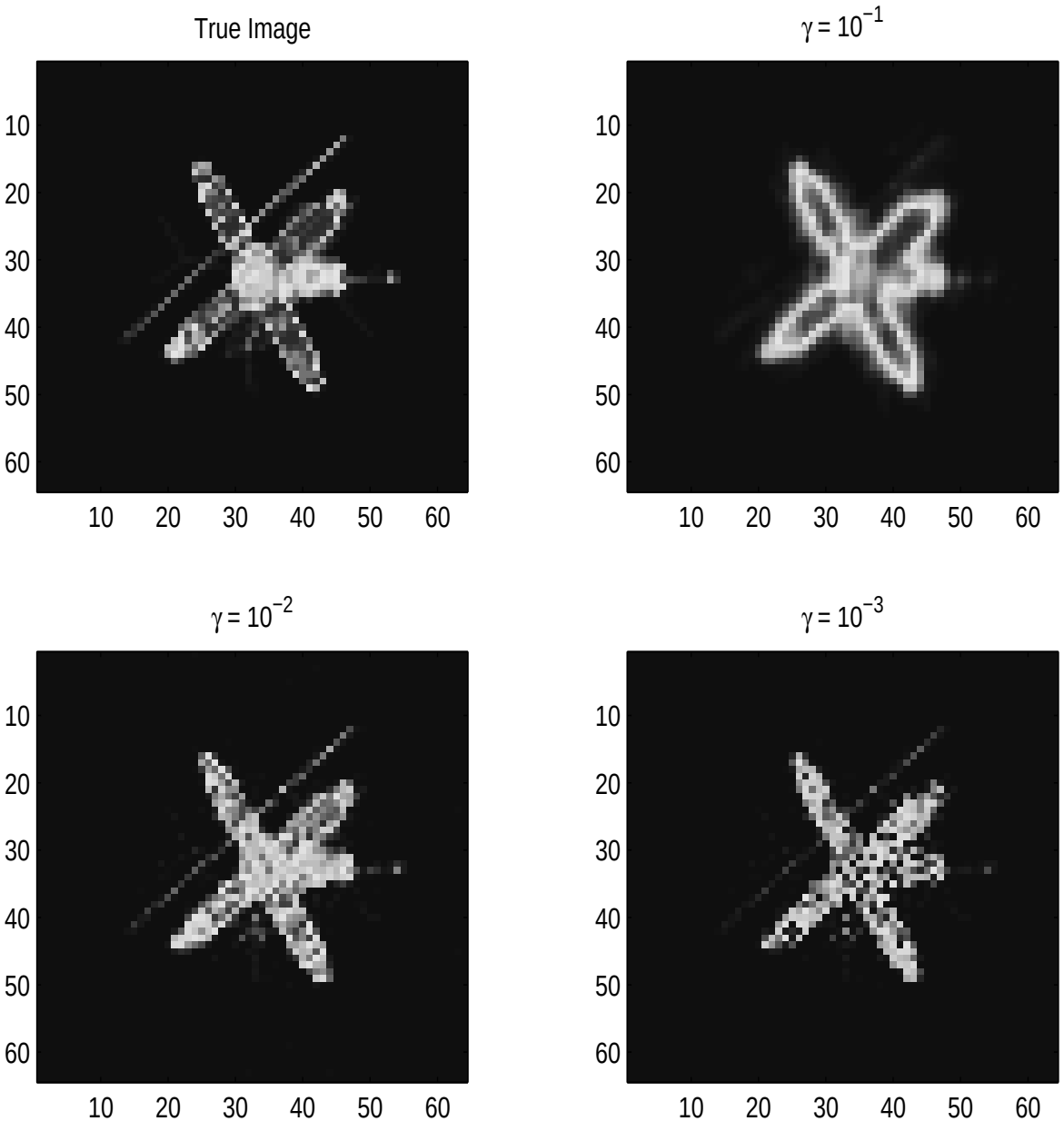

Figure 6.1: True and PDSCO restoration for each γ

6.7 Predictor-Corrector Method

Most implementations of interior methods for LP are based on Mehrotra's predictor-corrector algorithm [50]. This enhances the basic Newton search direction with a correction that has been very successful in LP with direct solvers. Since it solves two linear systems with the same matrix but different right-hand sides, the marginal cost is not great when we have sparse factors of $AD^2A^T + \delta^2 I$ or the KKT-type system (6.7).

In our case, we use an iterative solver because A is an operator and sparse factors are not available. Since there are no factors to re-use, the total work to solve two linear systems with different right-hand sides is twice as expensive as solving one linear system.

We find that predictor-corrector does help to reduce the number of iterations in general, but perhaps not the total work. We show a predictor-corrector run for the smaller satellite problem in Table 6.6. The infeasibilities decrease reasonably quickly, but the number of LSQR iterations for the last several iterations is almost twice as many as for the run in Table 6.2.

Itn	mu	stepx	stepz	Pinf	Dinf	Cinf	Objective	nf	center	atol	LSQR	Inexact
0				-1.1	0.0	-1.0	3.8463707e−01		10.0			
1	-1.4	0.706	0.709	-1.6	-0.5	-1.1	1.7371183e−01	1	39.7	-3.0	41	0.033
2	-1.8	0.884	0.512	-2.3	-0.8	-1.4	1.2935003e−01	1	71.7	-3.0	43	0.049
3	-2.4	1.000	0.416	-2.3	-1.0	-1.5	9.9007024e−02	1	122.8	-3.0	39	0.074
4	-2.8	1.000	0.190	-2.4	-1.1	-1.6	9.0619491e−02	1	131.6	-3.0	35	0.242
5	-2.9	1.000	0.051	-2.4	-1.2	-1.6	8.9042778e−02	1	835.7	-3.0	33	0.238
6	-2.9	1.000	0.005	-2.4	-1.2	-1.6	8.8917011e−02	1	5457.1	-3.0	33	0.240
7	-2.9	1.000	0.482	-2.6	-1.4	-1.9	7.3476884e−02	1	416.4	-4.0	78	0.031
8	-3.1	1.000	0.796	-3.3	-2.1	-2.6	6.2338762e−02	1	189.5	-4.0	78	0.036
9	-3.4	0.986	0.611	-3.1	-2.5	-2.9	5.6486357e−02	1	28.8	-4.0	103	0.044
10	-3.9	1.000	0.645	-3.4	-3.0	-3.3	5.4050416e−02	1	24.4	-4.0	102	0.065
11	-4.3	1.000	0.946	-5.2	-4.3	-3.9	5.3997211e−02	1	3.6	-4.0	64	0.007
12	-5.1	0.536	0.536	-5.6	-4.6	-4.3	5.3392333e−02	2	4.8	-4.9	81	0.019
13	-5.4	0.393	0.393	-5.8	-4.8	-4.5	5.3101542e−02	2	9.5	-5.3	94	0.011
14	-5.6	0.280	0.280	-5.9	-4.9	-4.6	5.2946902e−02	2	14.1	-5.5	109	0.008
15	-5.8	0.350	0.350	-6.1	-5.1	-4.8	5.2791394e−02	2	15.5	-5.6	115	0.006
16	-6.0	0.305	0.305	-6.3	-5.3	-5.0	5.2684438e−02	2	15.6	-5.8	123	0.006
17	-6.0	0.280	0.280	-6.4	-5.4	-5.1	5.2607105e−02	2	20.9	-6.0	129	0.004
Total											**1300**	

Table 6.6: Run using predictor-corrector algorithm

6.8 Two-level NNLS

Here we are still exploring the use of PDSCO's primal-dual interior method. Recall that the search directions are computed iteratively using LSQR. For the first few search directions the number of LSQR iterations is typically small, but as the NNLS solution is approached

the iterations tend to become excessive. Table 6.7 shows the general pattern of a PDSCO run. As μ gets smaller, the LSQR iterations increase steadily and the last three iterations account for about 70% of the total.

Itn	mu	stepx	stepz	Pinf	Dinf	Cinf	Objective	nf	center	atol	LSQR	Inexact
0				-0.2	-0.4	-0.1	3.7019139e+04		8.3			
1	-2.0	0.510	0.698	-0.6	-0.9	-0.5	2.4038080e+04	1	171.1	-6.0	**37**	0.000
2	-2.0	0.866	0.586	-1.3	-1.3	-1.0	9.8315290e+03	1	174.5	-6.0	**38**	0.000
3	-2.0	0.972	0.555	-1.6	-1.7	-1.3	7.0197681e+03	1	85.3	-6.0	**53**	0.000
4	-2.4	0.944	0.708	-1.9	-2.2	-1.8	4.7594066e+03	1	92.7	-6.0	**67**	0.001
5	-2.9	0.706	0.373	-1.9	-2.4	-2.0	3.7381865e+03	1	139.9	-6.0	**102**	0.003
6	-2.9	0.635	0.635	-2.3	-2.9	-2.3	3.4900512e+03	1	63.2	-6.0	**123**	0.003
7	-3.3	0.673	0.589	-2.6	-3.2	-2.6	3.1584890e+03	1	80.0	-6.0	**139**	0.013
8	-3.7	0.594	0.566	-2.9	-3.6	-2.9	2.9944344e+03	1	117.8	-6.0	**172**	0.034
9	-3.7	0.881	0.881	-3.4	-4.5	-3.4	2.8955504e+03	1	48.9	-6.0	**236**	0.079
10	-4.6	0.586	0.540	-3.4	-4.9	-3.6	2.8452653e+03	1	189.8	-6.0	**280**	0.193
11	-4.6	0.989	0.989	-4.1	-6.8	-4.2	2.8103661e+03	1	38.8	-7.0	**465**	0.059
12	-6.6	0.907	0.905	-4.1	-7.8	-4.7	2.8012345e+03	1	944.0	-7.0	**833**	0.397
13	-6.6	1.000	1.000	-4.8	-16.0	-5.3	2.8002690e+03	1	36.3	-8.0	**1392**	0.108
14	-7.0	1.000	1.000	-6.2	-16.0	-5.9	2.8001970e+03	1	13.3	-9.0	**2293**	0.075
15	-7.0	1.000	1.000	-6.9	-16.0	-6.4	2.8001962e+03	1	3.8	-9.0	**1861**	0.089
Total											**8091**	

Table 6.7: General pattern of PDSCO run

We therefore propose as a remedy a reduced least squares problem (with fewer rows and columns but a more complex operator) by exploiting the following facts:

- In some applications, such as astronomy, very few elements x_j are nonzero. The star image, for example, contains only 470 nonzeros out of 65536 pixels (see Table 2.1).
- Most of the nonzeros are identified during the early the primal-dual iterations, when LSQR requires very few iterations.
- Once they are known, we can treat them as unconstrained because we expect that they will be far away from zero at a solution.
- The variables can be partitioned, and unconstrained least squares problems may be solved in stages.

