Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

Qing Chu	Date

Mathematical Models and Numerical Methods for Wavefront Reconstruction

By

Qing Chu
Doctor of Philosophy

Mathematics

James G. Nagy, Ph.D.
Advisor

Michele Benzi, Ph.D.
Committee Member

Alessandro Veneziani, Ph.D.
Committee Member

Hao Gao, Ph.D.
Committee Member

Stuart M. Jefferies, Ph.D.

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the James T. Laney School of Graduate Studies

Date

Mathematical Models and Numerical Methods for Wavefront Reconstruction

By

Qing Chu
M.S., Graduate University of Chinese Academy of Sciences, 2008

Advisor: James G. Nagy, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2013

Abstract

Mathematical Models and Numerical Methods for Wavefront Reconstruction
By Qing Chu

Obtaining high resolution images of space objects from ground based telescopes is challenging, and often requires computational post processing methods to remove blur caused by atmospheric turbulence. In order for an image deblurring (deconvolution) algorithm to be effective, it is important to have a good approximation of the blurring operator. In space imaging, the blurring operator is defined in terms of the wavefront of light, and how it is distorted as it propagates through the atmosphere.

In this thesis we consider new mathematical models and algorithms to reconstruct the wavefront, which requires solving a large scale ill-posed inverse problem. We show that by exploiting and fusing information from multiple measurements, we are able to obtain better reconstructed wavefronts than existing methods. In addition, to fulfill the large scale requirement for astronomical uses, we present results of a parallel implementation utilizing the Trilinos project, a mathematical software library for solving problems from many academic and research fields.

Moreover, we study an symmetric successive over-relaxation (SSOR) preconditioner for this image reconstruction problem. Numerical results for different image reconstruction systems under variety of seeing conditions indicate good behavior of the SSOR preconditioner with respect to iteration numbers and computational time.

Mathematical Models and Numerical Methods for Wavefront Reconstruction

By

Qing Chu
M.S., Graduate University of Chinese Academy of Sciences, 2008

Advisor: James G. Nagy, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Mathematics
2013

Acknowledgments

A number of people have helped me to arrive at this point of my life. Even though words cannot express my truly gratefulness to all their kindness and support, I am trying to thank all of them for impacting my life.

First and foremost, I want to express my sincere gratitude to my advisor, Dr. James G. Nagy. Without your enthusiasm, patience, encouragement, inspiration and guidance, this thesis would not be possible. I appreciate all your contributions of time, ideas, advice and funding to support my Ph.D. study and research.

My sincere thanks also go to the other members of my thesis committee: Dr. Michele Benzi, Dr. Alessandro Veneziani, Dr. Stuart M. Jefferies and Dr. Hao Gao. Thank you for serving on my committee and thank you for the invaluable suggestions and insightful questions.

Thank you to the faculty, staff, postdoctoral fellows and graduate students in the Department of Mathematics and Computer Science of Emory University. Thank you for creating a friendly and stimulating environment and assisting me in many different ways.

I would like to thank all the teachers and friends in my life. I owe a huge debt of gratitude to Dr. Zhongze Li at Graduate University of Chinese Academy of Sciences. Thank you for leading me to and guiding me in the path of becoming a researcher. To my friends, for all the help and support from you, thank you all.

Last but not the least, I am indebted to my family. To my parents, I am grateful forever for the effort and the sacrifices you have made to educate me being a good person; thank you for your unconditional love and support. To my husband Zhen, without you, my life would not be so beautiful; thank you for your love, understanding and tolerance. To my daughter Ruiya, thank you for bringing me so much happiness; I love you always.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis mainly focuses on developing efficient approaches to restore images for astronomical uses. Various fields demand sharp visual observations, such as biomedical imaging, astronomical imaging and many others; see for example [21,51,56]. However, it is common that images are subject to loss of information due to image degradations. Image degradation is no small problem, and often requires an image restoration [48,75] technique to recover the original object. The presence of noise makes image restoration particularly difficult, and numerical methods are needed to compute efficient and reliable solutions. This chapter covers some mathematical topics of imaging problems, including mathematical models, ill-posedness and solution techniques.

Many image processing problems eventually require solving large scale ill-posed inverse problems [23]. Specifically, digital image processing techniques allow computers to assemble the collected data (e.g., electromagnetic energy) from a device into images that can be viewed by researchers. The assembling process is typically an inverse process: that is the image is reconstructed from indirect measurements of the corresponding object. Ill-posedness arises from the non-smoothing nature of the inverse process, which brings difficulties to computing a clear reconstruction in practice. Thus, numerical methods are developed to treat the ill-posedness, and efficiently and accurately solve this type of inverse problem.

1.1 Mathematical Model

Many restoration methods focus on modeling the blur and noise during the image formation process, and apply inversion algorithms to find an approximation of the original scene. Among the many excellent restoration techniques [1, 5], deconvolution requires solving an ill-posed inverse problem

$$g(x,y) = \int_{\mathcal{R}^2} k(x-\xi, y-\eta) f(\xi,\eta) d\xi d\eta + e(x,y)\,, \tag{1.1}$$

where f is the true object, g is the observed image, and e is additive noise. The kernel function k models the blurring operation, and is called the *point spread function* (PSF). By writing the kernel in the convolution form of $k(x-\xi, y-\eta)$, we automatically assume the blur is spatially invariant (the blurring kernel is independent of positions). A more general PSF, which includes the spatially variant case, would have the form $k(x,y;\xi,\eta)$.

1.1.1 Linear Least Squares Formulation

In many cases, the deconvolution model can be expressed as a linear system. The digital image deblurring problem is obtained from Equation (1.1) by discretizing the functions and approximating integration with a quadrature rule:

$$\boldsymbol{g} = \boldsymbol{K}\boldsymbol{f}_{\text{true}} + \boldsymbol{e}\,. \tag{1.2}$$

If the images are assumed to have $m \times n$ pixels, then $\boldsymbol{K} \in \mathcal{R}^{mn \times mn}$ and $\boldsymbol{g}$, $\boldsymbol{f}_{\text{true}}$, $\boldsymbol{e} \in \mathcal{R}^{mn}$. The matrix $\boldsymbol{K}$ is typically very ill-conditioned; more severe blurring usually corresponds with a more ill-conditioned $\boldsymbol{K}$. In the case of spatially invariant blur with $k(x-\xi, y-\eta)$, $\boldsymbol{K}$ can involve (depending on the imposed boundary conditions) circulant, Toeplitz, and Hankel structures [42].

For deconvolution, the matrix $\boldsymbol{K}$ is estimated based on experiments and observations, and often is not precisely known. This limits the quality of reconstructed images. One approach to improve the quality of the reconstructed

image is to collect more data, and solve a *multi-frame* image deconvolution problem. Specifically, several observations of the object are collected, resulting in multiple blurred image frames

$$\boldsymbol{g}_i = \boldsymbol{K}_i \, \boldsymbol{f}_{\text{true}} + \boldsymbol{e}_i \,, \quad i = 1, 2, \ldots, n_{\text{F}}$$

where n_{F} is the number of observed image frames. In this case, we can, for example, compute a reconstructed image by solving the overdetermined least squares problem

$$\min_{\boldsymbol{f}} \left\| \begin{bmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{n_{\text{F}}} \end{bmatrix} - \begin{bmatrix} \boldsymbol{K}_1 \\ \vdots \\ \boldsymbol{K}_{n_{\text{F}}} \end{bmatrix} \boldsymbol{f} \right\|_2^2 . \tag{1.3}$$

One problem with using this *ordinary* least squares (OLS) model to restore images is that we assume that each blurring kernel (and hence each matrix $\boldsymbol{K}_i$) is estimated with equal accuracy. In some cases, it might be more appropriate to use a *weighted* least squares (WLS) approach [33], to reconstruct the image $\boldsymbol{f}_{\text{true}}$. The idea is to assign larger weights, ω_i, to observation frames with better PSF estimates. Thus, in the case of spatially invariant blurs, the WLS solution can be written as

$$\min_{\boldsymbol{f}} \left\| \begin{bmatrix} \omega_1 \boldsymbol{g}_1 \\ \vdots \\ \omega_{n_{\text{F}}} \boldsymbol{g}_{n_{\text{F}}} \end{bmatrix} - \begin{bmatrix} \omega_1 \boldsymbol{K}_1 \\ \vdots \\ \omega_{n_{\text{F}}} \boldsymbol{K}_{n_{\text{F}}} \end{bmatrix} \boldsymbol{f} \right\|_2^2 . \tag{1.4}$$

1.1.2 Ill-Posedness and Regularization of Inverse Problems

French mathematician Jacques Hadamard introduced the basic concept of a well-posed problem in [37]. According to this definition, for a well-posed problem, the solution should be unique and exist for arbitrary data. More

specifically, a well-posed problem requires that its solution continuously depends on the data. This claim implies that if small perturbations of the data produce arbitrarily large changes in the solution, then the solution is not really a solution in the physical sense.

An ill-posed problem violates the requirements of well-posedness: solutions may not exist for all data; they may not be unique; and they may be unstable with respect to data perturbations. In a realistic situation, the data is never exact due to machine accuracy, noise and other possible aberrations during the measuring and recording process. Thus, the last requirement of well-posedness is particularly important. In practice, the coefficient matrix of a discrete linear system may be nonsingular, and thus in theory the linear system has a unique solution. However, the ill-posedness is reflected in the ill-conditioning of the coefficient matrix, and can be very sensitive to perturbations in the data. That is even if the solution of the ill-conditioned system exists and is unique, it often has very little physical resemblance to the solution corresponding to the noise free data. This can be seen by using the *singular value decomposition* (SVD) to analyze the discrete inverse solution of Equation (1.2).

If $\boldsymbol{K} \in \mathcal{R}^{mn \times mn}$, the SVD of $\boldsymbol{K}$ is

$$\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T ,$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $mn \times mn$ orthogonal matrices, and $\boldsymbol{\Sigma}$ is an $mn \times mn$ diagonal matrix with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{mn} \geq 0$ on its main diagonal. Suppose $\boldsymbol{K}$ is nonsingular, then the inverse solution takes the form

$$\boldsymbol{f} = \boldsymbol{K}^{-1}\boldsymbol{g} = \sum_{i=1}^{mn} \frac{\boldsymbol{u}_i^T \boldsymbol{g}}{\sigma_i} \boldsymbol{v}_i = \sum_{i=1}^{mn} \frac{\boldsymbol{u}_i^T \boldsymbol{g}_{\text{true}}}{\sigma_i} \boldsymbol{v}_i + \sum_{i=1}^{mn} \frac{\boldsymbol{u}_i^T \boldsymbol{e}}{\sigma_i} \boldsymbol{v}_i ,$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are left and right singular vectors of $\boldsymbol{K}$, and $\boldsymbol{g}_{\text{true}}$ represents the perfectly measured data without errors. From the above formulation, the inverse solution is composed of two terms: the exact solution of the object

$\sum_{i=1}^{mn} \frac{\boldsymbol{u}_i^T \boldsymbol{g}_{\text{true}}}{\sigma_i} \boldsymbol{v}_i$ and the error $\sum_{i=1}^{mn} \frac{\boldsymbol{u}_i^T \boldsymbol{e}}{\sigma_i} \boldsymbol{v}_i$. The following properties, which typically hold for discrete ill-posed problems [42], help to study the error term and approximate the exact solution.

- The singular values of the coefficient matrix decay to zero without a clear gap to indicate numerical rank. This property results in a large condition number of the system making it ill-conditioned.
- The singular vectors corresponding to smaller singular values are highly oscillatory, representing high-frequency information.
- The spectral components $|\boldsymbol{u}_i^T \boldsymbol{g}|$ of the noisy image decay faster than singular values of the coefficient matrix. This requirement to the data, or the noisy image, is called the discrete Picard condition [38].

The first two properties suggest that the general inverse solution $\boldsymbol{K}^{-1}\boldsymbol{g}$ is dominated by high-frequency components of the error term, which is magnified by division of small singular values. The last property states that by reconstructing the components corresponding to large singular values and filtering out the components corresponding to small singular values, a good approximated solution can be obtained.