After a few primal-dual iterations, we effectively have the problem

$$\begin{array}{ll} \underset{x_1,\,x_2}{\text{minimize}} & \dfrac{1}{2}\left\| \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b \right\|^2 \\ \text{subject to} & x_2 \geq 0, \end{array} \tag{6.19}$$

where A_1 has many rows, but few columns (a tall matrix), and where x_1 and x_2 are the partitions of x satisfying $\|x_2\| \ll \|x_1\|$. We can afford to compute a dense QR factorization

$$QA_1 = \begin{pmatrix} Y^T \\ Z^T \end{pmatrix} A_1 = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{6.20}$$

where Householder transformations allow both Y and Z to be represented in the same storage as A_1. Details about Householder transformations can be found in [16]. If A_1 is extremely sparse, we could also use the sparse QR factorization of Matstoms and Adlers [27, 1], which maintains sparsity in both Q and R. Problem (6.19) becomes

$$\boxed{\begin{array}{ll} \underset{x_1,\,x_2}{\text{minimize}} & \dfrac{1}{2}\left\| \begin{pmatrix} R & Y^TA_2 \\ 0 & Z^TA_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} Y^Tb \\ Z^Tb \end{pmatrix} \right\|^2 \\ \text{subject to} & x_2 \geq 0, \end{array}} \tag{6.21}$$

which is solved by the reduced problem

$$\boxed{\begin{array}{ll} \underset{x_2}{\text{minimize}} & \frac{1}{2}\left\| Z^TA_2x_2 - Z^Tb \right\|^2 \\ \text{subject to} & x_2 \geq 0, \end{array}} \tag{6.22}$$

followed by $Rx_1 = Y^T(b - A_2x_2)$. Problem (6.22) has fewer rows and columns but a more complex operator Z^TA_2. We can now restart the primal-dual method on (6.22), knowing that

- $\|Z^Tb\| \ll \|b\|$.
- $\|x_2\| \ll \|x_1\|$.
- Early primal-dual iterations will identify the largest elements of x_2 with relatively few LSQR iterations, in the same way that early iterations on the full problem identify x_1.
- We can solve the second level problem with a loose tolerance.

This process could be repeated if necessary, giving a multi-level procedure for computing the important elements of x in "batches", with each batch requiring few LSQR iterations. The storage required grows linearly with the number of large elements in x. Since we do not wish to store A and Q explicitly, we show how to compute the products Yv, Zw, Z^Ty,

$Y^T y$, $A_2 x_2$ and $A_2^T y$ for arbitrary vectors v, w, x_2 and y:

$$\begin{aligned}
Yv &= Q^T \begin{pmatrix} I \\ 0 \end{pmatrix} v = Q^T \begin{pmatrix} v \\ 0 \end{pmatrix} \\
Zw &= Q^T \begin{pmatrix} 0 \\ I \end{pmatrix} w = Q^T \begin{pmatrix} 0 \\ w \end{pmatrix} \\
Y^T y &= \begin{pmatrix} I & 0 \end{pmatrix} Q\, y \\
Z^T y &= \begin{pmatrix} 0 & I \end{pmatrix} Q\, y \\
A_2\, x_2 &= \begin{pmatrix} A_1 & A_2 \end{pmatrix} \begin{pmatrix} 0 \\ x_2 \end{pmatrix} = A \begin{pmatrix} 0 \\ x_2 \end{pmatrix} \\
A_2^T\, y &= \begin{pmatrix} 0 & I \end{pmatrix} \begin{pmatrix} {A_1}^T \\ {A_2}^T \end{pmatrix} y = \begin{pmatrix} 0 & I \end{pmatrix} (A^T y).
\end{aligned}$$

The right-hand side b may be expressed as

$$b = Yb_Y + Zb_Z.$$

Given Q, we can determine b_Y and b_Z by forming Qb, since

$$Qb = \begin{bmatrix} Y^T \\ Z^T \end{bmatrix} (Yb_Y + Zb_Z) = \begin{bmatrix} b_Y \\ b_Z \end{bmatrix}. \tag{6.23}$$

Conversely, given b_Y and b_Z, b is obtained as follows:

$$Q^T \begin{bmatrix} b_Y \\ b_Z \end{bmatrix} = \begin{bmatrix} Y & Z \end{bmatrix} \begin{bmatrix} b_Y \\ b_Z \end{bmatrix} = Yb_Y + Zb_Z \equiv b.$$

From the orthogonality of Q, we obtain

$$\|Q\, b\|^2 = \|b\|^2 = \|Y^T b\|^2 + \|Z^T b\|^2 = \|b_Y\|^2 + \|b_Z\|^2$$

and, using $\|Z^T b\|^2 \ll \|b\|^2$, we have

$$\|Y^T b\|^2 = \|b\|^2 - \|Z^T b\|^2 \approx \|b\|^2.$$

In practice, A_1 contains a huge number of small values, so we can suppress values smaller than a pre-determined tolerance and work with approximate QR factors: $A_1 \approx \overline{A}_1 = \overline{QR}$.

We then recover x_1 by solving

$$\underset{y}{\text{minimize}} \ \|A_1 R^{-1} y - \overline{b}\|^2, \quad Rx_1 = y,$$

where $\overline{b} = b - A_2 x_2$. We can also obtain an approximation of A_1 using the singular values of A_1 if they are clustered well [16].

For the sparse QR factorization, we have to consider the ordering of the given matrix A_1 to reduce the fill-in. Since the size of A_1 for some images is tremendous, sparse-matrix technology must be employed. The important thing is to choose an adequate tolerance for $\overline{A}_1$. From the distribution of the PSF, we can judge the cutoff value. For example, 10^{-4} is a good cutoff value for the star image.

The results of the two-level method for the star image are given in Table 6.8 and Figure 6.2. They show that the two-level method allows us to detect many important pixels that we cannot recover with a single application of PDSCO unless we solve the single problem to high accuracy. The number of pixels whose value is between 100 and 1e+3 increases from 315 to 354, which shows more stars in the two-level restored image.

	x^*	Single	Two-level
[1e+04 , 1e+05)	5	4	5
[1e+03 , 1e+04)	84	85	85
[100 , 1e+03)	381	**315**	**354**
[10 , 100)	0	454	1774
[1 , 10)	0	36002	2116
[0.1 , 1)	0	28676	2099
[0.01 , 0.1)	0	0	41542
[0.001 , 0.01)	0	0	17539
[0.0001 , 0.001)	0	0	19
[0 , 1e−04)	65066	0	0

Table 6.8: Distribution of single and two-level restoration for star image

The two-level method also produces a better solution in terms of the 2-norm of the residual vector and the range of the pixels values, as shown in Table 6.9.

Note that we drop the nonnegativity constraints $x_1 \geq 0$ in (6.21) by assuming that we guess a correct set of variables for x_1 that really are far from zero at a solution. They are uniquely defined for any x_2. Hence, we must check that the computed x_1 is really nonnegative. If not, we may have to re-do the procedure with more conservative decisions on how to choose A_1. However, the negative elements may be negligibly small, as shown

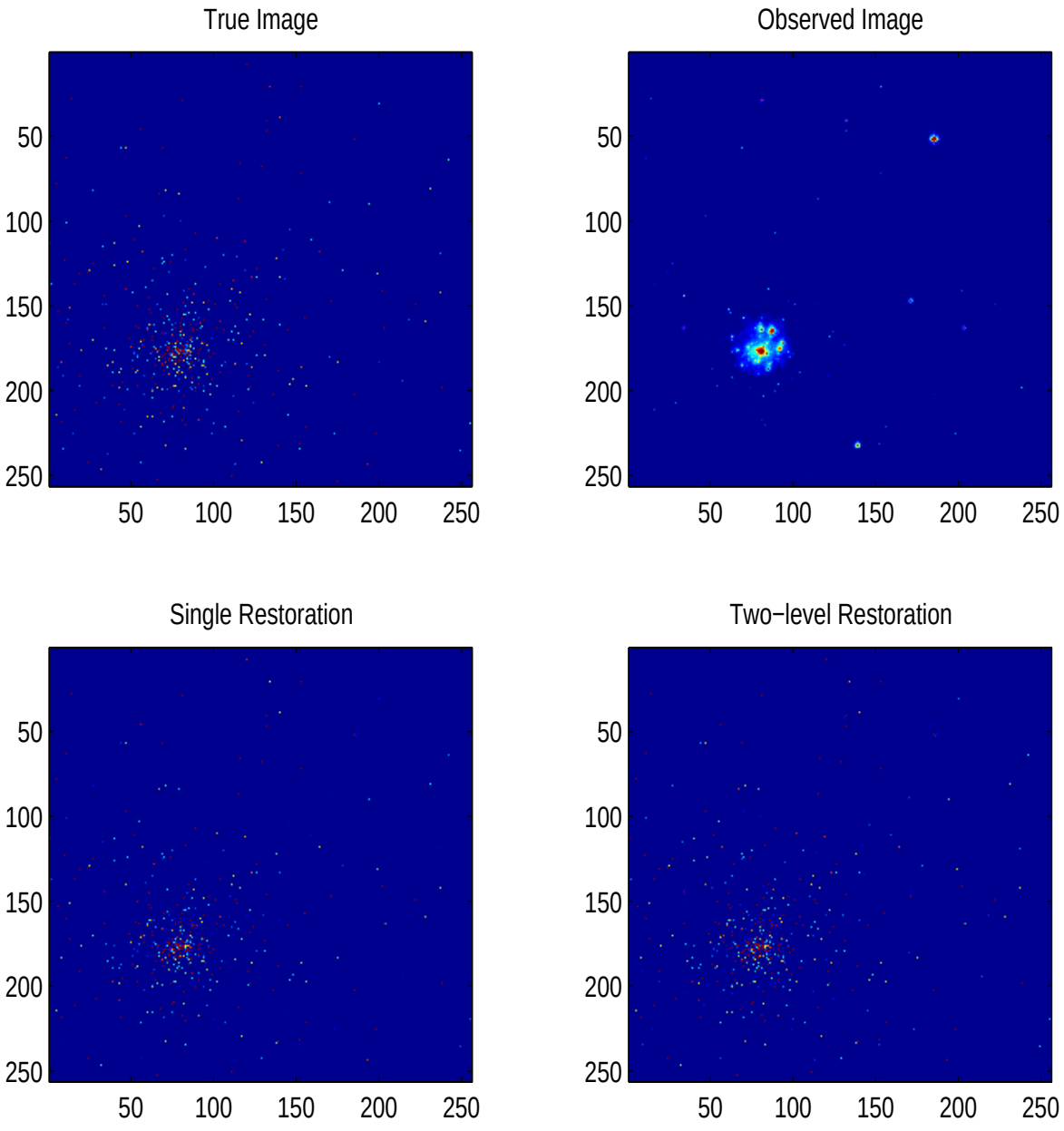

Figure 6.2: Use of two-level method

	x^*	Single	Two-level
$\|b - Ax\|_2$	539.58	552.40	504.95
$\max x_j$	31651.06	31229.29	31247.55
$\min x_j$	0.0	0.12	-84.25

Table 6.9: Statistics of single and two-level restored images

in Table 6.9. Only three pixels are negative in the two-level restoration, but they hardly have an impact on the quality of image. By re-doing with a more conservative x_1, we could obtain a nonnegative x_1, but this is not necessary in our experiment.