Two popular types of methods are used to treat ill-posedness by additional information. One imposes a smoothness constraint to stabilize the inversion process, and is referred to as regularization. Examples include Tikhonov regularization (see [59, 71]) and regularizing conjugate gradient iteration (see [2, 18]); many other approaches are described in, for example, [38]. The other approach uses statistical properties as the additional information, and is called Bayesian methods. The rest of this subsection will go over Tikhonov regularization and some related aspects that will be used throughout this thesis.

Tikhonov regularization can be formulated in several ways. In the sense of SVD filtering, the computed solution can be written as

$$\boldsymbol{f}_{\text{filt}} = \sum_{i=1}^{mn} \phi_i \frac{\boldsymbol{u}_i^T \boldsymbol{g}}{\sigma_i} \boldsymbol{v}_i \,,$$

where $\boldsymbol{f}_{\text{filt}}$ is the filtered solution, $\boldsymbol{u}_i$, $\boldsymbol{v}_i$, $\boldsymbol{g}$ and σ_i are as previously defined, and ϕ_i is used to denote the filter coefficient.

In filtering methods, the solution $\boldsymbol{f}_{\text{filt}}$ is a weighted sum of the right singular vectors $\boldsymbol{v}_i$. For its coefficients, $\boldsymbol{u}_i^T \boldsymbol{g}/\sigma_i$ are the coefficients in a simple SVD solution as stated previously, and the filter factors ϕ_i are introduced to damp the highly oscillatory components corresponding to smaller singular values. We notice that the filter factors play important roles in filtering methods because they control the degree of filtering. That is for large singular values, let $\phi_i \approx 1$ to keep most information contained in the corresponding singular vectors; while for small singular values, let $\phi_i \approx 0$ to filter out the noise. To be more specific, for Tikhonov regularization, ϕ_i are defined for each singular component as

$$\phi_i = \frac{\sigma_i^2}{\sigma_i^2 + \alpha^2} \,,$$

where $\alpha > 0$ is defined as the regularization parameter, and thus $\phi_i \in (0, 1)$, for all i.

An equivalent minimization formulation of Tikhonov regularization is as follows:

$$\min_{\boldsymbol{f}} \left\{ \| \boldsymbol{g} - \boldsymbol{K}\boldsymbol{f} \|_2^2 + \alpha^2 \| \boldsymbol{f} \|_2^2 \right\} = \min_{\boldsymbol{f}} \left\| \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{K} \\ \alpha \boldsymbol{I} \end{bmatrix} \boldsymbol{f} \right\|_2^2 , \qquad (1.5)$$

and in the case of the multi-frame deconvolution corresponding to Equa-

tion (1.3), the regularized version is

$$\min_{\boldsymbol{f}} \left\| \begin{bmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{n_{\mathrm{F}}} \\ \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{K}_1 \\ \vdots \\ \boldsymbol{K}_{n_{\mathrm{F}}} \\ \alpha \boldsymbol{I} \end{bmatrix} \boldsymbol{f} \right\|_2^2 . \tag{1.6}$$

As we discussed previously, regularization methods are motivated by adding an additional smoothness constraint to the original minimization $\min_{\boldsymbol{f}} \|\boldsymbol{g} - \boldsymbol{K}\boldsymbol{f}\|_2^2$ to avoid the unrealistically large norm of the naïve inverse solution $\boldsymbol{f} = \boldsymbol{K}^{-1}\boldsymbol{g}$. By adding the additional penalty term $\alpha^2\|\boldsymbol{f}\|_2^2$, we ensure that both the norm of the residual and the norm of the solution are somewhat small. To summarize, the regularization parameter α controls the weighting between the two ingredients:

- The square norm of the residual $\|\boldsymbol{g} - \boldsymbol{K}\boldsymbol{f}\|_2^2$ measures how well $\boldsymbol{f}$ predicts the given noisy data $\boldsymbol{g}$. Therefore, if this term is too large, then $\boldsymbol{f}$ cannot be considered as a good solution. On the other hand, if this term is too small, then the solution $\boldsymbol{f}$ will be dominated by the error term.

- The square norm of the solution $\|\boldsymbol{f}\|_2^2$ measures the regularity of the solution. By SVD analysis, the small singular values correspond to high-frequency components in singular vectors. Therefore, by controlling the norm of $\boldsymbol{f}$, hopefully we can suppress the large noise components.

- α determines the balance between the two terms. The larger the regularization parameter α is, the more regularity we introduce to the solution $\boldsymbol{f}$, and the smaller α corresponds to more weight on fitting the noisy data term.

It is crucial to choose an appropriate regularization parameter α to balance the two terms. Unfortunately, there is no universal method suitable for every

single case; sometimes experimentation may be needed. However, there are methods that can be used to guide our computation. We list three ways to compute regularization parameters. In the case of Tikhonov regularization, we denote the norm of the residual with filtering coefficients ϕ_i and regularization parameter α as

$$\alpha = \|\boldsymbol{g} - \boldsymbol{K}\boldsymbol{f}_{\text{tik}}\|_2^2 = \sum_{i=1}^{mn} \left((1-\phi_i)\boldsymbol{u}_i^T\boldsymbol{g}\right)^2 = \sum_{i=1}^{mn} \left(\left(\tfrac{\alpha^2}{\sigma_i^2+\alpha^2}\right)\boldsymbol{u}_i^T\boldsymbol{g}\right)^2 . \qquad (1.7)$$

- **The discrepancy principle** is seeking to choose a regularized solution $\boldsymbol{f}_{\text{tik}}$ such that

  $$\|\boldsymbol{g} - \boldsymbol{K}\boldsymbol{f}_{\text{tik}}\|_2 = \tau\delta$$

  where $\tau > 1$, and δ is a good estimate of the error in the observation $\boldsymbol{g}$. For Tikhonov regularization, the residual norm by Equation (1.7), varies monotonically with respect to α, so there is a unique regularized solution satisfying the above criterion. Note that as $\delta \to 0$, or in other words, as the norm of the residual approaches 0, the regularized solution goes to the exact solution. Thus the major advantage of the discrepancy principle is its theoretical simplicity. One disadvantage of this method is the dependence on the prior δ. The computed regularization parameter is very sensitive to the accuracy of this estimate. In practice, under-smoothing (too small α) occurs in the case of too small an estimate of δ.

- **The L-curve** is a graphical tool to directly illustrate the balance between the size of the regularized solution and its fit to the given data when the regularization parameter varies. It is actually a log-log plot of the norm of a regularized solution versus the norm of the corresponding residual which contains two different parts: one part is quite flat where the regularized solution norm dominates; the other part is more vertical where the residual norm dominates. The log-log scale emphasizes the

different characteristics of the two parts. Intuitively, the corner of the L-curve corresponds to the optimal regularization parameter. An operational definition of the corner which involves computing the curvature of the L-curve can be found in [41]. Figure 1.1 provides an example of L-curve generated by software package Regularization Tools [40]. Although L-curve is often a good practical tool, it has theoretical dis-

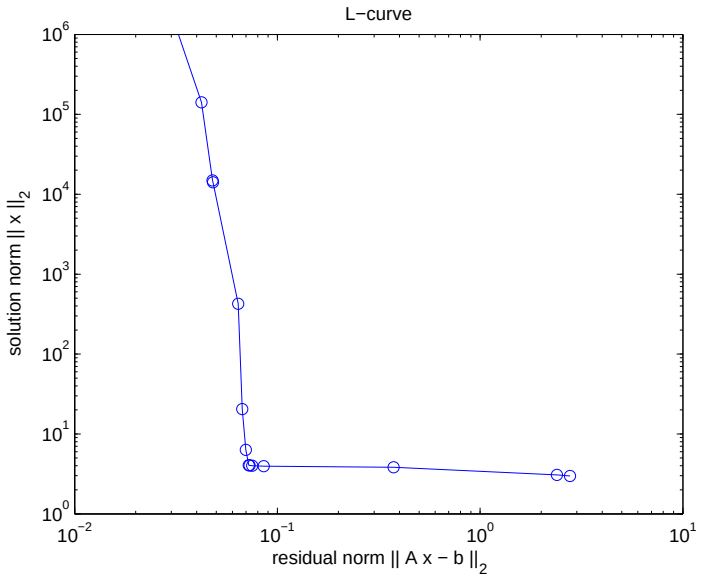

Figure 1.1: An example of the L-curve. The data is generated from the `shaw` problem in Regularization Tools. The regularized solution is obtained by Tikhonov regularization.

advantages compared to the discrepancy principle. In particular, the approximated solution fails to converge to the true solution as the dimension of the system goes to infinity, or as the error norm goes to zero [27, 74].

- **Generalized cross validation** (GCV) is different from the previous

two methods. It is aimed to find the regularization parameter which predicts the exact data as well as possible. GCV does not require a priori information. Instead, GCV determines the regularization parameter α that minimizes the GCV function. In the case of Tikhonov regularization, the GCV functional is given by

$$G(\alpha) = \frac{\|\boldsymbol{g} - \boldsymbol{K}\boldsymbol{f}_{\text{tik}}\|_2^2}{(\text{trace}(\boldsymbol{I}_{mn} - \boldsymbol{K}\boldsymbol{K}_{\text{tik}}^{\dagger}))^2}\,,$$

where $\boldsymbol{K}_{\text{tik}}^{\dagger} = (\boldsymbol{K}^T\boldsymbol{K} + \alpha^2\boldsymbol{I})^{-1}\boldsymbol{K}^T$ is the pseudoinverse of the matrix $\begin{bmatrix}\boldsymbol{K}\\ \alpha\boldsymbol{I}\end{bmatrix}$ in regularized system (1.5), mapping the given data $\boldsymbol{g}$ to the regularized solution $\boldsymbol{f}_{\text{tik}} = \boldsymbol{K}_{\text{tik}}^{\dagger}\boldsymbol{g}$. For the denominator, the trace of a matrix is the sum of its main diagonal entries, and is invariant under orthogonal transformation. Thus, by SVD of $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, and a diagonal matrix $\boldsymbol{\Phi}$ with filtering coefficients ϕ_i on its main diagonal, we have

$$\begin{aligned}
\text{trace}(\boldsymbol{I}_{mn} - \boldsymbol{K}\boldsymbol{K}_{\text{tik}}^{\dagger}) &= \text{trace}(\boldsymbol{I}_{mn} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T)\\
&= \text{trace}(\boldsymbol{U}(\boldsymbol{I}_{mn} - \boldsymbol{\Phi})\boldsymbol{U}^T)\\
&= \text{trace}(\boldsymbol{I}_{mn} - \boldsymbol{\Phi})\\
&= mn - \sum_{i=1}^{mn}\phi_i\,.
\end{aligned}$$

In this case, the expression of the GCV functional becomes

$$G(\alpha) = \frac{\sum_{i=1}^{mn}\left(\frac{\alpha^2\boldsymbol{u}_i^T\boldsymbol{g}}{\sigma_i^2+\alpha^2}\right)^2}{\left(\sum_{i=1}^{mn}\frac{\alpha^2}{\sigma_i^2+\alpha^2}\right)^2}\,.$$

The approximated solution of GCV method may also fail to converge as the error norm goes to zero [27].

1.1.3 Solution Techniques

Due to the large scale of the regularized system, iterative methods are often needed to solve the regularized systems (1.5) or (1.6). Many iterative algorithms, including steepest descent and the *conjugate gradient* (CG) method (e.g., the CGLS [18] and LSQR [58] implementation), can be applied. It is also worth mentioning that many iterative methods can also be directly used to solve the unregularized least squares problem as iterative regularization when the iteration index is treated as the regularization parameter. This is because these types of iterative methods have "semi-convergence" behavior, for which the early iterations reconstruct components of the solution corresponding to large singular values, while components corresponding to small singular values are reconstructed at later iterations. Thus the quality of the reconstruction is improved at early iterations and then degrades at later iterations. Furthermore, another hybrid type of method has been developed to overcome the drawback of iterative regularization in finding an appropriate stopping criteria. This type of method incorporates a regularization method within the iterations; see [39] for more details.