We also show the effectiveness of the two-level method in terms of optimality tolerance. Table 6.10 shows the results of a single run for each optimality tolerance and a two-level run with 10^{-5}. For the single run with 10^{-7}, the number of LSQR iterations and running time are almost double those with 10^{-5}. On the other hand, the second level can be solved loosely and the total number of LSQR iterations is quite small. The increase in running time is due to the QR factorization of A_1, whose dimension is 65536×105.

	x^*	Single (OptTol)			Two-level (OptTol)
		10^{-5}	10^{-6}	10^{-7}	SSD, then 10^{-5}
[1e+4 , 1e+5)	5	4	4	4	5
[1e+3 , 1e+4)	84	85	86	86	85
[100 , 1e+3)	**381**	**315**	**349**	**353**	**354**
[10 , 100)	0	454	938	1453	1774
[1 , 10)	0	36002	3530	1983	2116
[0.1 , 1)	0	28676	59779	4621	2099
[1e−2 , 0.1)	0	0	850	56416	41542
[1e−3 , 1e−2)	0	0	0	620	17539
[1e−4 , 1e−3)	0	0	0	0	19
[0 , 1e−4)	65066	0	0	0	0
LSQR itns		**215**	**349**	**431**	**151**
$\|b - Ax\|$		552	506	499	504
Time (sec)		**806**	**1244**	**1508**	**1282**

Table 6.10: Distribution of single and two-level restored images. A_1 is 65536×105.

The two-level method is especially suited to basis pursuit de-noising (BPDN) problems, where the solution is very sparse and A has many more columns than rows [8]. The performance is shown in Table 6.11 on data from Wavelab, developed by Donoho *et al.* [49]. In most cases, the two-level approach finds the sparse solution within fewer iterations.

Problem	n	n_1	Single (itns)	Two-level (itns)
example 1	22528	11	82	50
example 2	2048	1	26	28
example 3	2048	13	512	280
example 4	4096	36	738	277
example 5	2048	1	373	301
example 6	4096	9	254	137

Table 6.11: LSQR itns with single and two-level run on BPDN examples
n = dimension of x, n_1 = dimension of x_1

Figures 6.3–6.5 show the single and two-level restored images for examples 1, 3, and 4. For example 1, the two-level restored image has a fat tail, but single restoration does not. As shown in Table 6.12, two-level restoration has smaller residual norm and the range of restored values of two-level restoration is closer to that of the original signal. Similarly for examples 3 and 4.

	x^*	Single	Two-level
$\|x^* - x\|_2$	0	12.5774	6.9939
$\max x_j$	12.1977	9.3923	13.7092
$\min x_j$	-8.6250	-6.0482	-7.7836

	x^*	Single	Two-level
$\|x^* - x\|_2$	0	7.8823	5.7358
$\max x_j$	19.0088	18.9563	19.0055
$\min x_j$	-7.3111	-7.1306	-7.4500

	x^*	Single	Two-level
$\|x^* - x\|_2$	0	5.6289	5.4015
$\max x_j$	13.0502	13.6215	13.6610
$\min x_j$	0.17799	0.8963	0.8276

Table 6.12: Statistics of single and two-level restored images (BPDN examples 1, 3, 4)

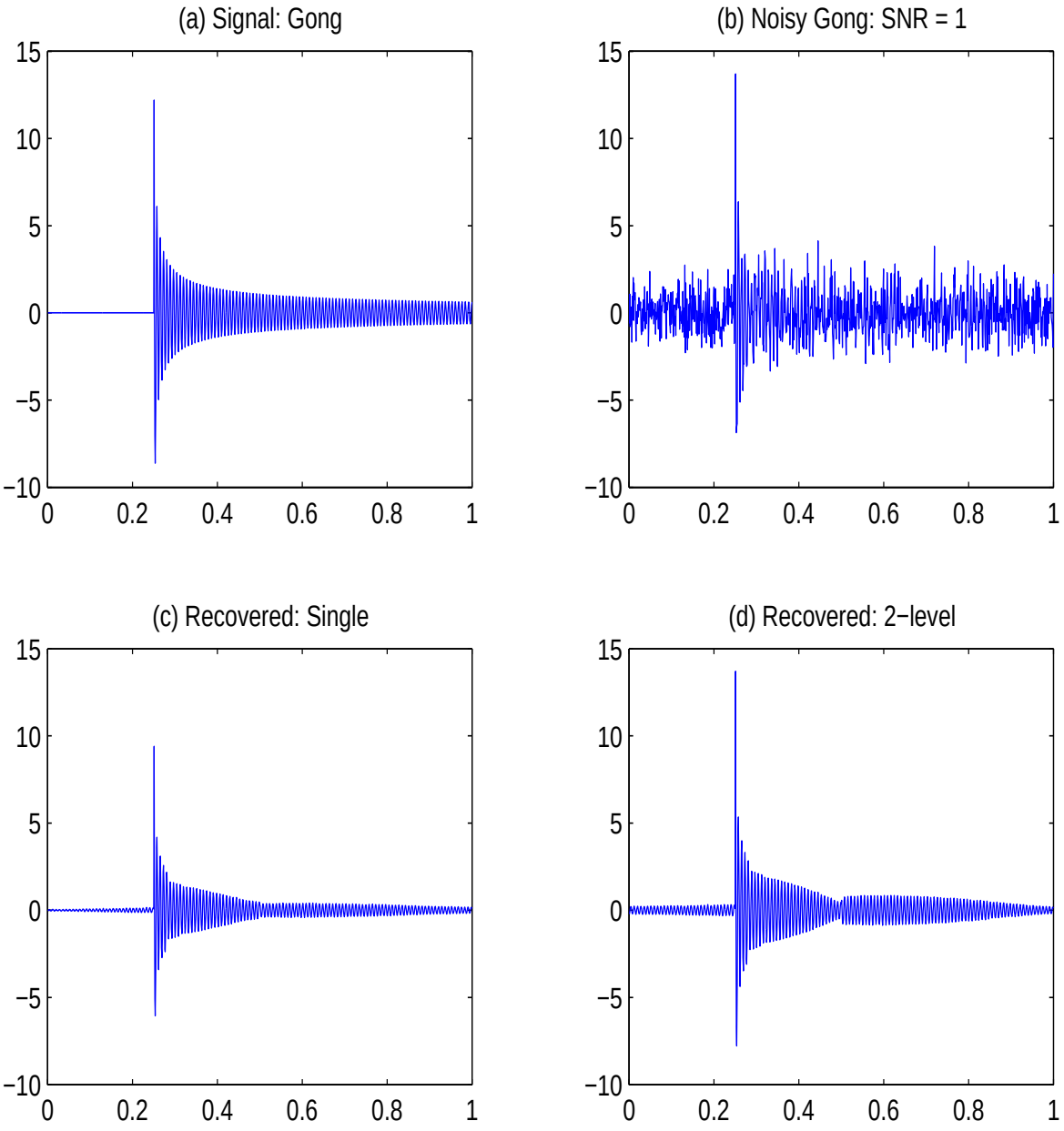

Figure 6.3: Single solver and two-level solver (example 1)

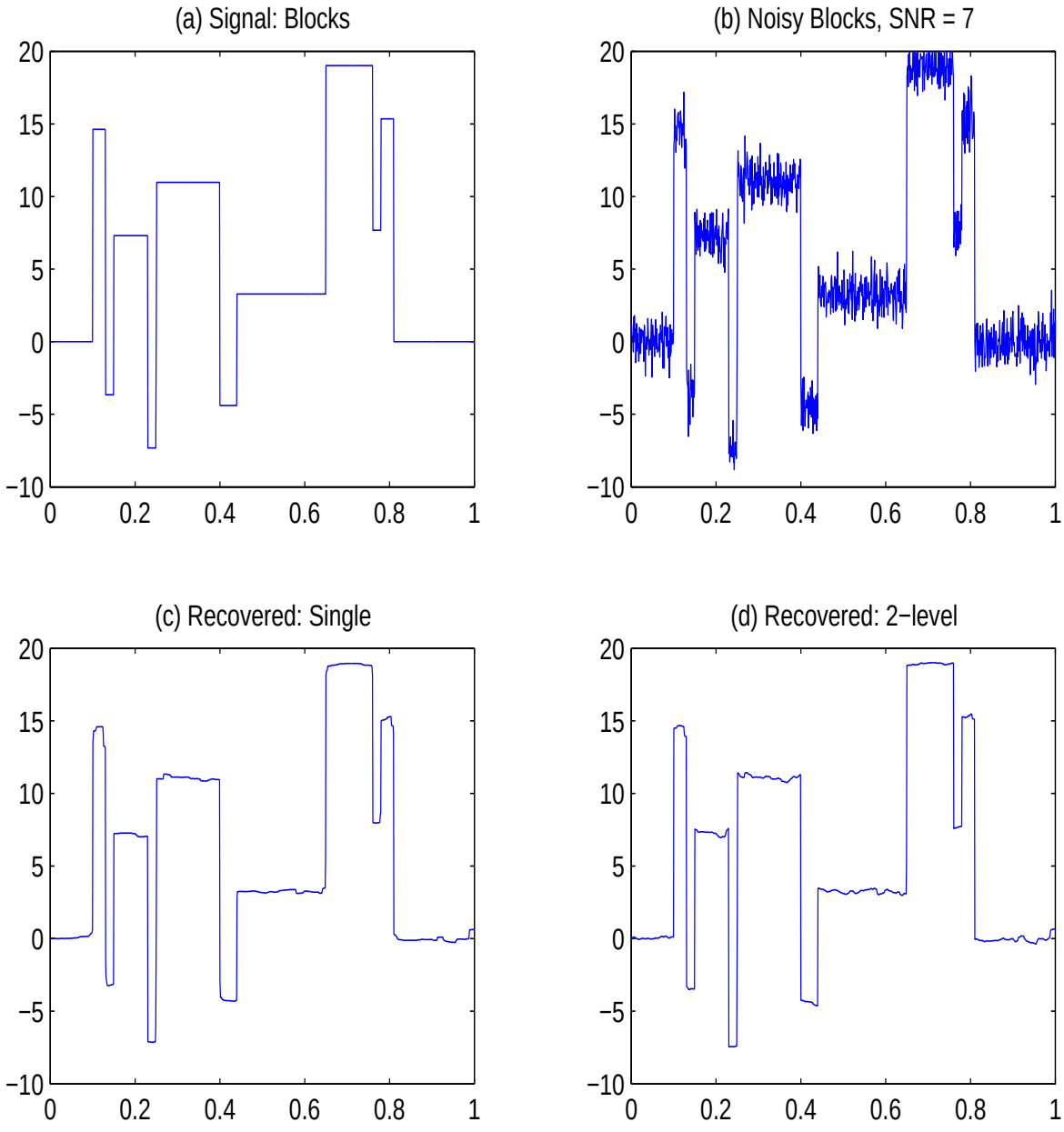

Figure 6.4: Single solver and two-level solver (example 3)