Special techniques can sometimes be applied to solve Tikhonov regularization problems. For example, in image deconvolution, if the blurring operation is spatially invariant, and we impose periodic boundary conditions to model the image scene outside the field of view, then $\boldsymbol{K}$ is a block circulant matrix with circulant blocks. In this case, instead of SVD, we factorize $\boldsymbol{K}$ by the spectral decomposition which has the form

$$\boldsymbol{K} = \mathcal{F}^* \boldsymbol{\Lambda} \mathcal{F}$$

where $\mathcal{F}$ is the 2-dimensional unitary discrete Fourier transform matrix, $\mathcal{F}^* = \mathcal{F}^{-1}$, and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{K}$. In this case, the solution to the Tikhonov regularized least squares problem can be

written as

$$\boldsymbol{f} = \mathcal{F}^* \left(|\boldsymbol{\Lambda}|^2 + \alpha^2 \boldsymbol{I} \right)^{-1} \bar{\boldsymbol{\Lambda}} \mathcal{F} \boldsymbol{g} \,,$$

where $|\boldsymbol{\Lambda}|^2$ is a diagonal matrix whose diagonal elements are the square modulus of those in $\boldsymbol{\Lambda}$, and $\bar{\boldsymbol{\Lambda}}$ is the complex conjugate of $\boldsymbol{\Lambda}$. Matrix-vector multiplications with $\mathcal{F}$ and $\mathcal{F}^*$ can be done very efficiently using, respectively, forward and inverse 2-dimensional *fast Fourier transforms* (FFT). In addition, the diagonal entries of $\boldsymbol{\Lambda}$ can be computed efficiently by computing an FFT of the PSF; for further details, see [75]. In the case of spatially invariant blurs with periodic boundary conditions, the solution of the multi-frame problem can be written as

$$\boldsymbol{f} = \mathcal{F}^* \left(\sum_{i=1}^{n_{\mathrm{F}}} |\boldsymbol{\Lambda}_i|^2 + \alpha^2 \boldsymbol{I} \right)^{-1} \left(\sum_{i=1}^{n_{\mathrm{F}}} \bar{\boldsymbol{\Lambda}}_i \mathcal{F} \boldsymbol{g}_i \right) .$$

Furthermore, if we consider the weighted least squares formulation, then

$$\boldsymbol{f} = \mathcal{F}^* \left(\sum_{i=1}^{n_{\mathrm{F}}} \omega_i |\boldsymbol{\Lambda}_i|^2 + \alpha^2 \boldsymbol{I} \right)^{-1} \left(\sum_{i=1}^{n_{\mathrm{F}}} \omega_i \bar{\boldsymbol{\Lambda}}_i \mathcal{F} \boldsymbol{g}_i \right) . \tag{1.8}$$

1.2 Outline of the Work

The contributions of this work have been to model and efficiently solve an inverse problem that estimates the PSF, which includes a parallel implementation and investigation of preconditioning techniques.

In the rest of this thesis, we will particularly focus on a restoration problem arising from astronomical imaging. Chapter 2 will provide some background information and mathematical models with an algorithm efficiently solving this problem. Chapter 3 will focus on the parallel implementation of the essential part of the algorithm, and preconditioning techniques are investigated in Chapter 4. Described in Chapter 5, a forward simulation implementation is provided to generate the testing data. Finally, some remarks and conclusions are provided in Chapter 6.

Chapter 2

Image Restoration in Astronomy

Ground based astronomy practice has a long history which could be traced back to prehistory. Ever since Galileo used his refractor telescope to observe the sky, telescopes became one of the most important tools for humans to understand the universe. Today, ground based telescopes are still very important tools in astronomy.

However, for many years, the resolution available to optical astronomers is limited by the properties of the atmosphere, such as absorption, dispersion and turbulence [43]. Note that since the dispersion of the air over the visible and infrared spectral bands, which is the working wavelength of ground based telescopes today, is very small, astronomers seek to remove distortions in recorded images caused by atmospheric turbulence.

2.1 The Effect of Atmospheric Turbulence

Atmospheric turbulence is caused by temperature distortions in the atmosphere, and consequently introduces the refraction index [6]. When the optical wavefronts from light reflected off a space object pass through the atmosphere, the original planar wavefronts are perturbed by the refractive index

variations. When reaching the telescope, the oscillatory wavefronts become non-planar. As a result, the images of space objects obtained by telescopes on the ground are degraded along the optical path through the atmosphere. A very simple example of atmospheric turbulence is sparkling stars on a clear night. Figure 2.1 illustrates the speckle pattern of an essentially point-like star.

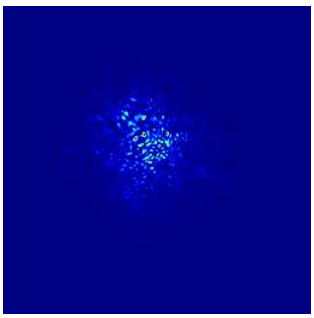

Figure 2.1: A point-like star is recorded in a speckle patten.

According to [63], the refraction fluctuation resulting from temperature differences within telescopes and along the path can be significantly reduced by carefully measuring the temperature and choosing an appropriate observation time, such as at night (during the day, the sun heats the instruments). The major source of the image degradation comes from turbulence due to the velocity gradients at the interface between air layers at different altitudes and the wind velocity which drives the layer movement. Therefore, the quality of images may vary under different seeing conditions. Typically, a sharp image is obtained during a clear night, and with the object directly above the telescope; conversely, a blurry image is more likely to be captured during the day, or when the object is close to the horizon. Figure 2.2 gives an example of image quality under different seeing conditions.

In a realistic imaging situation, the incoming wavefront will be distorted

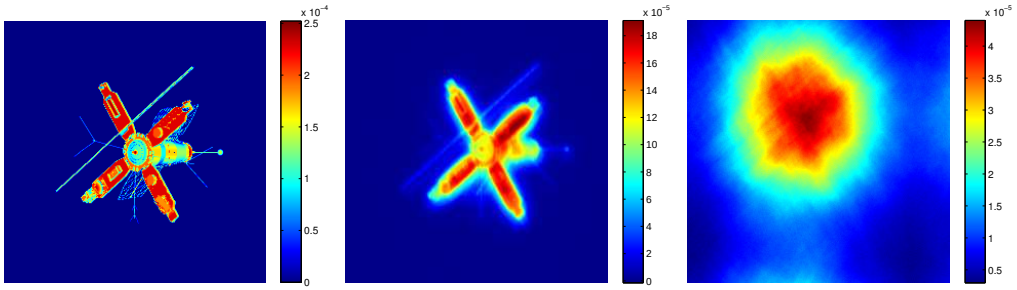

Figure 2.2: Illustration of the atmospheric blur. From left to right: the true object, a satellite, a simulation of the blurry image under good seeing conditions, and a simulation of the blurry image under poor seeing conditions.

by atmospheric turbulence. The severity of blurring caused by atmospheric turbulence depends on many factors, including weather, temperature, wavelength, and the diameter of the telescope. As we have pointed out, viewing objects directly above the telescope site on a clear night will have significantly better seeing conditions than looking during daylight hours at objects close to the horizon. Astronomers often quantify seeing conditions in terms of the ratio d/r_0, where d is the diameter of the telescope and r_0 is called the *Fried parameter*, which is related to the wavelength, and provides a statistical description of the level of atmospheric turbulence at a particular site. It is not essential to understand the precise definitions and characteristics of the Fried parameter, except that the variance of wavefront turbulence is proportional to $(d/r_0)^{5/3}$ [73]. That is,

- Good seeing conditions correspond to "small" d/r_0, such as $d/r_0 \lesssim 10$.
- Poor seeing conditions correspond to "large" d/r_0, such as as $d/r_0 \gtrsim 20$.

2.2 Mathematical Model

Although the blurring caused by the turbulence can be partially removed through sophisticated (and expensive) imaging devices, such as adaptive optics telescopes, computational post-processing techniques are also often needed to further improve the resolution of the image. As we have already illustrated that deconvolution can be used to address the loss of information problem due to image degradation. We use the deconvolution model as in Equation (1.1), and the corresponding discretization as linear least squares as shown in Equation (1.2).

2.2.1 The Blurring Kernel Model

The blurring operation for deconvolution problems is assumed to be known a priori. In order for deconvolution to produce a good approximation of the object, more study about refractive index, which is closely related to the blurring operation, or the PSF, should be done. Note that as referred to in the previous subsection, the refractive index variations that interfere with the propagation of light leads to atmospheric turbulence. As a result, wavefronts are non-planar at the telescope. Then by the Fourier optics model [35] of the atmospheric turbulence, the PSF k depends on the wavefront of incoming light at the telescope's mirror; if the wavefront function is known, then k is known. More specifically, $k(x, y; \xi, \eta) = k(x - \xi, y - \eta)$, with

$$k(s,t) = \left|\mathcal{F}^{-1}\left\{P(s,t)e^{i(1-\omega(s,t))}\right\}\right|^2 = \left|\mathcal{F}^{-1}\left\{P(s,t)e^{i\phi(s,t)}\right\}\right|^2 , \qquad (2.1)$$

where $\omega(s,t)$ is a function that models the shape of the wavefront of incoming light at the telescope, $i = \sqrt{-1}$, $P(s,t)$ is a characteristic function that models the shape of the telescope aperture (that is $P(s,t)$ is 1 inside the pupil and 0 outside, e.g., a circle or an annulus), $\mathcal{F}^{-1}$ is the 2-dimensional inverse Fourier transform, and $\phi(s,t) = 1 - \omega(s,t)$ is the phase error, or

the deviation from planarity of the wavefront ω. Then we can compute the PSF and consequently by deconvolution approach, the true object f can be computed.

Figure 2.3 shows examples of wavefront phases and kernels corresponding to the blurred images in Figure 2.2 for both good ($d/r_0 = 5$) and poor ($d/r_0 = 45$) seeing conditions. Note that one should interpret the wavefront "images" as color encoded contour plots. That is, these are not actually color images, but we display using a false color map so that it is easier to see how they fluctuate. The color bars show that the severity of the fluctuation, or deviation from planarity, is more significant in the case of poor seeing conditions.

2.2.2 Diffraction Limited Imaging Model

Another factor in ground based imaging that should be taken into account is the diffraction. Due to the limited size of the instrument, diffraction occurs and influences the imaging process. That is, ideally, the best image one can get is the diffraction limited image, which contains only diffraction from telescopes without any distortion or noise. When evaluating quality of reconstructed images, it is, perhaps, more appropriate to compare computed results to the diffraction limited image, and not to the true object.

In this subsection, we describe diffraction limited imaging. Here, Figure 2.4 shows the diffraction effect on images. In the ideal situation, where the atmosphere causes no distortion of the incoming wavefront, $\omega(s,t) = 1$ and $\phi(s,t) = 0$. In this *diffraction limited* case,

$$k_0(s,t) = \left|\mathcal{F}^{-1}\left\{P(s,t)\right\}\right|^2$$

where $P(s,t)$ is the pupil aperture function. Note that if $P(s,t) = 1$ for all s and t, then $k_0(s,t)$ is a Dirac delta function, and (except for noise) there is no distortion in the observed image g. However, in a realistic situation,

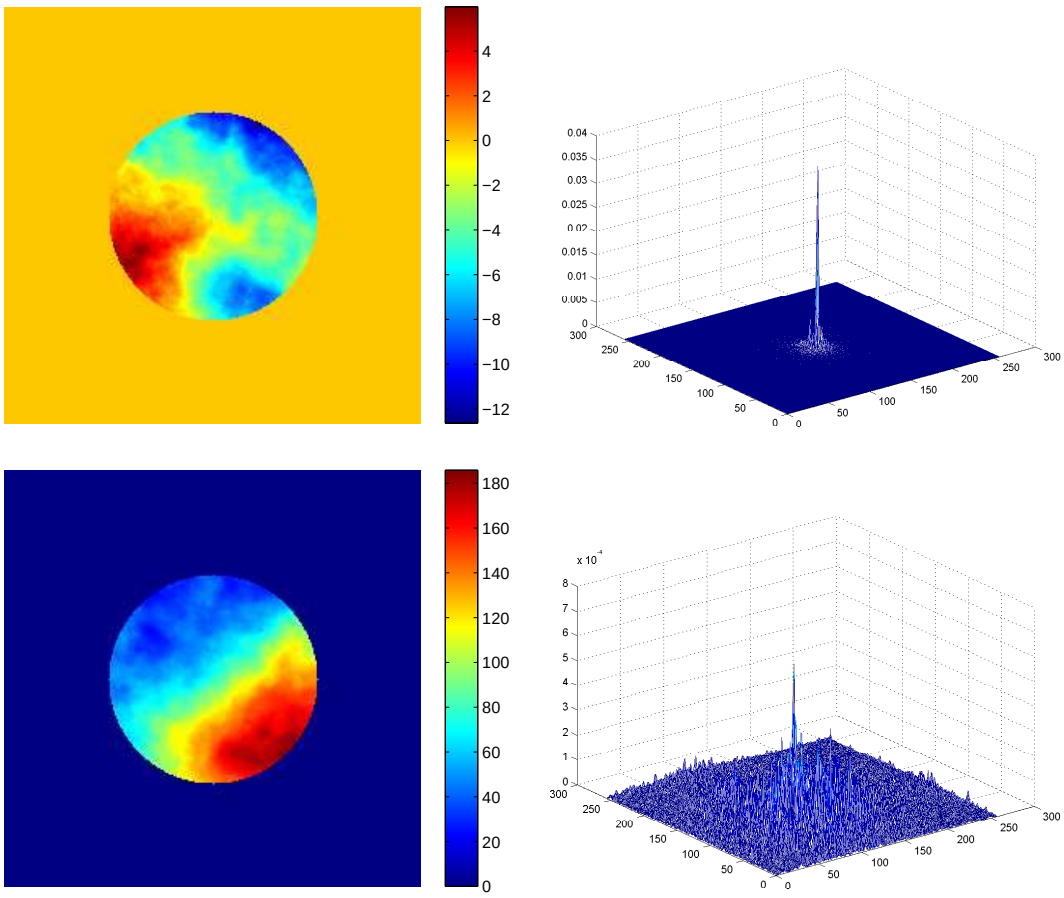

Figure 2.3: Examples of good and poor seeing conditions. Each row shows a color contour of a simulated wavefront incident at the telescope, and the corresponding blurring kernel. The top row illustrates good seeing conditions, with $d/r_0 = 5$, and the bottom row illustrates poor seeing conditions with $d/r_0 = 45$.

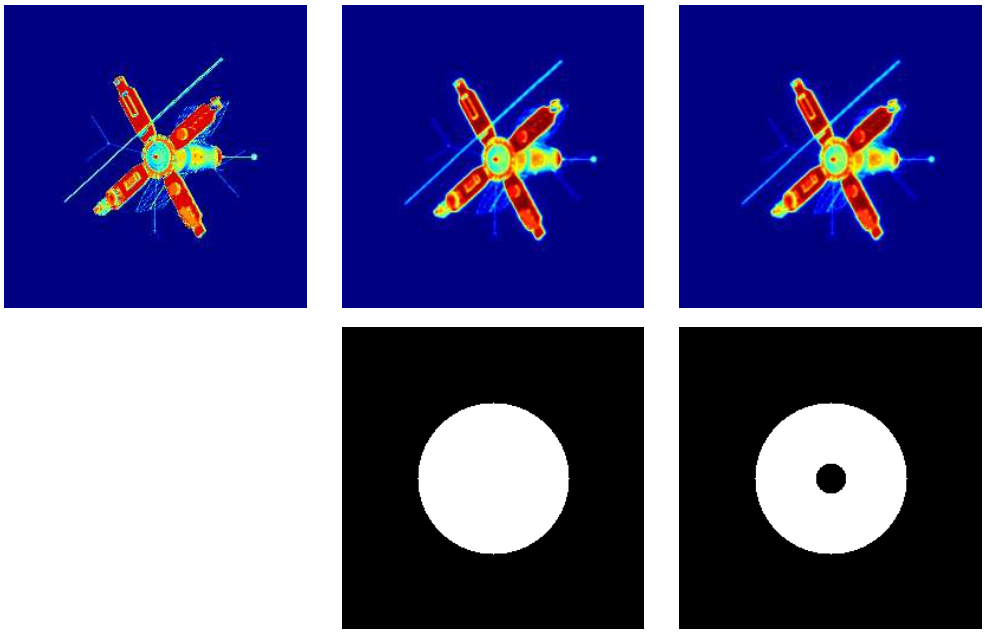

Figure 2.4: Examples of diffraction limited images. The top row shows the true object along with the diffraction limited observations with, respectively, a circular and an annulus pupil aperture. The diffraction limited images are not as sharp as the true image. The bottom row displays the pupil apertures used for this example.

$P(s,t) = 1$ in at most a finite region (e.g., within a circle or annulus defined by the telescope aperture), and thus it is impossible to obtain a perfect image. The best result we can hope to obtain is the noise free, diffraction limited image

$$f_0(x,y) = \int_{\mathcal{R}^2} k_0(x-\xi, y-\eta) f(\xi,\eta)\,.$$

2.2.3 Wavefront Reconstruction

By discretization and integration approximation, the matrix $\boldsymbol{K}$ of the linear least squares problem can be generated from Equation (2.1) provided ϕ, the wavefront, is known. In Chapter 1 we observed that if the matrix $\boldsymbol{K}$ is known, then very good results can be computed using fairly simple image deblurring algorithms [72, 75]. For a good estimate of $\boldsymbol{K}$, we need the wavefront ϕ. In practice, we can only access its gradients, which are measured using a *wavefront sensor* (WFS) in the telescope. A WFS is standard technology in adaptive optics systems, and many papers have been written about efficiently reconstructing the wavefront from the gradient measurements; see, for example [7, 47, 60]. That is, in addition to observing images, the telescope collects the additional data

$$\begin{bmatrix} \boldsymbol{\phi}_x \\ \boldsymbol{\phi}_y \end{bmatrix} = \begin{bmatrix} \boldsymbol{W}\boldsymbol{D}_x \\ \boldsymbol{W}\boldsymbol{D}_y \end{bmatrix} \boldsymbol{\phi} + \boldsymbol{\varepsilon}\,, \tag{2.2}$$

where $\boldsymbol{\phi}_x$ and $\boldsymbol{\phi}_y$ are discrete, noisy ($\boldsymbol{\varepsilon}$ is used to denote noise) measurements of the horizontal and vertical derivatives of $\boldsymbol{\phi}$; $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ are discrete, horizontal and vertical derivative operators. The precise structure of $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ depends on the sensor geometry [28, 47] but they essentially model finite difference approximations. $\boldsymbol{W}$ is a diagonal matrix containing ones and zeros; one for locations that fall within the pupil aperture, and zero otherwise. Several approaches have been proposed to efficiently solve Equation (2.2), including [31, 62]. For the computational experiments reported in this work,

we use the approach described in [8], which solves the least-squares minimization problem making use of Kronecker products and the generalized singular value decomposition.

A difficulty with using Equation (2.2) is that the gradient measurements are given on a relatively coarse grid compared to the observed image data. More specifically:

- Generally, to satisfy the Nyquist sampling theorem [57], the ratio of pupil aperture size to number of pixels on the *charge-coupled device* (CCD) array is 0.5. That is, if the CCD array that collects image data contains 256×256 pixels, then the diameter of the pupil aperture is 128 pixels.

- Gradient measurements are taken from a sensor on the mirror, so its grid is only on the pupil aperture region.

- Gradient measurements that are taken use 3×3 pixels. Thus, for a 256×256 CCD array, assuming a circular aperture with diameter equal to 128 pixels, we obtain at best gradient approximations on a 30×30 grid.

Figure 2.5 provides an illustration, where the small `x` marks denote pixels in the CCD array, and the small circles denote locations at which the gradient approximations are measured by the WFS.

Interpolation of the gradient data to a fine grid can be used to reconstruct the wavefront and corresponding PSFs. Although this approach may work well when the seeing conditions are good, the accuracy of the resulting wavefront and PSF may not allow for quality restorations.

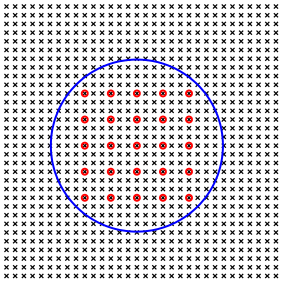

Figure 2.5: Example of a wavefront sensor. The large circle indicates the pupil aperture regions, the small `x` marks denote pixels in the CCD array, and the small circles denote locations at which the gradient approximations are measured by the WFS.

2.3 Frozen Flow Hypothesis and the Linear Formulation

From [53], in order to obtain a good wavefront estimate from its low resolution gradient measurements, we introduce the *frozen flow hypothesis* (FFH). FFH assumes that

- The entire spatial pattern of a random turbulent field is transported along with the wind velocity.
- Turbulent eddies do not change significantly as they are carried across the telescope by the wind.
- Typical velocities within the turbulence are small compared with the overall fluid (wind) velocity.

By FFH, multi-frame wavefronts or their gradients are included to construct the turbulence on a finer grid. That is, for a short period of time, the wavefronts, and the corresponding gradients, are assumed frozen. Then the

overlap regions of the adjacent frames contribute additional information for the reconstruction. Next, we provide more details about how FFH will be applied in our model.

2.3.1 Frozen Flow Hypothesis

FFH assumes that atmospheric turbulence can be modeled by a series of independent static layers, each moving across the telescope aperture with the prevailing wind at the altitude of that particular layer. Because of its simplicity, FFH is frequently used as the basis for numerical studies of ground telescope imaging problems, particularly in the modeling of *adaptive optics* (AO) systems [11]. Although FFH is observed not to hold in the real world over long time scales, a number of studies have shown that it is a reasonable approximation for short but still interesting periods [29, 61, 70].

To use FFH to reconstruct wavefront gradients, several frames of data are collected over a short time interval; each gives gradient measurements at a different set of grid points. This is illustrated in Figure 2.6 for two different velocity profiles. For ease of presentation, we consider only one layer; clearly multiple, overlapping layers will provide even more grid points in regions where the various layers overlap each other. Note that the composite grid resolution depends on the velocity profile; in the example shown in the top part of Figure 2.6, the velocity (direction and magnitude) remains constant from frame to frame, and the magnitude of the velocity is relatively small. A more extreme situation is illustrated in the bottom part of Figure 2.6, where there is a nonlinear change in the velocity from frame to frame.