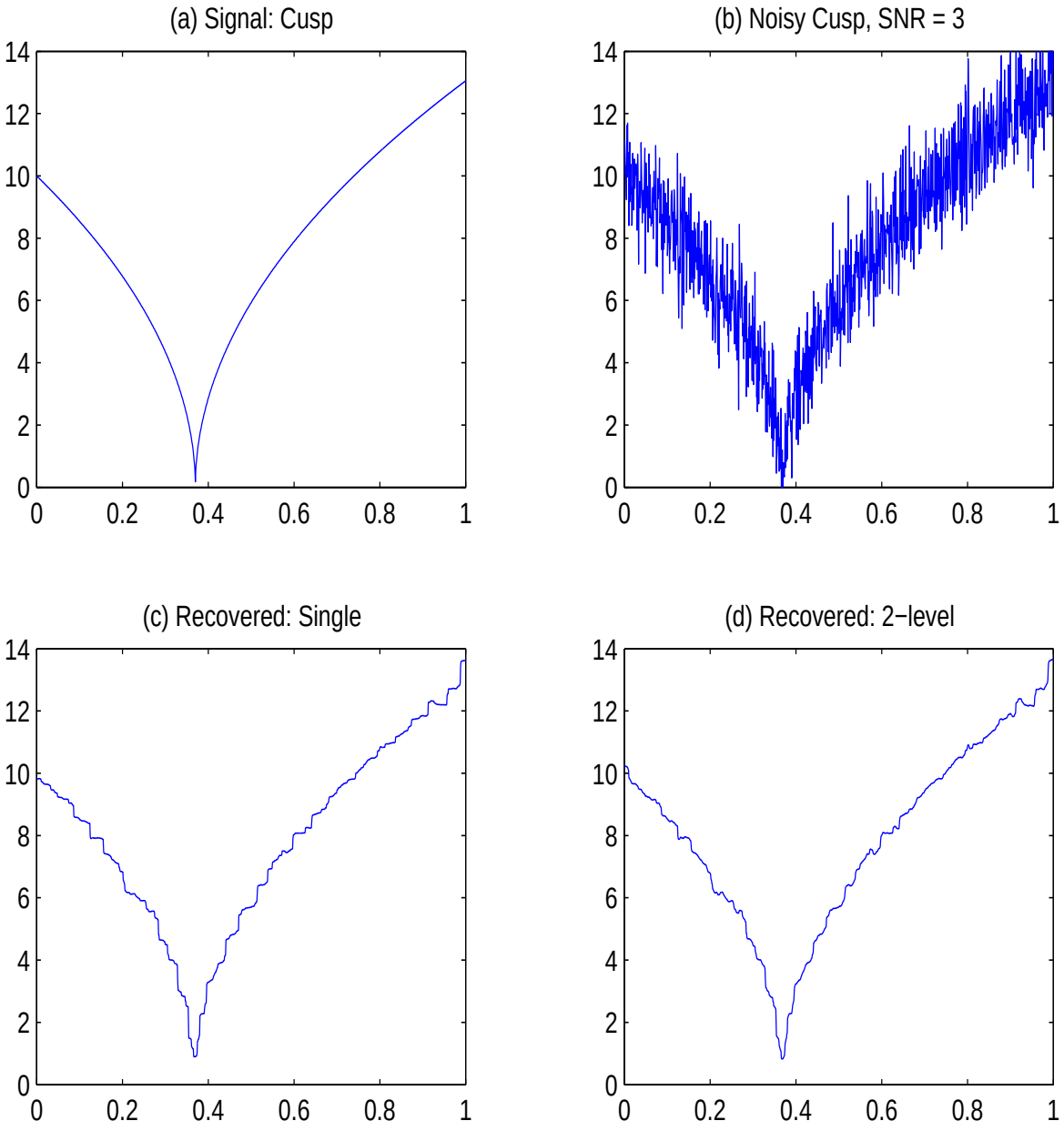

Figure 6.5: Single solver and two-level solver (example 4)

Chapter 7

NNLS using MINOS

In this chapter, we review the reduced-gradient method that has been implemented in MINOS and explain how to exploit MINOS to solve the NNLS problem. The usage of MINOS for the ML problem is discussed in Chapter 8.

7.1 MINOS

MINOS is a software package for solving large-scale optimization problems. It is especially effective for linear programs and for problems with a nonlinear objective function and sparse linear constraints, such as quadratic programs (Murtagh and Saunders [29]). MINOS can also process large numbers of nonlinear constraints (Murtagh and Saunders [35]). The nonlinear functions should be smooth but need not be convex. For linear programs, MINOS uses a sparse implementation of the primal simplex method. For nonlinear objective functions (and linear constraints), MINOS uses a reduced-gradient method with quasi-Newton approximations to the reduced Hessian. To solve the NNLS problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & F(x) = \frac{1}{2}\|b - Ax\|^2 + \frac{1}{2}\|\gamma x\|^2 \\ \text{subject to} & x \geq 0, \end{array}$$

we need a subroutine `funobj` to compute the objective function $F(x)$ and its gradient $\nabla F(x)$:

$$\begin{aligned} r &= b - Ax \\ F(x) &= \tfrac{1}{2}\|r\|^2 + \tfrac{1}{2}\|\gamma x\|^2 \\ \nabla F(x) &= -A^T r + \gamma^2 x. \end{aligned}$$

The amount of work done by `funobj` is two matrix-vector products, Ax and $A^T r$, which is the same amount of work per iteration as for LSQR in Chapter 6. Since MINOS must have at least one constraint and our formulation does not have any, we must create a dummy constraint: $-\infty < e^T x < \infty$. Adding a slack variable, we have $e^T x + s = 0$, which becomes

$$Bx_B + Sx_S + Nx_N = 0,$$

where $B = 1$. The slack variable for this constraint becomes the single "basic" variable throughout. At each stage, x_S, a set of "superbasic" variables, is free to move (along with the basic variables) and the remaining "nonbasic" variables are held fixed (usually on their bound).

Just as we do with PDSCO, we can use the solution from other methods as a starting point for MINOS. Here we use the solution from SSD. Let $S = \{j;\ x_{\text{ssd}}(j) \geq x_c\}$ and $N = \{j;\ x_{\text{ssd}}(j) < x_c\}$ be index sets for the superbasic and nonbasic variables, where x_c is a cutoff value. We set $x_c = 0.1 \cdot \max(x_{ssd})$ for our test. We then set

$$\begin{aligned} x^{(0)}(S) &= x_{\text{ssd}}(S), \\ x^{(0)}(N) &= 0. \end{aligned}$$

7.2 Computational Results

We use MINOS through TOMLAB, which is a general-purpose MATLAB environment for optimization research [46]. Table 7.1 shows the general pattern of a MINOS run on the star image, where **nsb** is the number of superbasic variables, **nobj** is the number of times subroutine **funobj** has been called to evaluate the nonlinear objective function and gradient, and **rg** is the 2-norm of reduced-gradient. For images in which only a small percentage of the pixels are significant, the reduced-gradient method should work well (for moderate-sized images). However, the reduced-gradient method updates only a few variables at a time, and it works with a dense triangular matrix as large as the number of superbasic variables. As a result, it might not be a good method when the total number of nonzero pixels is more than one or two thousand.

Table 7.2 shows the distribution of MINOS and PDSCO restorations. By comparing the number of pixels in the second and third bin, we notice that MINOS is able to recover more important pixels than PDSCO with the two-level method, and the residual norm is a little better. Figure 7.1 compares the two restorations. More computational results on other data sets are presented in Chapter 9 with other methods for NNLS and ML.

Itn	rg	ninf	sinf	Objective	LU	nobj	nsb	cond(H)
1	2.0E-03	0	0.000E+00	2.88818088E-02	1	8	429	1.0E+00
2	1.2E-03	0	0.000E+00	2.42483751E-02	1	11	429	1.0E+00
3	7.8E-04	0	0.000E+00	2.33768748E-02	1	14	429	1.1E+00
4	-1.1E-03	0	0.000E+00	2.32912899E-02	1	16	434	1.1E+00
5	5.2E-04	0	0.000E+00	2.30449551E-02	1	19	434	2.3E+00
6	-9.4E-04	0	0.000E+00	2.28824677E-02	1	22	439	3.4E+00
7	5.1E-04	0	0.000E+00	2.28179334E-02	1	24	439	3.8E+00
8	2.7E-04	0	0.000E+00	2.27412858E-02	1	27	439	8.8E+00
9	-8.1E-04	0	0.000E+00	2.26328623E-02	1	30	444	8.9E+00
10	3.5E-04	0	0.000E+00	2.26073087E-02	1	32	444	9.3E+00
				...				
292	-1.0E-04	0	0.000E+00	2.01542866E-02	1	754	1188	2.8E+01
293	3.1E-05	0	0.000E+00	2.01513552E-02	1	757	1188	2.8E+01
294	-1.0E-04	0	0.000E+00	2.01506263E-02	1	760	1193	2.8E+01
295	4.9E-05	0	0.000E+00	2.01484693E-02	1	762	1193	2.8E+01
296	-1.0E-04	0	0.000E+00	2.01456274E-02	1	765	1198	2.8E+01
297	3.4E-05	0	0.000E+00	2.01453559E-02	1	767	1198	2.8E+01
298	-1.0E-04	0	0.000E+00	2.01433250E-02	1	770	1201	2.8E+01
299	4.6E-05	0	0.000E+00	2.01432504E-02	1	772	1201	2.8E+01
300	-1.0E-04	0	0.000E+00	2.01403794E-02	1	775	1206	2.8E+01

Table 7.1: General pattern of MINOS run on star image

	x^*	PDSCO	MINOS
[1e+04 , 1e+05)	5	4	4
[1e+03 , 1e+04)	84	**85**	**87**
[100 , 1e+03)	381	**315**	**324**
[10 , 100)	0	454	15
[1 , 10)	0	36002	0
[0.1 , 1)	0	28676	0
[0.01 , 0.1)	0	0	0
[0.001 , 0.01)	0	0	0
[0.0001 , 0.001)	0	0	0
[0 , 1e−04)	65066	0	65106
Iterations		215	500
Residual	539.58	552.40	537.36