2.3.2 Linear Model of the Wavefront Motion

Through appropriate coordinates, we could represent the movement of the wavefront and the gradients by shifting their images. Suppose $\phi_x(x, y)$ and

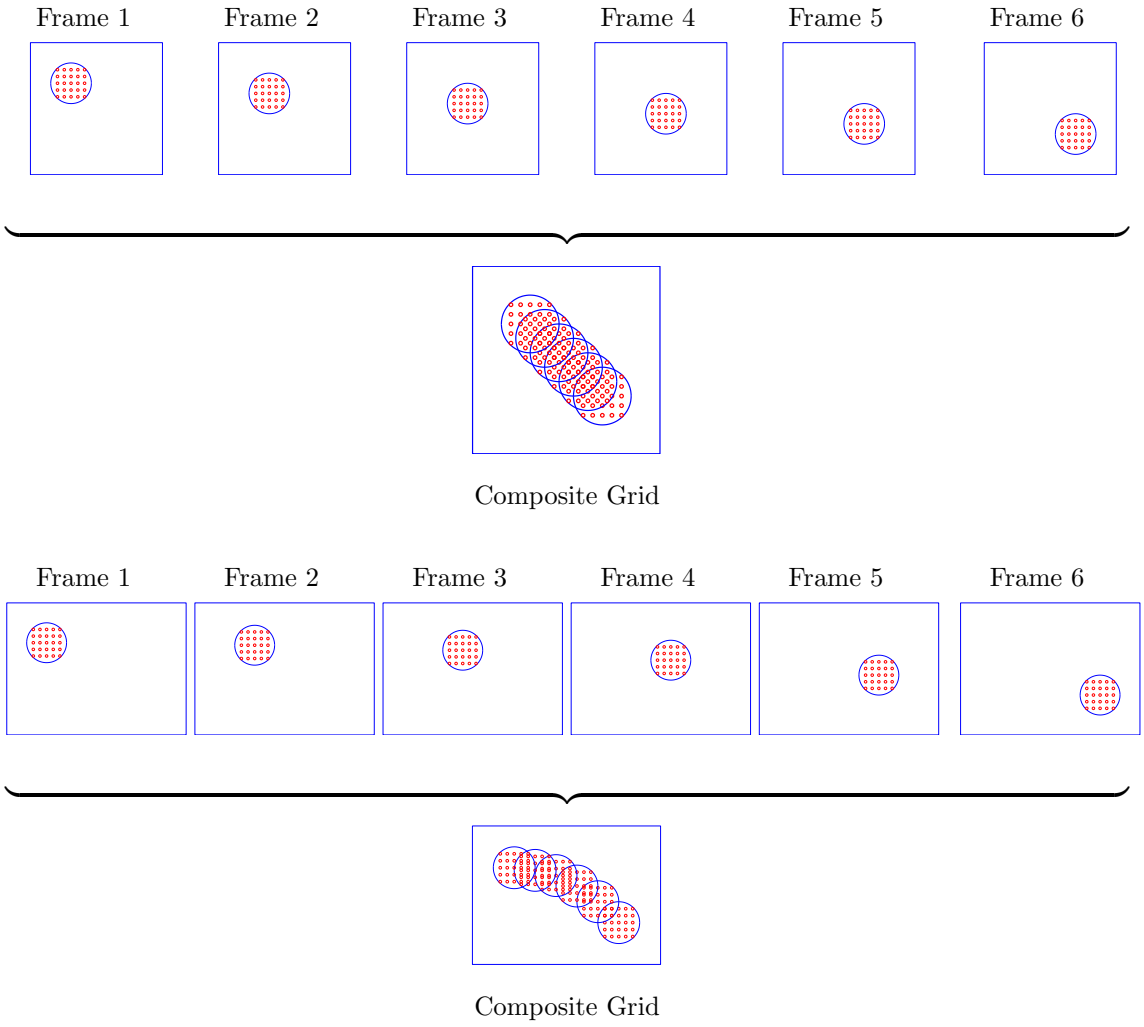

Figure 2.6: Illustrations of building a composite, high resolution grid using gradient grid points from several frames. The first two rows illustrate a situation when the velocity remains constant from frame to frame. The bottom two rows illustrate a situation when the velocity changes nonlinearly from frame to frame.

$\phi_y(x,y)$ are functions describing the shape of the gradient fields, and $\boldsymbol{\Phi}_x$ and $\boldsymbol{\Phi}_y$ are arrays of discrete samples of $\phi_x(x,y)$ and $\phi_y(x,y)$; that is,

$$\begin{aligned}\boldsymbol{\Phi}_x(i,j) &= \boldsymbol{D}_x\phi(x_i,y_j) = \phi_x(x_i,y_j)\,, \quad i=1,2,\ldots,n,\ \ j=1,2,\ldots,n,\\ \boldsymbol{\Phi}_y(i,j) &= \boldsymbol{D}_y\phi(x_i,y_j) = \phi_y(x_i,y_j)\,, \quad i=1,2,\ldots,n,\ \ j=1,2,\ldots,n,\end{aligned}$$

where $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ are discrete derivative operators as defined in [8]. $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ take the form:

$$\boldsymbol{D}_x = F \otimes H, \qquad \boldsymbol{D}_y = H \otimes F,$$

where $\otimes$ denotes Kronecker product, and H and F are $(n-1)\times n$ taking the form

$$H = \begin{bmatrix} 1 & -1 & & & \\ 0 & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix},\ F = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & \ddots & \\ & & & 1 & 1 \end{bmatrix}.$$

Using FFH, we can assume changes in the gradients, from frame to frame, are modeled as a rigid movement of $\phi_x(x,y)$ or $\phi_y(x,y)$. Rigid motion of coordinates in a plane can be described through a 3×3 affine transformation. So if we let $\boldsymbol{\Phi}_x^{(m)}$ and $\boldsymbol{\Phi}_y^{(m)}$ be the discretization of $\phi_x(x,y)$ and $\phi_y(x,y)$ after a rigid movement, then

$$\begin{aligned}\boldsymbol{\Phi}_x^{(m)}(i,j) &= \phi_x(\hat{x}_i,\hat{y}_j)\\ \boldsymbol{\Phi}_y^{(m)}(i,j) &= \phi_y(\hat{x}_i,\hat{y}_j)\,,\end{aligned} \quad \text{where} \quad \begin{bmatrix}\hat{x}_i\\ \hat{y}_j\\ 1\end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix} x_i\\ y_j\\ 1\end{bmatrix}, \tag{2.3}$$

where m denotes the frame index.

In practice the function ϕ_x and ϕ_y are not known at every point (x,y) (all that is known are the discrete values $\boldsymbol{\Phi}_x$ and $\boldsymbol{\Phi}_y$), so it may not be possible

to evaluate $\phi_x(\hat{x}_i, \hat{y}_j)$ or $\phi_y(\hat{x}_i, \hat{y}_j)$, unless $\hat{x}_i = x_{\hat{i}}$ and $\hat{y}_j = y_{\hat{j}}$ for integers $\hat{i}$ and $\hat{j}$, $1 \leq \hat{i} \leq n$ and $1 \leq \hat{j} \leq n$. However, approximations of $\phi_x(\hat{x}_i, \hat{y}_j)$ and $\phi_y(\hat{x}_i, \hat{y}_j)$ can be computed by interpolating known values of ϕ_x and ϕ_y near $\phi_x(\hat{x}_i, \hat{y}_j)$ and $\phi_y(\hat{x}_i, \hat{y}_j)$. Suppose, as illustrated in Figure 2.7, that $\phi_x(x_{\hat{i}}, y_{\hat{j}})$, $\phi_x(x_{\hat{i}+1}, y_{\hat{j}})$, $\phi_x(x_{\hat{i}}, y_{\hat{j}+1})$ and $\phi_x(x_{\hat{i}+1}, y_{\hat{j}+1})$ are four known pixel values surrounding the unknown value $\phi_x(\hat{x}_i, \hat{y}_j)$. Bilinear interpolation uses a weighted average of the four pixels surrounding $\phi_x(\hat{x}_i, \hat{y}_j)$ for the approximation. The same idea can be used to compute $\phi_y(\hat{x}_i, \hat{y}_j)$. Assuming, without loss of generality that the distance between pixel centers is one, then the weights for bilinear interpolation are given as

$$\begin{aligned}
\boldsymbol{\Phi}_x^{(m)}(i,j) &= \phi_x(\hat{x}_i, \hat{y}_j) \\
&\approx (1-\Delta x_i)(1-\Delta y_j)\phi_x(x_{\hat{i}}, y_{\hat{j}}) \;+\; (1-\Delta x_i)\Delta y_j \phi_x(x_{\hat{i}}, y_{\hat{j}+1}) \\
&\quad + \Delta x_i(1-\Delta y_j)\phi_x(x_{\hat{i}+1}, y_{\hat{j}}) \;+\; \Delta x_i \Delta y_j \phi_x(x_{\hat{i}+1}, y_{\hat{j}+1}),
\end{aligned}$$

where $\Delta x_i = \hat{x}_i - x_{\hat{i}}$ and $\Delta y_j = \hat{y}_j - y_{\hat{j}}$. This also holds for the $\boldsymbol{\Phi}_y^{(m)}(i,j)$.

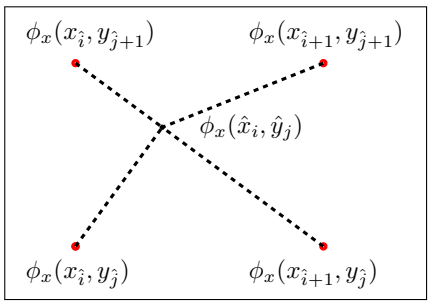

Figure 2.7: Illustration of bilinear interpolation, where a weighted average of the four known discrete values is used to approximate $\phi_x(\hat{x}_i, \hat{y}_j)$.

In order to define a matrix-vector multiplication to represent the interpolation above, we need to vectorize $\boldsymbol{\phi}_x = \text{vec}(\boldsymbol{\Phi}_x)$, $\boldsymbol{\phi}_y = \text{vec}(\boldsymbol{\Phi}_y)$ and $\boldsymbol{\phi}_x^{(m)} = \text{vec}(\boldsymbol{\Phi}_x^{(m)})$, $\boldsymbol{\phi}_y^{(m)} = \text{vec}(\boldsymbol{\Phi}_y^{(m)})$ from the discrete data arrays (e.g.,

through lexicographical ordering), and thus we can write

$$\boldsymbol{\phi}_x^{(m)} = \boldsymbol{A}_m \boldsymbol{\phi}_x, \quad \boldsymbol{\phi}_y^{(m)} = \boldsymbol{A}_m \boldsymbol{\phi}_y \,,$$

where $\boldsymbol{A}_m$ is a sparse matrix that contains the interpolation weights. Specifically, the k-th row of $\boldsymbol{A}_m$ contains the weights for the pixel in the k-th entry of $\boldsymbol{\phi}_x^{(m)}$ or $\boldsymbol{\phi}_y^{(m)}$. That is, in the case of bilinear interpolation, there are at most four nonzero entries per row, given by

$$(1-\Delta x_i)(1-\Delta y_j), \quad (1-\Delta x_i)\Delta y_j, \quad \Delta x_i(1-\Delta y_j), \quad \Delta x_i \Delta y_j \,.$$

For the consideration of data storage and computational cost for practical problems, we emphasize that by using a sparse data format (e.g., compressed row [26]) to represent $\boldsymbol{A}_m$, we need only keep track of the nonzero entries and their locations in the matrix $\boldsymbol{A}_m$. Moreover, this discussion assumes that the affine transformation used in Equation (2.3) is known from wind velocity information.

As explained in section 2.2.3, we cannot measure the wavefront directly, but instead we observe gradients on a low resolution grid. The mathematical formulation of this process is given by:

$$\phi_x^{(m)} = \boldsymbol{R}\boldsymbol{W}\boldsymbol{A}_m\boldsymbol{D}_x\phi \;\text{ and }\; \phi_y^{(m)} = \boldsymbol{R}\boldsymbol{W}\boldsymbol{A}_m\boldsymbol{D}_y\phi,$$

where $\boldsymbol{W}$ is an indicator matrix that grabs a specified section of $\boldsymbol{\phi}_x$ and $\boldsymbol{\phi}_y$, and $\boldsymbol{R}$ is a sparse downsampling (or restriction) matrix that transforms high resolution data to a lower resolution. More specifically,

- $\boldsymbol{W}$ is a full row rank, under-determined matrix with zeros and ones. For example, suppose

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_1 & \phi_4 & \phi_7 \\ \phi_2 & \phi_5 & \phi_8 \\ \phi_3 & \phi_6 & \phi_9 \end{bmatrix}, \qquad \phi = \mathrm{vec}(\boldsymbol{\phi})$$

and

$$\boldsymbol{W} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

then

$$\mathrm{vec}\left(\begin{bmatrix} \phi_4 & \phi_7 \\ \phi_5 & \phi_8 \end{bmatrix} \right) = \boldsymbol{W}\phi.$$

- For $\boldsymbol{R}$, suppose $\boldsymbol{\phi} \in \mathcal{R}^{n \times n}$ is an array of data on a high resolution, $n \times n$ grid, and that we want to downsample this to an $m \times m$ grid, where $s = n/m$ is an integer. Then

  $$\boldsymbol{R} = (\boldsymbol{R}_1 \otimes \boldsymbol{R}_1)/(s^2)\,,$$

  where $\boldsymbol{R}_1 = \boldsymbol{I}_m \otimes \mathbf{1}_\mathbf{s}^\mathbf{T}$, $\boldsymbol{I}_m$ is an $m \times m$ identity matrix, and $\mathbf{1}_\mathbf{s}^\mathbf{T}$ is vector of length s containing all ones. Note that $\boldsymbol{R}$ is under-determined, but has full row rank.

Assuming that we obtain m frames of data, we have

$$\begin{bmatrix} \boldsymbol{\phi}_x^{(1:m)} \\ \boldsymbol{\phi}_y^{(1:m)} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{AD}_x \\ (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{AD}_y \end{bmatrix} \boldsymbol{\phi}$$

where $\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{D}_x, \boldsymbol{D}_y$ were previously described, $\otimes$ denotes Kronecker product, $\boldsymbol{I}$ is an $m \times m$ identity matrix, and

$$\boldsymbol{\phi}_x^{(1:m)} = \begin{bmatrix} \boldsymbol{\phi}_x^{(1)} \\ \boldsymbol{\phi}_x^{(2)} \\ \vdots \\ \boldsymbol{\phi}_x^{(m)} \end{bmatrix}, \quad \boldsymbol{\phi}_y^{(1:m)} = \begin{bmatrix} \boldsymbol{\phi}_y^{(1)} \\ \boldsymbol{\phi}_y^{(2)} \\ \vdots \\ \boldsymbol{\phi}_y^{(m)} \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \vdots \\ \boldsymbol{A}_m \end{bmatrix}.$$

In this model, $\boldsymbol{\phi}$ represents a large, global wavefront, but at the time each frame of data is collected, the telescope detects only a small subregion of information, which is modeled by the matrix $\boldsymbol{W}$.

Note that it is impossible to reconstruct the whole global wavefront ϕ because we cannot collect enough gradient data to cover the whole wavefront region. However, we can construct a composite of the collected information on a high resolution grid by two steps: first, solve an underdetermined least squares problem for the composite horizontal and vertical gradient measurements:

$$\begin{bmatrix} \boldsymbol{\phi}_x^{\text{composite}} \\ \boldsymbol{\phi}_y^{\text{composite}} \end{bmatrix} = \arg\min_{\boldsymbol{\phi}_x, \boldsymbol{\phi}_y} \left\| \begin{bmatrix} \boldsymbol{\phi}_x^{(1:m)} \\ \boldsymbol{\phi}_y^{(1:m)} \end{bmatrix} - \begin{bmatrix} (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}\boldsymbol{\phi}_x \\ (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}\boldsymbol{\phi}_y \end{bmatrix} \right\|_2^2 \tag{2.4}$$

and compute gradient measurements on finer grids for each frame by

$$\begin{bmatrix} \hat{\boldsymbol{\phi}}_x^{(1:m)} \\ \hat{\boldsymbol{\phi}}_y^{(1:m)} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{I} \otimes \boldsymbol{W})\boldsymbol{A}\boldsymbol{\phi}_x^{\text{composite}} \\ (\boldsymbol{I} \otimes \boldsymbol{W})\boldsymbol{A}\boldsymbol{\phi}_y^{\text{composite}} \end{bmatrix};$$

next, use the computed gradients to solve an underdetermined least squares problem for each frame

$$\boldsymbol{\phi}^{(i)} = \arg\min_{\boldsymbol{\phi}} \left\| \begin{bmatrix} \hat{\boldsymbol{\phi}}_x^{(i)} \\ \hat{\boldsymbol{\phi}}_y^{(i)} \end{bmatrix} - \begin{bmatrix} \boldsymbol{D}_x \\ \boldsymbol{D}_y \end{bmatrix} \boldsymbol{\phi} \right\|_2^2, \quad i = 1, 2, \ldots, m\,. \tag{2.5}$$

We remark that both (2.4) and (2.5) are underdetermined, and can be sensitive to noise in the measured data. We use Tikhonov regularization [36, 72] to obtain the reconstructed images.

2.3.3 Wavefront Motion in the Multi-Layered Assumption

The previous subsection focused on the single layer wavefront problem. Considering a more realistic model, the atmosphere above the telescope can be

split into several dominant layers, which move with different velocities [30]. For the multi-layered model, in [22], we assume that the composite high resolution wavefront at the telescope is the sum of the high resolution wavefront at each altitude. Then we have

$$\phi = \sum_{j=1}^{L} c_j \phi_j \tag{2.6}$$

where L is the number of turbulent layers, ϕ is the wavefront at the telescope, ϕ_j is the wavefront of the jth layer, and c_j is a constant such that $\sum_{j=1}^{L} c_j = 1$ and specifies relative dominance in the contribution of each layer to the total turbulent field. For example, if it is assumed that all layers contribute equally to the total wavefront hitting the telescope, then $c_1 = c_2 = \cdots = c_L$. On the other hand, if it is assumed that the kth layer is the dominant layer of the atmospheric turbulence, then $c_k \gg c_j$, $j = 1, \ldots, k-1, k+1, \ldots, L$. In our experiments, we always assume $c_1 = c_2 = \ldots = c_L$.

Similar to the single layer turbulent model, in the multi-layered case, we model this problem as

$$\begin{bmatrix} \boldsymbol{\phi}_x^{(1:m)} \\ \boldsymbol{\phi}_y^{(1:m)} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_1\boldsymbol{D}_x & \cdots & (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_L\boldsymbol{D}_x \\ (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_1\boldsymbol{D}_y & \cdots & (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_L\boldsymbol{D}_y \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_1 \\ \vdots \\ \boldsymbol{\phi}_L \end{bmatrix}$$

where $\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{D}_x, \boldsymbol{D}_y$ were previously described, $\otimes$ denotes Kronecker product, $\boldsymbol{I}$ is an $m \times m$ identity matrix, $\boldsymbol{\phi}_j$ and $\boldsymbol{A}_j$, $j = 1, \ldots, L$, denotes the wavefront and the matrix that defines the motion of the atmosphere for layer j, and

$$\boldsymbol{A}_j = \begin{bmatrix} \boldsymbol{A}_{1,j} \\ \boldsymbol{A}_{2,j} \\ \vdots \\ \boldsymbol{A}_{m,j} \end{bmatrix}, \quad j = 1, 2, \ldots, L.$$

Note $A_{i,j}$ is the motion matrix for the ith frame of the jth layer.

We need to reconstruct the wavefront for each frame. This can be done by two steps. First, solve

$$\begin{bmatrix} \boldsymbol{\phi}_{x,(1:L)}^{\text{composite}} \\ \boldsymbol{\phi}_{y,(1:L)}^{\text{composite}} \end{bmatrix} = \underset{\boldsymbol{\phi}_{x,(1:L)},\boldsymbol{\phi}_{y,(1:L)}}{\arg\min} \left\| \begin{bmatrix} \boldsymbol{\phi}_x^{(1:m)} \\ \boldsymbol{\phi}_y^{(1:m)} \end{bmatrix} - \begin{bmatrix} (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_1\boldsymbol{\phi}_{x,1} & \cdots & (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_L\boldsymbol{\phi}_{x,L} \\ (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_1\boldsymbol{\phi}_{y,1} & \cdots & (\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_L\boldsymbol{\phi}_{y,L} \end{bmatrix} \right\| \tag{2.7}$$

and compute

$$\begin{bmatrix} \hat{\boldsymbol{\phi}}_x^{(1:m)} \\ \hat{\boldsymbol{\phi}}_y^{(1:m)} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{I} \otimes \boldsymbol{W})\boldsymbol{A}_1\boldsymbol{\phi}_{x,1}^{\text{composite}} & \cdots & (\boldsymbol{I} \otimes \boldsymbol{W})\boldsymbol{A}_L\boldsymbol{\phi}_{x,L}^{\text{composite}} \\ (\boldsymbol{I} \otimes \boldsymbol{W})\boldsymbol{A}_1\boldsymbol{\phi}_{y,1}^{\text{composite}} & \cdots & (\boldsymbol{I} \otimes \boldsymbol{W})\boldsymbol{A}_L\boldsymbol{\phi}_{y,L}^{\text{composite}} \end{bmatrix},$$

where $\boldsymbol{\phi}_{x,j}$, $\boldsymbol{\phi}_{y,j}$, $j = 1, \ldots, L$, are composite gradients of the jth layer, $\boldsymbol{\phi}_x^{(1:m)}$, $\boldsymbol{\phi}_y^{(1:m)}$, $\hat{\boldsymbol{\phi}}_x^{(1:m)}$, $\hat{\boldsymbol{\phi}}_y^{(1:m)}$, $\boldsymbol{I}$, $\boldsymbol{R}$ and $\boldsymbol{W}$ are defined as before, and

$$\boldsymbol{\phi}_{x,(1:L)} = \begin{bmatrix} \boldsymbol{\phi}_{x,1} \\ \boldsymbol{\phi}_{x,2} \\ \vdots \\ \boldsymbol{\phi}_{x,L} \end{bmatrix}, \quad \boldsymbol{\phi}_{y,(1:L)} = \begin{bmatrix} \boldsymbol{\phi}_{y,1} \\ \boldsymbol{\phi}_{y,2} \\ \vdots \\ \boldsymbol{\phi}_{y,L} \end{bmatrix},$$

Next, solve

$$\boldsymbol{\phi}^{(i)} = \arg\min_{\boldsymbol{\phi}} \left\| \begin{bmatrix} \hat{\boldsymbol{\phi}}_x^{(i)} \\ \hat{\boldsymbol{\phi}}_y^{(i)} \end{bmatrix} - \begin{bmatrix} \boldsymbol{D}_x \\ \boldsymbol{D}_y \end{bmatrix} \phi \right\|_2^2, \quad i = 1, \ldots, m. \tag{2.8}$$

Again, (2.7) and (2.8) are underdetermined.

2.4 Numerical Results

2.4.1 Parameter Settings

In this section, using MATLAB, we present results from some numerical experiments using a realistic model of atmospheric turbulence. Specifically, assume that the diameter of the telescope is 3.7 m, the light wavelength is

0.744×10^{-6} m, the propagation distance is 25 km, there are three dominant layers moving in different directions, and we assume that 50 frames of data are collected. Gaussian white noise (1%) was added to the measured gradients, as well as to the blurred images. We report on results using these basic parameters, modifying only the turbulence strength.

Table 2.1: Wind velocities in pixel/frame of different turbulent layers.

layer	location (km)	x-velocity	y-velocity
1	0	0.1730	0
2	11	0	0.9686
3	15	0.3669	0.3669

There are three regularized systems to be solved for reconstructing the images: first, we need to compute gradients on a finer grid (2.7); next, we construct wavefronts for each frame (2.8); and finally, to restore the image, we compute an FFT-based WLS solution using Equation (1.8). To choose weights for our experiments, since we know the true PSFs, we compute the relative errors of PSFs obtained from the reconstructed wavefronts, and use the reciprocals of the errors as the weights. We realize that this scheme for choosing weights is not possible for a realistic problem, but a very similar approach based on the sampling of the overlapping frames could be used in practice; frames with better sampling have larger weights, while frames with poor sampling have smaller weights.