Table 7.2: Distribution of PDSCO and MINOS restorations of star image

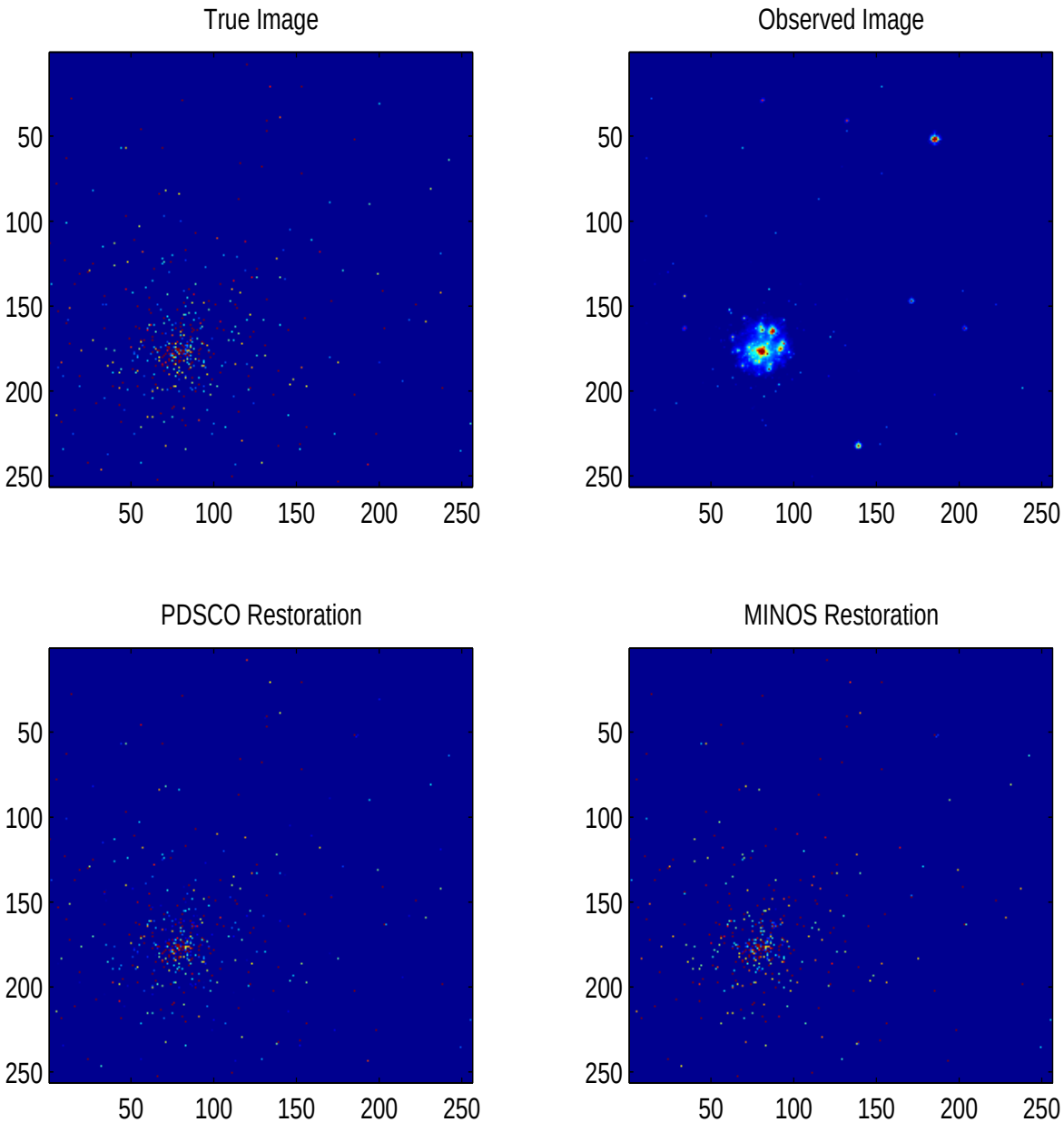

Figure 7.1: PDSCO and MINOS restoration

Chapter 8

ML, ℓ_1 and Chebyshev Approximation

This chapter covers the maximum likelihood (ML) formulation discussed in Section 4.1.2. The ML model is used extensively in medical imaging, where assuming the Poisson property of the noise is appropriate. Hence it is meaningful to have a fast and efficient method for the ML model. We also investigate the possibility of ℓ_1 and Chebyshev approximation as an alternative to the least squares model.

8.1 ML using SSD

Recalling the ML model

$$\boxed{\begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{J}(x) = \sum (Ax)_i - \sum b_i \log (Ax)_i \\ \text{subject to} & x \geq 0, \end{array}} \tag{8.1}$$

we derive the gradient of $\mathcal{J}(x)$ with respect to z using the elementwise parameterization $x = e^z$ and the chain rule:

$$\nabla_z \mathcal{J}(x) = X \nabla_x \mathcal{J}(x) = X A^T Y^{-1} (Ax - b),$$

where $y = Ax$, $Y = \text{diag}(y)$ and $X = \text{diag}(x)$. By assuming $A^T e = e$, we have

$$\nabla_z \mathcal{J}(x) = X A^T (e - Y^{-1} b) = x - X A^T Y^{-1} b.$$

Finally, we obtain the fixed point iteration by setting $\nabla_z \mathcal{J}(x) = 0$:

$$x_{k+1} = X_k A^T {Y_k}^{-1} b, \quad X_k = \mathrm{diag}(x_k), \quad Y_k = \mathrm{diag}(y_k).$$

In spite of the ease of implementation from the fixed point relationship, this SSD method converges very slowly. In addition, SSD has the following disadvantages:

- Some elements of y can be very small, causing failure of SSD in some instances.
- The assumption $A^T e = e$ holds only for pixels that are away from the boundary.

In fact, there is a tradeoff between the improvement in resolution of the image and the extra cost of obtaining a better image. In medical imaging, slightly better resolution does not help improve the accuracy of diagnosis. This might justify the SSD method if the first few iterations are good enough. Indeed, the first few iterations SSD produces are often not bad. This is one of the reasons that SSD has been widely used in medical imaging. However, it is not true for all applications, where slight improvement is worth extra cost. Because of these drawbacks of SSD, we resort to PDSCO and MINOS as alternatives.

8.2 ML using PDSCO

Our formulation with regularization is

$$\boxed{\begin{array}{ll} \underset{x,r,p}{\text{minimize}} & \mathcal{J}(x,r,p) = e^T r - \sum b_i \log r_i + \frac{1}{2}\|\gamma x\|^2 + \frac{1}{2}\|\gamma r\|^2 + \frac{1}{2}\|p\|^2 \\ \text{subject to} & Ax - r + \delta p = 0, \\ & x,\, r \geq 0. \end{array}} \tag{8.2}$$

The gradient and the Hessian of $\mathcal{J}(x,r,p)$ are

$$\begin{aligned} \nabla \mathcal{J}_{x,r}(x,r,p) &= \begin{pmatrix} \gamma^2 x \\ e - BR^{-1}e + \gamma^2 r \end{pmatrix} \\ \nabla^2 \mathcal{J}_{x,r}(x,r,p) &= \begin{pmatrix} \gamma^2 I & 0 \\ 0 & BR^{-2} + \gamma^2 I \end{pmatrix}, \end{aligned} \tag{8.3}$$

where $B = \text{diag}(b)$, and $R = \text{diag}(r) = \text{diag}(Ax)$. The nonlinear equations defining the central trajectory in the interior method are $p = \delta\, y$ and

$$\begin{aligned} Ax - r + \delta^2 y &= 0 \\ A^T y + z &= \gamma^2 x \\ w - y &= e - BR^{-1}e + \gamma^2 r \\ XZe &= \mu e \\ RWe &= \mu e, \end{aligned} \tag{8.4}$$

where z and w are dual variable corresponding to x and r and μ is the barrier parameter. We solve this system of equations by Newton's method as discussed in Chapter 6:

$$\begin{bmatrix} A & -I & \delta^2 I & 0 & 0 \\ -\gamma^2 I & 0 & A^T & I & 0 \\ 0 & -BR^{-2} - \gamma^2 I & -I & 0 & I \\ Z & 0 & 0 & X & 0 \\ 0 & W & 0 & 0 & R \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \\ \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} \overline{r} \\ t_1 \\ t_2 \\ v_1 \\ v_2 \end{bmatrix}, \tag{8.5}$$

where $\overline{r} = r - Ax - \delta^2 y$, $t_1 = \gamma^2 x - A^T y - z$, $t_2 = e - BR^{-1}e + \gamma^2 r + y - w$, $v_1 = \mu e - Xz$, and $v_2 = \mu e - Rw$. This reduces to

$$\begin{bmatrix} -H_1 & 0 & A^T \\ 0 & -H_2 & -I \\ A & -I & \delta^2 I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \overline{r} \end{bmatrix}, \tag{8.6}$$

where $H_1 = X^{-1}Z + \gamma^2 I$, $H_2 = BR^{-2} + \gamma^2 I + R^{-1}W$, $w_1 = t_1 - X^{-1}v_1$, $w_2 = t_2 - R^{-1}v_2$, $\Delta z = X^{-1}(v_1 - Z\Delta x)$, $\Delta w = R^{-1}(v_2 - W\Delta r)$. Eliminating Δx and Δr gives

$$\min \left\| \begin{pmatrix} D_1 A^T \\ D_2 \\ \delta I \end{pmatrix} \Delta y - \begin{pmatrix} D_1 w_1 \\ -D_2 w_2 \\ \overline{r}/\delta \end{pmatrix} \right\|, \tag{8.7}$$

where $D_1^2 = H_1^{-1}$, $D_2^2 = H_2^{-1}$.

For some i, b_i can be negative because the observed image is corrupted by random noise, which could have negative values depending on the statistical property (distribution) of the noise. If the noise has big negative values as in normal distributions, then $b = Ax + \eta$ can be negative as well. Thus, BR^{-2} can contain negative elements, which implies $\nabla^2 \mathcal{J}_{x,r}(x, r, p)$ in (8.3) is not positive definite. All the methods based on the positivity of b might be

seriously flawed. However, a remedy is possible in our case if we assume $Ae = e$. From the model we have $Ax + \eta = b$. We add αe ($\alpha > 0$) to both sides so that right-hand side is nonnegative. Then, $A(x + \alpha e) + \eta = b + \alpha e$ because of $Ae = e$. From the change of variable,

$$A\overline{x} + \eta = \overline{b}, \quad \overline{x} \geq \alpha e. \tag{8.8}$$

The SSD method cannot be applied to this problem with lower bounds because it is based on the parameterization $x = e^z$, which makes x nonnegative. However, the interior method can be easily modified to handle general bounds [42, 43].

8.3 ML using MINOS

We can also apply MINOS to the ML formulation, similarly to its use in Chapter 7:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{J}(x) = \sum (Ax)_i - \sum b_i \log (Ax)_i + \frac{1}{2}\|\gamma x\|^2 \\ \text{subject to} & x \geq 0. \end{array} \tag{8.9}$$

8.4 Chebyshev and ℓ_1 Approximation

We can convert the ℓ_1 approximation formulation

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \|b - Ax\|_1 \\ \text{subject to} & x \geq 0 \end{array} \tag{8.10}$$

into an LP as follows:

$$\begin{array}{ll} \underset{x,\,u,\,v}{\text{minimize}} & e^T(u + v) \\ \text{subject to} & Ax - u + v = b, \\ & x, u, v \geq 0, \end{array} \tag{8.11}$$

where $x \in \mathbf{R}^n, u, v \in \mathbf{R}^m$. We can also transform the ∞-norm formulation

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \|b - Ax\|_\infty \\ \text{subject to} & x \geq 0 \end{array} \tag{8.12}$$

the following LP:

$$\begin{array}{ll} \underset{x,u,v,z}{\text{minimize}} & z \\ \text{subject to} & Ax + u - ze = b, \\ & Ax - v + ze = b, \\ & x, u, v, z \geq 0, \end{array} \tag{8.13}$$

where z is a scalar and $x, u, v \in \mathbf{R}^n$. In the solution from the ℓ_1 approximation, Ax has a few large deviations from b, which damages the restored image severely because the quality of images depends on the range of pixel values. On the other hand, Chebyshev approximation (8.12) finds the solution where Ax deviates from b by a small amount componentwise. Therefore it makes more sense in image restoration. The distributions of each restoration are shown in Table 8.1. As we see, ℓ_1 restoration has a wide range of values.