For each system, we need to assign regularization parameters, and since (2.7) and (2.8) are solved iteratively using LSQR, we need to choose a stopping tolerance for these. Specifically, in the first least squares problem we used Tikhonov regularization with a regularization parameter 1e-3 and LSQR

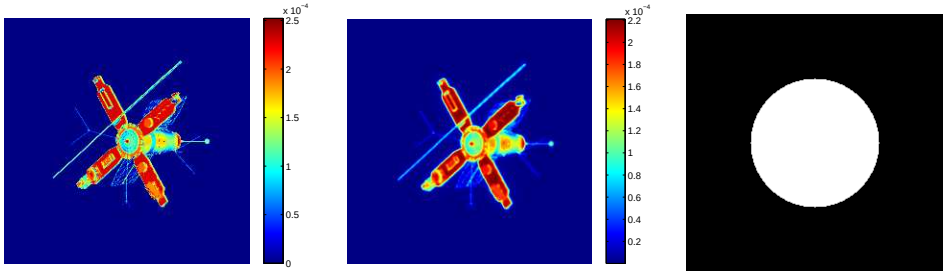

Figure 2.8: Some images needed for comparison. From left to right are the true object, the diffraction limited image and the pupil mask.

to solve, with a stopping tolerance (relative residual) of 1e-3. In the second least squares problem, we again used 1e-3 as a regularization parameter and LSQR to solve, with a stopping tolerance (relative residual) of 1e-6. In the final multi-frame deconvolution problem, we used Tikhonov regularization and FFT-based spectral decompositions in GCV to choose the regularization parameter.

The size of the least squares system for the FFH reconstruction depends on the image size and wind velocity (which determines the size of the composite gradients), as well as the downsampling factor. In our experiments, the image size of each frame is 256×256, and using the wind velocity listed in Table 2.1, the grid for the composite gradients is 304×304. The downsampling factor is 4, resulting in a low resolution grid that is 64×64. Therefore, the size of the least squares problem in (2.7) is $204,800 \times 277,248$. The true image, the pupil mask and the corresponding the diffraction limited image are illustrated in Figure 2.8

2.4.2 Experiment 1: Good Seeing Conditions

In this first test, we simulate motion of the wavefront using the velocity profile listed in Table 2.1 with $d/r_0 = 5$. In this case, the seeing conditions are good, and so we expect a smooth wavefront, and an observed image with

very little blurring. This is illustrated in Figure 2.10 and 2.13.

In order to restore the image, we need to compute the composite horizontal and vertical gradient measurements on a finer grid by solving the least squares problem (2.7), and then reconstruct the approximate wavefront phase by solving the least squares problem (2.8). The reconstructed high resolution gradients, wavefront and PSF are shown in Figure 2.9 and Figure 2.10.

We also use a naïve approach, which simply interpolates the gradients to a finer grid. To obtain a quantitative measure of the effectiveness of our FFH approach, we compare the reconstructed PSFs with the true PSFs; Figure 2.11 shows a plot of the relative errors of the reconstructed PSFs for each of the 50 frames. The FFH approach produces much better approximations of the PSFs than the naïve approach. The more accurate PSFs result in a slightly better reconstructed image, which are displayed in Figure 2.13; to better evaluate the reconstruction, Figure 2.13 provides the diffraction limited and the blurred images for comparison. Because the seeing conditions are very good in this example, the naïve approach does quite well, and there is only a slight improvement with our FFH approach.

The numerical results we show in Figure 2.11 and the middle reconstructions in Figure 2.13 are based on the true wind profile, with which the affine transformations in (2.3) are generated. In order to consider more realistic situations, we perturb the wind velocities for different layers. Figure 2.12 and the bottom row in Figure 2.13 show the relative errors of the computed PSFs and reconstructed images with noisy wind velocities under the good seeing conditions, and noises for different layers have different angles and speeds. Note that Figure 2.12 implies that with 1% noise wind profile, the relative errors of the PSFs are even smaller than with the true wind profile. This phenomenon may result from the noisy measured gradients. Since both the measured data and the wind profile are noisy, they may be balanced out. On the other hand, with 5% noise, the relative errors are worse than that

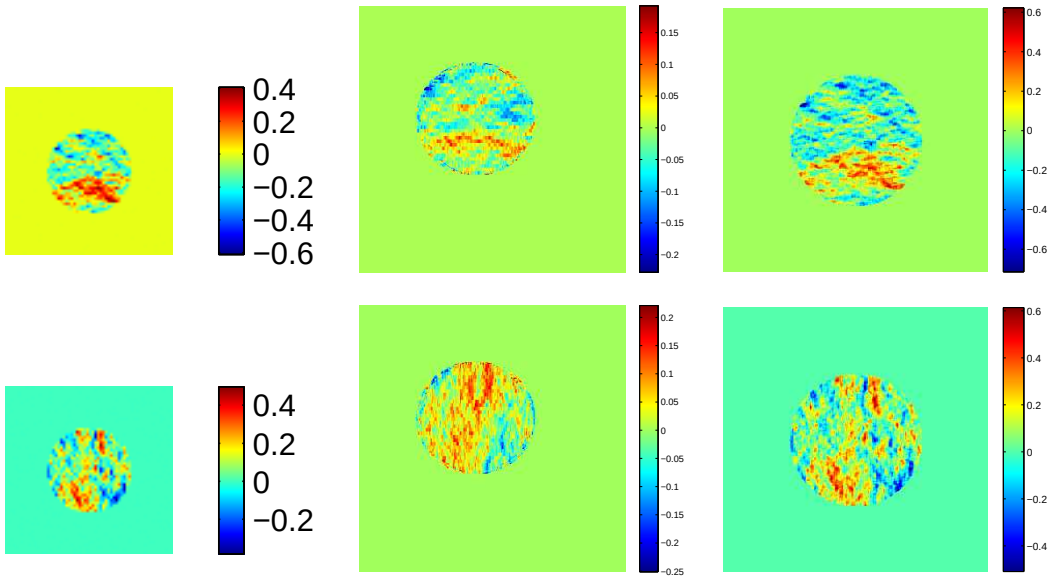

Figure 2.9: The measured and the reconstructed gradients on coarse and fine grids respectively ($d/r_0 = 5$). The top row shows the measured gradients, the reconstructed composite gradients of the 1st layers and the reconstructed gradients of the 1st frame along the horizontal direction. The bottom row shows the measured gradients, the reconstructed composite gradients of the 1st layer and the reconstructed gradients of the 1st frame along the vertical direction.

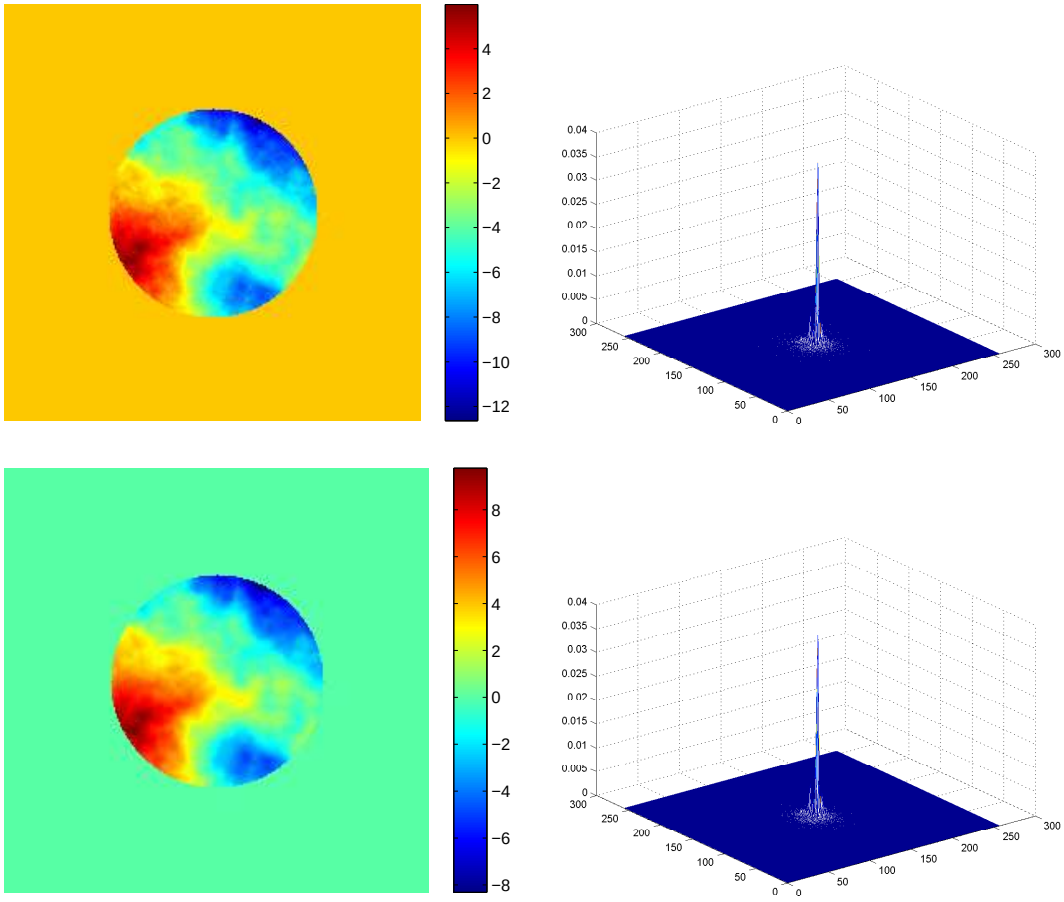

Figure 2.10: The true and the reconstructed phase and PSF of the 1st frame ($d/r_0 = 5$). The top row shows the true wavefront incident at the telescope and the corresponding blurring kernel. The bottom row shows the reconstructed wavefront and the corresponding blurring kernel.

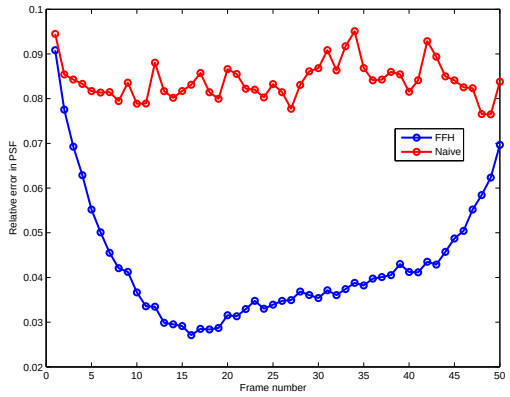

Figure 2.11: Plot of the relative errors of the reconstructed PSFs for each frame using two different approaches, the FFH and naïve approach, when $d/r_0 = 5$.

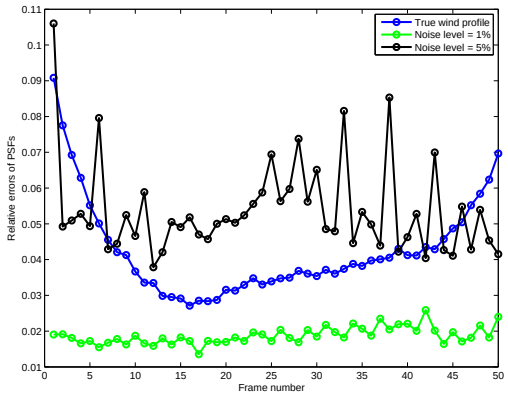

Figure 2.12: Plot of the relative errors of the reconstructed PSFs, with 0%, 1% and 5% noise in wind profile respectively, by FFH approach when $d/r_0 = 5$.

with the true wind velocity. However, the reconstructed images in both noisy cases look almost the same as that with true wind velocity.

2.4.3 Experiment 2: Poor Seeing Conditions

In this example, we simulate motion of the wavefront for the case $d/r_0 = 20$. In this case, the seeing conditions are poor, and so we expect to see more severe oscillations in the wavefront, and much more blurring in the observed image as compared with the previous example. This is illustrated in Figure 2.15 and 2.18.

Again, we compared our FFH approach with the naïve scheme of interpolating the low resolution gradients to the higher resolution grid by investigating the quality of the reconstructed PSFs for each approach.