			True	Observed	ℓ_1 Approx.	Chebyshev
[1e+04	,	1e+05)	264	0	230	252
[1e+03	,	1e+04)	265	1162	243	322
[100	,	1e+03)	69	1963	247	2086
[10	,	100)	5	971	235	1436
[1	,	10)	2	0	1056	0
[0.1	,	1)	0	0	2085	0
[0.01	,	0.1)	0	0	0	0
[0.001	,	0.01)	0	0	0	0
[0	,	1e−03)	3491	0	0	0

Table 8.1: Distribution of ℓ_1 and Chebyshev restoration

We present the different residual norms in Table 8.2, where x^*, x^1, and x^∞ are the true solution, ℓ_1 restoration, and Chebyshev restoration. However, in both restorations, PDSCO required many LSQR iterations compared with its performance on the NNLS or ML formulations.

Figure 8.1 shows the restored images from ℓ_1 and Chebyshev approximation. As expected, Chebyshev restoration looks better than ℓ_1 restoration.

	2-norm	1-norm	∞-norm
$b - Ax^*$	18283.02	616639.14	1436.40
$b - Ax^1$	1913.31	64075.21	311.21
$b - Ax^\infty$	4694.88	277542.75	133.15
Iterations	**432**	**6188**	**2042**

Table 8.2: Different norms of residuals

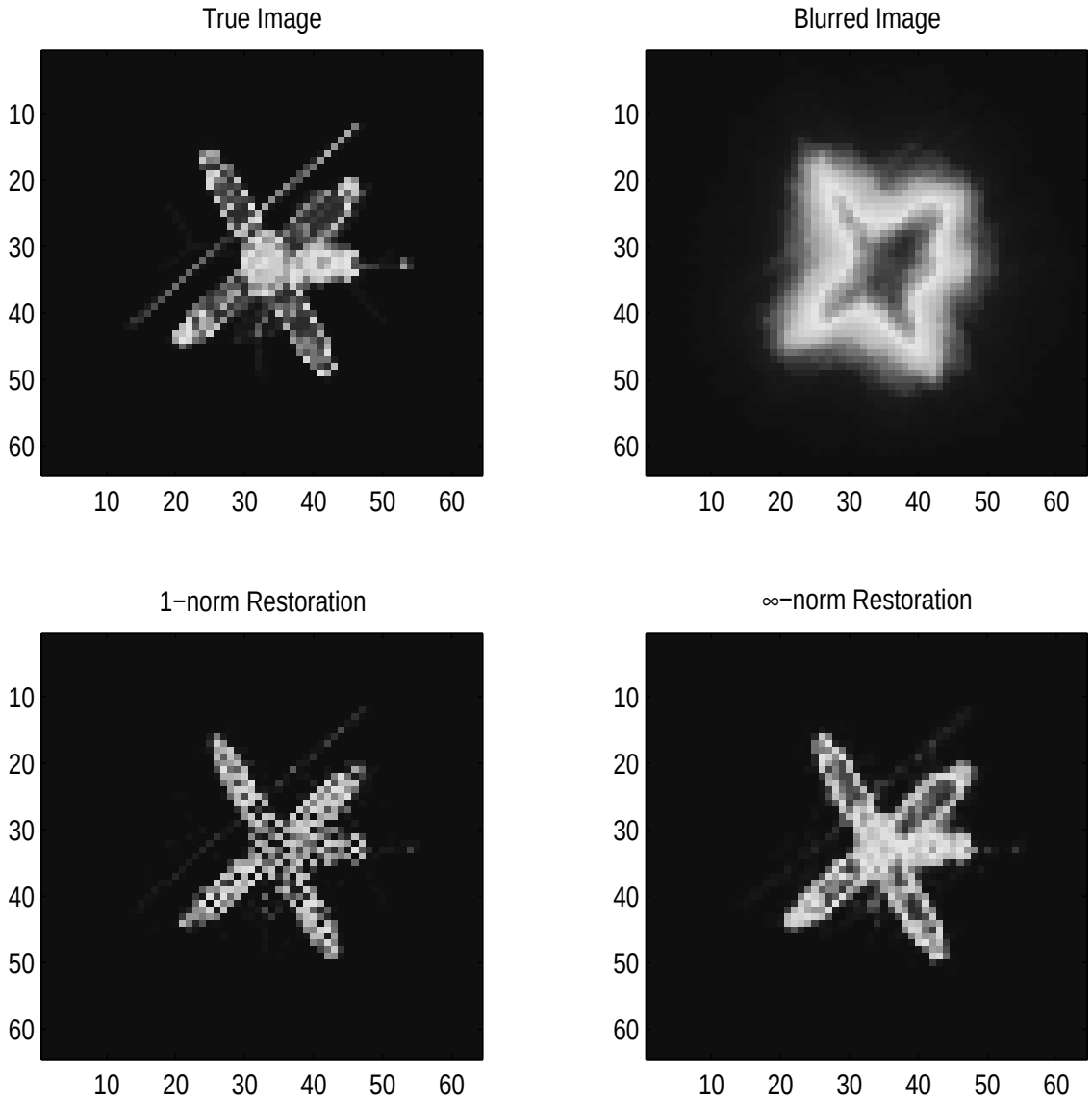

Figure 8.1: Chebyshev and ℓ_1 restoration

Chapter 9

Computational Results

In this chapter, we present some numerical results and compare each restoration method introduced in this dissertation. We give results from SSD, PDSCO, and MINOS applied to NNLS (4.3), and results from PDSCO applied to ML (4.5). The other methods SSD, MINOS for ML, Chebyshev and ℓ_1 approximation are excluded because their performance is poor.

9.1 Comparison of the Smaller Satellite Image

Figure 9.1 and Table 9.1 show the restored images of the smaller version of satellite example from each method. It is hard to see the difference clearly by looking at the pictures in black and white. The images are in color on the computer screen, which allows us to differentiate them easily. By appearance, the PDSCO, MINOS, and ML restorations look good. Three methods (SSD, PDSCO, and MINOS) solve the NNLS problem and PDSCO solves the ML model using an interior method.

As pointed out in Chapter 5, SSD works reasonably well at the beginning, but does not make much progress towards the solution after that. The restoration after two thousand iterations is not as good as the PDSCO restoration.

For this example, PDSCO with the two-level method and MINOS show very good performance in terms of the number of iterations and the distribution of the solution. PDSCO on the ML model takes more than twice as many LSQR iterations for the same optimality tolerance, but the restored image itself looks good.

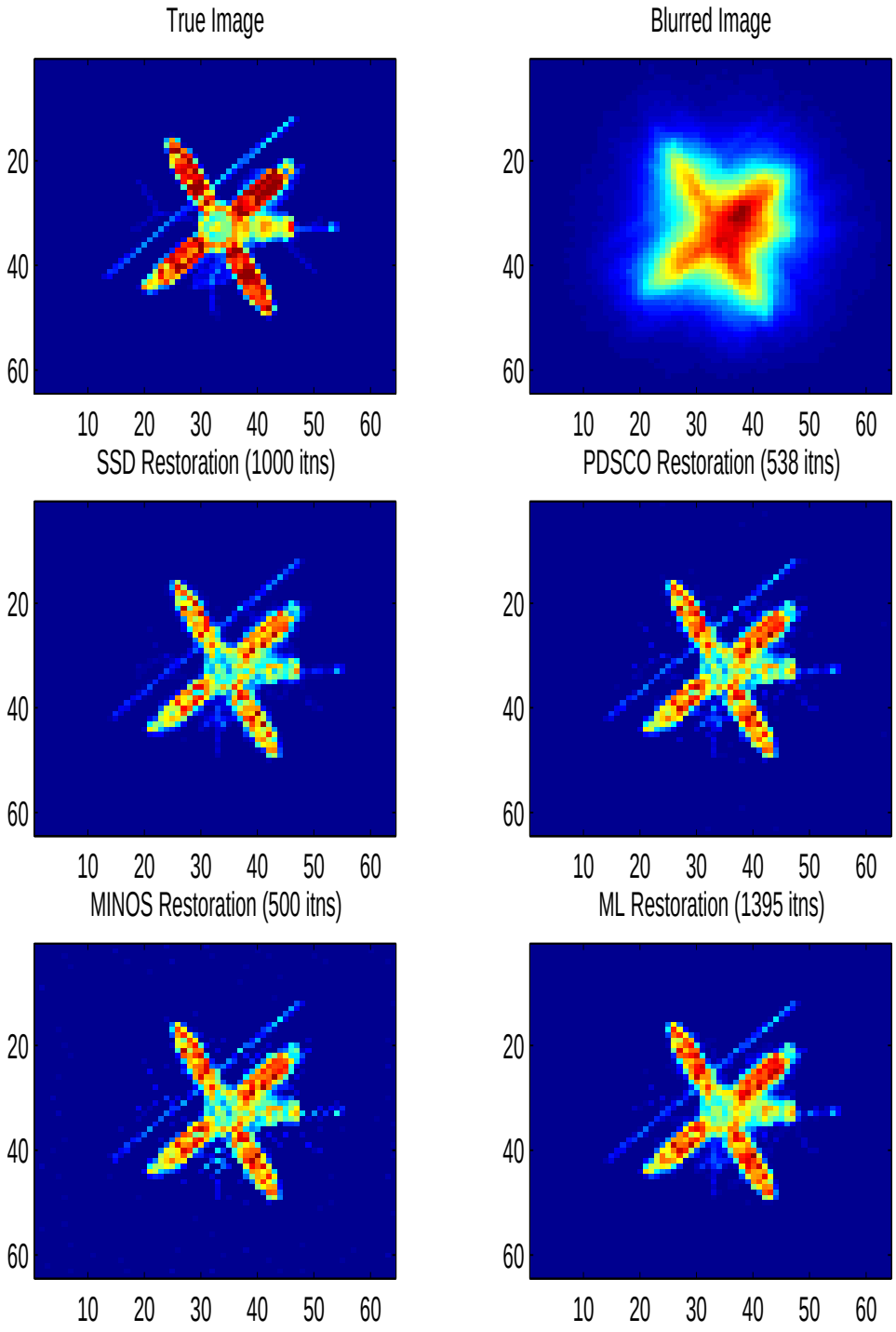

Figure 9.1: Restored images of the smaller version of satellite image

	True	SSD	PDSCO	MINOS	ML
[1e+04 , 1e+05)	264	247	257	259	261
[1e+03 , 1e+04)	265	274	258	263	274
[100 , 1e+03)	69	237	183	99	297
[10 , 100)	5	1314	3308	1	2466
[1 , 10)	2	1067	90	0	798
[0.1 , 1)	0	649	0	0	
[0.01 , 0.1)	0	257	0	0	
[0.001 , 0.01)	0	42	0	0	
[0.0001 , 0.001)	0	7	0	0	
[1e−05 , 0.0001)	0	2	0	0	
[0 , 1e−05)	3491	0	0	3474	0
Iterations		2000	538	500	1395
Residual	18283	1700	1781	1722	2138

Table 9.1: Distribution of true and restored images of smaller version of satellite example

9.2 Comparison of Satellite Image

To see the difference in performance of each method, we do the same tests on the original satellite image, which is four times bigger than the smaller version. Figure 9.2 and Table 9.2 show the restored images of the satellite image for each restoration method. In this example, PDSCO and ML restoration are good by appearance. The PDSCO restoration looks better than SSD and MINOS for NNLS. MINOS has difficulties in handling many nonzeros. In the recovered image, some bright pixels are located close to the boundary, and even in the center of the recovered image, many pixels are separated. PDSCO with the two-level method shows very good performance in terms of the number of iterations and the distribution of the solution. ML restoration looks a little blurry and the exact shape is not recovered even though the color in the body looks acceptable.

9.3 Comparison of Star Image

The above two examples have many nonzeros in the images, but the star image is very sparse. Figure 9.3 and Table 9.3 show the restored star images for each restoration method. In this example, the PDSCO and MINOS restorations look good. Their NNLS restorations look better than SSD's. For the ML model, the image quality is reasonable, but PDSCO requires many LSQR iterations.

	True	SSD	PDSCO	MINOS	ML
[1e+03 , 1e+04)	5572	5157	5403	5523	5589
[100 , 1e+03)	1024	6005	2869	0	5473
[10 , 100)	82	18366	3653	0	22796
[1 , 10)	2	25399	12333	1	31678
[0.1 , 1)	0	10478	41278	0	0
[0.01 , 0.1)	0	131	0	0	0
[0.001 , 0.01)	0	0	0	0	0
[0.0001 , 0.001)	0	0	0	0	0
[1e−05 , 0.0001)	0	0	0	0	0
[0 , 1e−05)	58858	0	0	60012	0
Iterations		500	431	1000	584
Residual	10122.4	10123.3	9983.25	10122.3	10076.7

Table 9.2: Distribution of true and restored images of satellite example

	True	SSD	PDSCO	MINOS	ML
[1e+04 , 1e+05)	5	4	4	4	4
[1e+03 , 1e+04)	84	86	85	87	81
[100 , 1e+03)	381	348	315	324	371
[10 , 100)	0	854	454	15	1004
[1 , 10)	0	5559	36002	1	17936
[0.1 , 1)	0	17201	28676	0	46140
[0.01 , 0.1)	0	22600	0	0	0
[0.001 , 0.01)	0	13502	0	0	0
[0.0001 , 0.001)	0	4131	0	0	0
[1e−05 , 0.0001)	0	843	0	0	0
[0 , 1e−05)	65066	408	0	65106	0
Iterations		1000	215	500	7130
Residual	539.58	501.02	552.40	537.36	931.53

Table 9.3: Distribution of true and restored images of star problem

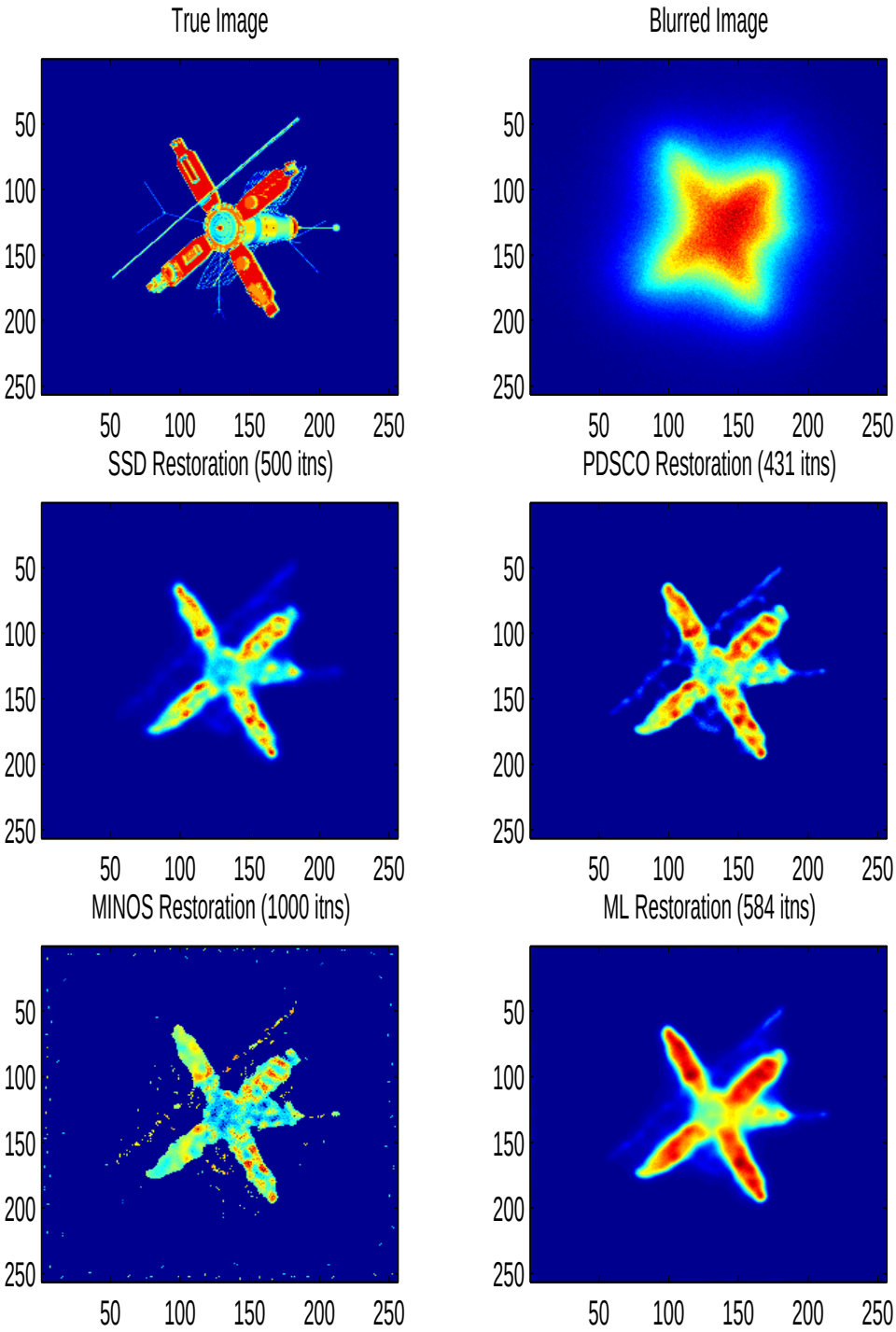

Figure 9.2: Restored images of the satellite image

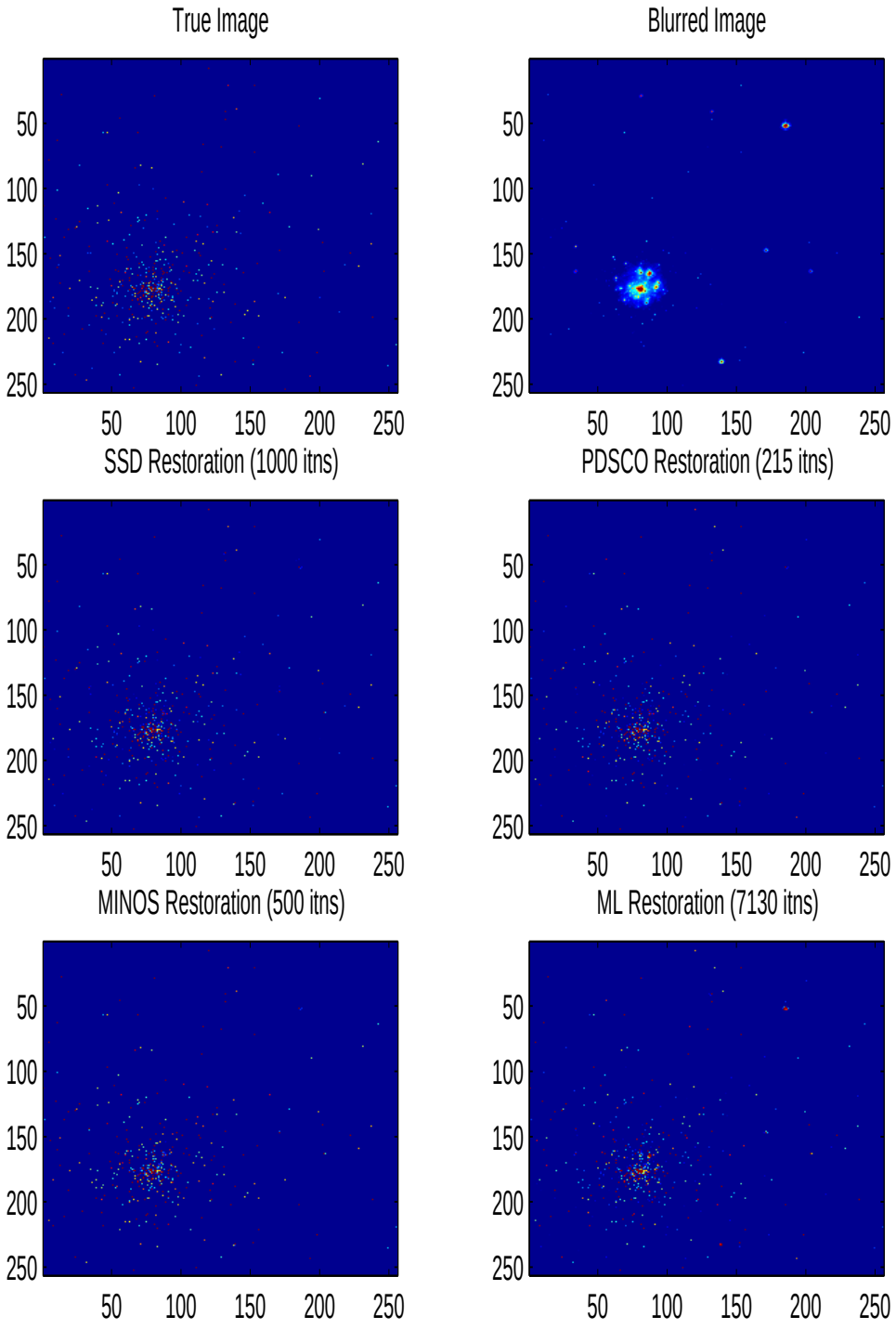

Figure 9.3: Restored images of the star image

Chapter 10

Conclusions and future research

In this chapter, we present an overview of the dissertation and discuss its contributions. We also discuss new research directions.

10.1 Summary

In this dissertation, we have studied numerical optimization methods for image restoration with nonnegativity constraints. There are several formulations for image restoration and the appropriate model is chosen to match the application. Two major models are nonnegative least squares (NNLS) and maximum likelihood (ML). Our formulations are slightly different from the original least squares problem and ML model with nonnegativity constraints. By change of variables, our models have separable objective functions with linear equalities and nonnegativity constraints.

We have developed a primal-dual interior approach for image restoration. The method can be applied to NNLS and ML with different formulations. Many useful computational techniques are included in our implementation, PDSCO. One of the main features is an inexact Newton's method based on the iterative solver LSQR. The advantage of the inexact method is that the computational work can be reduced by controlling the accuracy of the iterative solver. In addition, regularization, scaling, preconditioning, warm and hot start, and predictor-corrector features have been implemented and evaluated on sparse images and Basis Pursuit (BP) de-noising problems.

The reduced-gradient method is an alternative approach. It is especially good for sparse images and problems with inherently sparse solutions. BP de-noising problems are prime examples.

For NNLS, we have developed a two-level method for use with PDSCO. The idea is that

early primal-dual iterations identify the largest elements of x with relatively few LSQR iterations. Using QR factors of a few columns of A that correspond to the large elements of x, we can reduce the original problem to a slightly smaller problem. We then apply PDSCO to the reduced problem, needing only moderate accuracy to solve for the smaller components of x.

10.2 Future Research

Although many types of degradation can be approximated by linear and space invariant processes, nonlinear and space variant techniques are more general and accurate. The advantage of the linear space invariant approach is that extensive tools of linear systems theory readily become available for the image restoration problem.

There are several directions for future research. The first is relaxation of the space invariance of the blurring operator. Without space invariance, the matrix-vector product cannot be done using the Fourier transform because we lose the Toeplitz property of $\mathbf{H}$. Thus, computing the matrix-vector multiplications $\mathbf{H}x$ and $\mathbf{H}^T y$ for arbitrary vectors x and y will be very expensive. Because of the enormous size and density of $\mathbf{H}$ we must resort to iterative methods, for which an efficient matrix-vector multiplication routine is critical. Therefore, problem structure and other forms of sparsity must be exploited.

The second is extension to the nonlinear operator. Solving general nonlinear problems is quite challenging in theory and practice, but there are some tractable nonlinear problems such as those having convexity.

The next point is the utilization of noise information. As with the Wiener filter, we should be able to get better images with noise information. There are some cases where the noise distribution is known. New models that make use of the noise information can then be considered.

Finally, it would be interesting to find other large-scale applications where NNLS solutions are required. Solving large-scale NNLS is computationally demanding. The interior methods we have developed show excellent performance under certain conditions, and they apply to a broader range of objective functions. For example, PDSCO has already proved remarkably efficient on extremely large problems of the form (6.2) in cases where the objective includes the entropy function $\sum_j x_j \log x_j$.

Appendix A

List of Symbols

A^T	transpose of A
$\overline{A}$	conjugate of A
A^*	conjugate transpose of A
$A(i,:)$	i-th row of A
$A(:,j)$	j-th column of A
$A \otimes B$	Kronecker product of A and B
$A.*B$	Hadamard (element by element) product of A and B
$F(F_N)$	Fourier matrix (of order N)
$F(u,v)$	2-D Fourier transform of $f(x,y)$
$H(u,v)$	2-D Fourier transform of $h(x,y)$
$\mathbf{H}$	matrix representing degradation model
$\mathrm{diag}(a_1,\ldots,a_n)$	the diagonal matrix with elements $a_1,\ldots,a_n$
$\mathrm{diag}(\Lambda)$	vector of diagonal elements of diagonal matrix Λ
Ω	$\mathrm{diag}(1,w,w^2,\ldots,w^{n-1}),\ w=\exp(2\pi j/n),\ j=\sqrt{-1}$
$\|x\|_1$	1-norm of vector x
$\|x\|$	2-norm of vector x
e	vector of ones
e_j	j-th column of identity matrix
∇f	gradient of the real-valued function f
$\nabla^2 f$	Hessian of the real-valued function f

Appendix B

Properties of Kronecker Product

(1) $(\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B)$, α is a scalar

(2) $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$

(3) $C \otimes (A + B) = (C \otimes A) + (C \otimes B)$

(4) $C \otimes (A \otimes B) = (C \otimes A) \otimes B$

(5) $(A \otimes B)(C \otimes D) = (AC \otimes BD)$; $\quad (C \otimes D)(A \otimes B) = (CA \otimes DB)$

(6) $\overline{(A \otimes B)} = \overline{A} \otimes \overline{B}$

(7) $(A \otimes B)^T = A^T \otimes B^T$

(8) $(A \otimes B)^* = A^* \otimes B^*$

(9) $\text{rank}\,(A \otimes B) = \text{rank}(A)\,\text{rank}(B)$

(10) $\mathbf{Tr}(A \otimes B) = \mathbf{Tr}(A)\ \mathbf{Tr}(B)$

(11) $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$

(12) $\det\,(A \otimes B) = \det(A)^n \det(B)^m$, A is $m \times n$ matrix

(13) There exists a permutation matrix P such that $(B \otimes A) = P^*(A \otimes B)P$

(14) $\phi(A \otimes B) = \{\mu_i \nu_i\}$, where $\phi(A) = \{\mu_i\}, \phi(B) = \{\nu_i\}$, and $\phi(A)$ is the spectrum of A

Bibliography

[1] Mikael Adlers. *Topics in Sparse Least Squares Problems.* Linköpings Universitet: Doctoral Dissertation, 2000.

[2] Åke Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[3] N. L. Boland. A dual-active-set algorithm for positive semi-definite quadratic programming. *Mathematical Programming*, 78:1–27, 1997.

[4] Richard A. Carreras and Michael C. Roggemann. Image restoration using nonlinear optimization techniques for partially compensated imaging system. *Inverse Problems in Scattering and Imaging*, 1767:307–317, July 1992.

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

[6] T. F. Chan. An optimal circulant preconditioner for Toeplitz systems. *SIAM Journal on Scientific Computing*, 9(3):766–771, 1988.

[7] T. F. Chan and J. A. Olkin. Preconditioner for Toeplitz-block matrices. *Numerical Algorithms*, 6:89–101, 1993.

[8] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[9] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem.* Academic Press, Inc., 1992.

[10] Philip J. Davis. *Circulant Matrices.* John Wiley & Sons, Inc., 1979.

[11] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, April 1982.

[12] A. P. Dempster, N. M. Laid, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc., Series B*, 39:1–38, 1977.

[13] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley and Sons, New York and Toronto, 1968.

[14] R. Freund, G. Golub, and N. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992.

[15] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.

[16] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (Third Edition)*. The Johns Hopkins University Press, 1996.

[17] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison and Wesley, 1992.

[18] Martin Hanke, James G. Nagy, and Curtis Vogel. Quasi-newton approach to nonnegative image restoration. *Linear Algebra and its Applications*, 316:223–236, September 2000.

[19] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[20] Calvin A. Johnson and Ariela Sofer. A primal-dual method for large-scale image reconstruction in emission tomography. *SIAM Journal on Optimization*, 11(3):691–715, 2001.

[21] Julie Kamm and James G. Nagy. Optimal Kronecker product approximation of block Toeplitz matrices. *SIAM Journal on Matrix Analysis and Applications*, 22:155–172, 2000.

[22] Linda Kaufman. Implementing and accelerating the EM algorithm for positron emission tomography. *IEEE Transactions on Medical Imaging*, MI-6(1), 1987.

[23] Linda Kaufman. Maximum likelihood, least squares, and penalized least squares for PET. *IEEE Transactions on Medical Imaging*, 12(2), 1993.

[24] Yuying Li and Fadil Santosa. A computational algorithm for minimizing total variation in image restoration. *IEEE Transactions on Image Processing*, 5(6):987–995, June 1996.

[25] David G. Luenberger. *Optimization by Vector Space Methods.* John Wiley & Sons, Inc., 1968.

[26] David G. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, 1984.

[27] Pontus Matstoms. Sparse QR factorization in MATLAB. *ACM Transactions on Mathematical Software*, 20:136–159, 1994.

[28] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods. Springer-Verlag, New York, 1988.

[29] Bruce A. Murtagh and Michael A. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, 14:41–72, 1978.

[30] Bruce A. Murtagh and Michael A. Saunders. *MINOS 5.5 User's Guide.* Technical Report SOL 83-20R, Department of Operations Research, 1998.

[31] Katta G. Murty. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, 1988.

[32] James G. Nagy and Dianne P. O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, July 1998.

[33] James G. Nagy, Robert J. Plemmons, and Todd C. Torgersen. Iterative image restoration using approximate inverse preconditioning. *IEEE Transactions on Image Processing*, 5(7):1151–1162, July 1996.

[34] James G. Nagy and Z. Strakoš. Enforcing nonnegativity in image reconstruction algorithms. *Mathematical Modeling, Estimation, and Imaging*, 4121:182–190, 2000.

[35] Christopher C. Paige and Michael A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, March 1982.

[36] Christopher C. Paige and Michael A. Saunders. Algorithm 583 LSQR: Sparse Linear Equations and Least Squares Problems. *ACM Transactions on Mathematical Software*, 8(2):195–209, June 1982.

[37] P. Pardalos, Y. Ye, and C. Han. An interior-point algorithm for large-scale quadratic problems subject to box constraints. *Lecture Notes in Control and Information Sciences, Springer, Berlin and New York*, 144:413–422, 1990.

[38] Luis F. Portugal, Joaquim J. Júdice, and Luis N. Vicente. A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation*, 63(208):625–643, October 1994.

[39] William K. Pratt. *Digital Image Processing.* New York:Wiley, 1991.

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

[41] Michael A. Saunders and Byunggyoo Kim. MATLAB code `pdsco.m`. http://www.stanford.edu/group/SOL/software/, 2002.

[42] Michael A. Saunders and John A. Tomlin. Solving regularized linear programming using barrier methods and KKT systems. SOL Report 96-4, December 1996.

[43] Michael A. Saunders and John A. Tomlin. Stable reduction to KKT systems in barrier methods for linear and quadratic programming. SOL Report 96-3, August 1996.

[44] L. Shepp and Y. Vardi. Maximum likelihood reconstruction in positron emission tomography. *IEEE Transactions on Medical Imaging*, pages 113–122, 1982.

[45] ftp://ftp.stsci.edu/software/stsdas/testdata/restore/sims/star_cluster.

[46] http://www.tomlab.biz.

[47] Curt Vogel. *Computational Methods for Inverse Problems.* To appear in Frontiers in Applied Mathematics Series, SIAM, Philadelphia, 2002.

[48] Curtis R. Vogel and Mary E. Oman. Fast, robust total variation-based reconstruction of noisy, blurred images. *IEEE Transactions on Image Processing*, 7(6):813–824, June 1998.

[49] http://www-stat.stanford.edu/~wavelab.

[50] Stephen J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, 1997.