Figures 2.16 shows the relative errors of the reconstructed PSFs, and Figure 2.18 displays the reconstructed images obtained by the two approaches

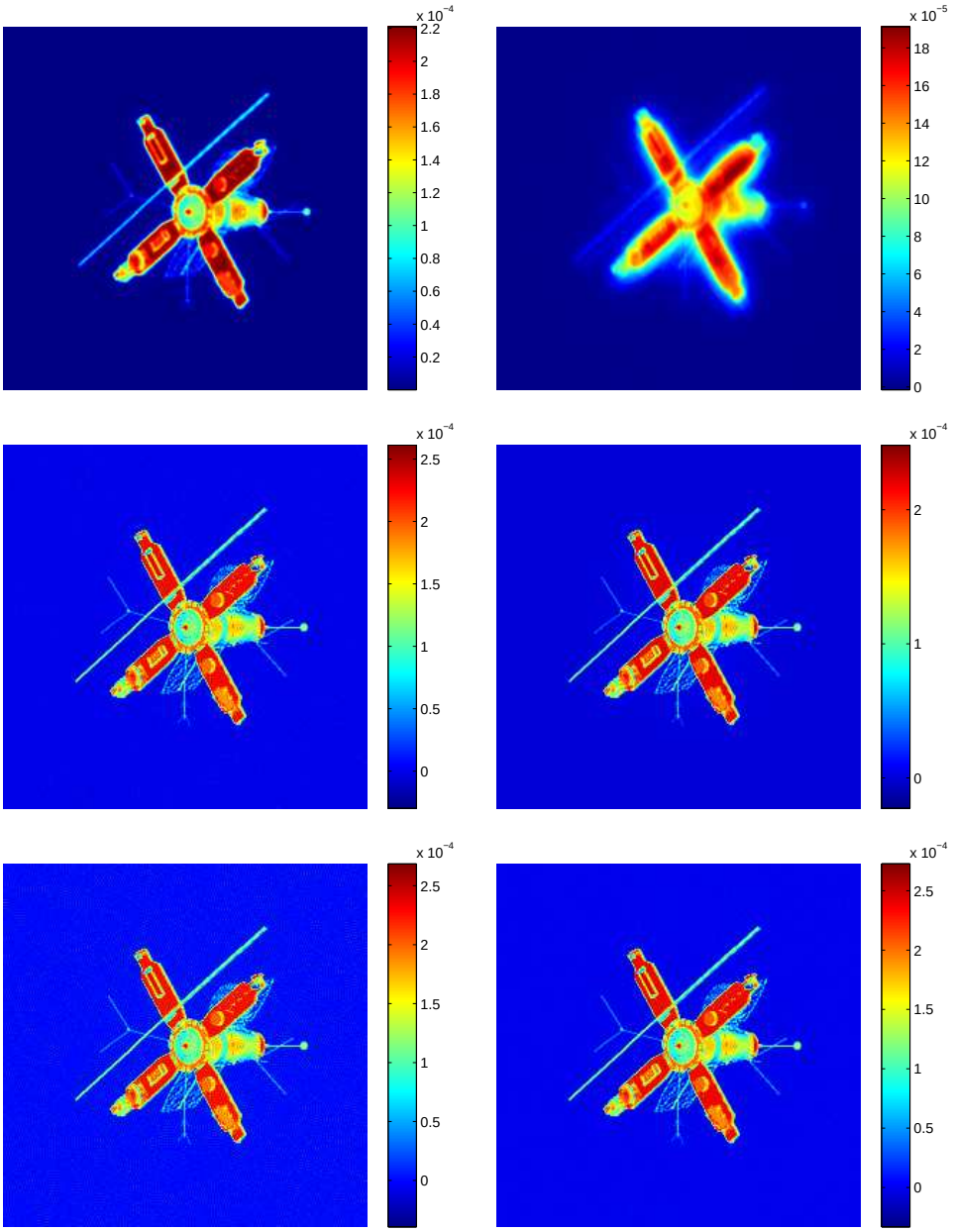

Figure 2.13: Comparison of reconstructed images ($d/r_0 = 5$). Form left to right, the top row shows the diffraction limited image and the first frame blurred image; the middle row shows the reconstructions by FFH model (relative error = 0.1238) and by naïve approach (relative error = 0.1110); the bottom row shows the reconstructions by FFH model with 1% noise wind profile (relative error = 0.1532), and with 5% noise wind profile (relative error = 0.1181).

compared to the diffraction limited image and the blurred image. In this case, there is a clear advantage to using the more accurate PSFs from our FFH approach to reconstruct the image.

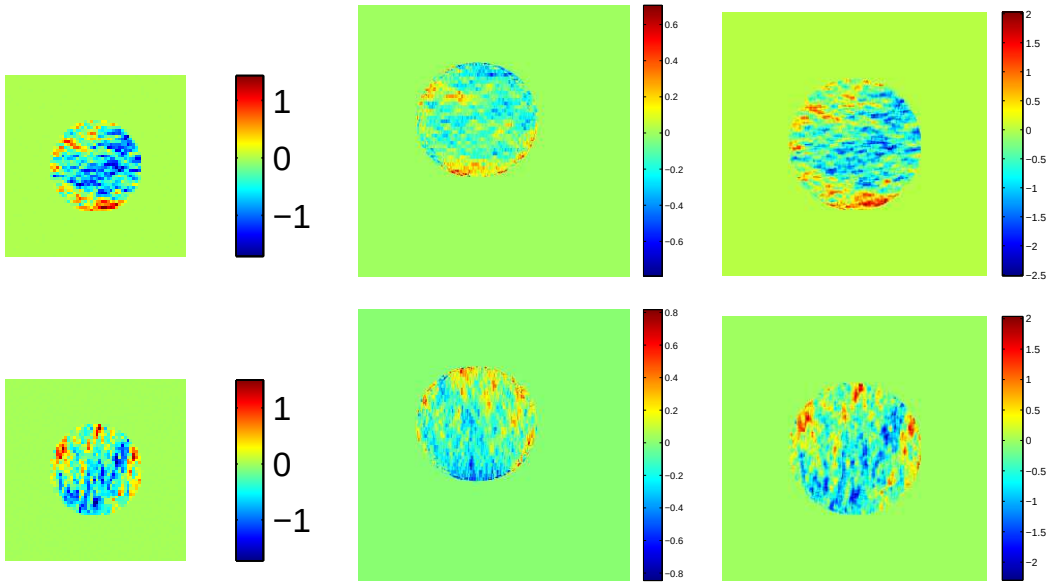

Figure 2.14: The measured and the reconstructed gradients on coarse and fine grids respectively ($d/r_0 = 20$). The top row shows the measured gradients, the reconstructed composite gradients of the 1st layers and the reconstructed gradients of the 1st frame along the horizontal direction. The bottom row shows the measured gradients, the reconstructed composite gradients of the 1st layer and the reconstructed gradients of the 1st frame along the vertical direction.

Again, Figure 2.17 and the bottom row in 2.18 show the results with noisy wind profile. Clearly, with the noisy wind profile, the relative errors of the PSFs are worse. However with 1% noise, the relative errors are smaller than using the naïve approach with the true wind velocity. Comparing the reconstructed images, using FFH with 1% noise in the wind profile, we even obtain an image with better contrast.

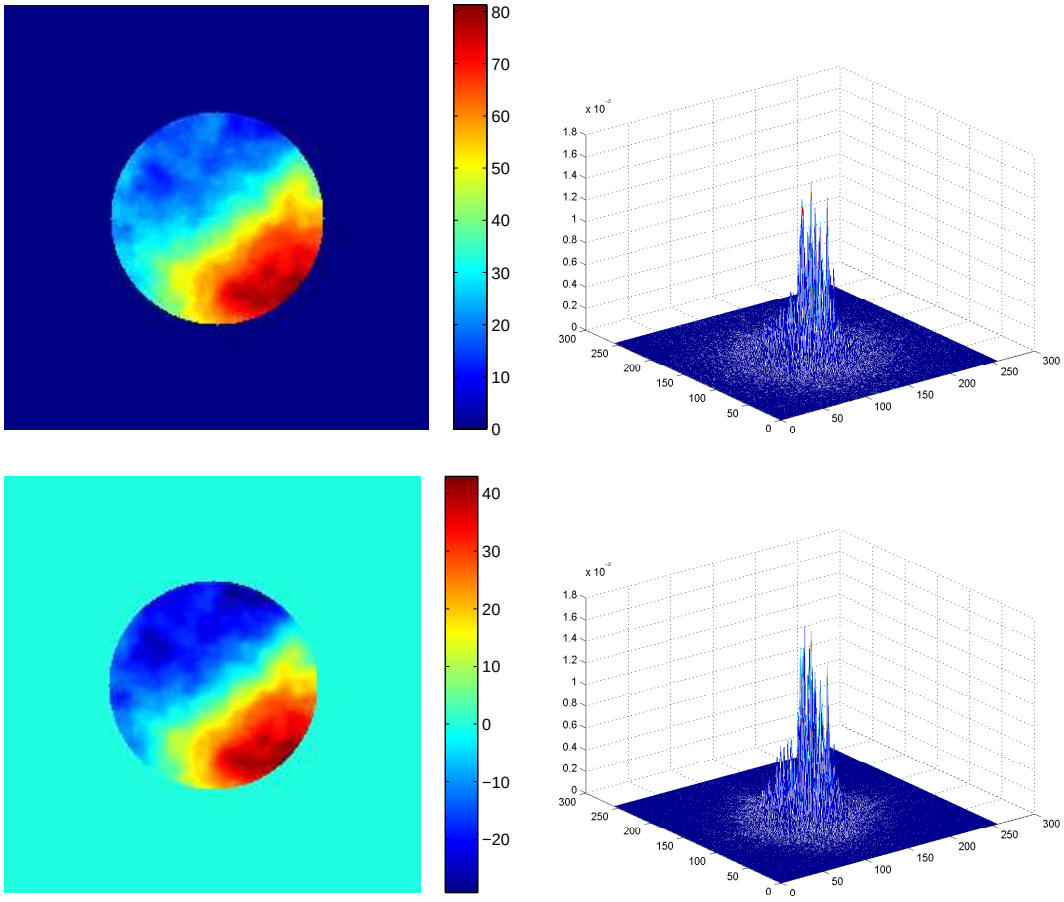

Figure 2.15: The true and the reconstructed phase and PSF of the 1st frame ($d/r_0 = 20$). The top row shows the true wavefront incident at the telescope and the corresponding blurring kernel. The bottom row shows the reconstructed wavefront and the corresponding blurring kernel.

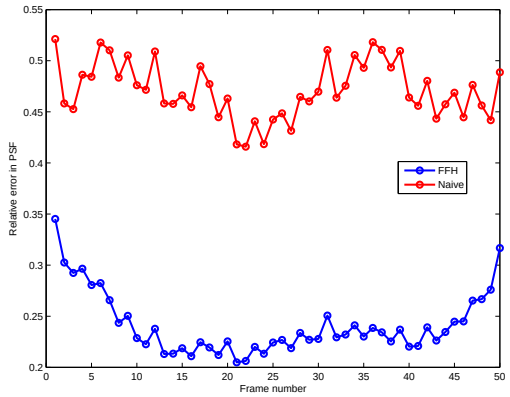

Figure 2.16: Plot of the relative errors of the reconstructed PSFs for each frame using two different approaches, the FFH and naïve approach, when $d/r_0 = 20$.

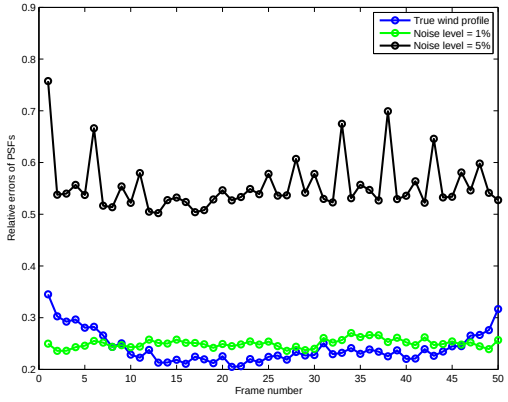

Figure 2.17: Plot of the relative errors of the reconstructed PSFs for different perturbed wind profiles, adding 0%, 1% and 5% noise respectively, by FFH approach when $d/r_0 = 20$.

2.4.4 Experiment 3: Extremely Poor Seeing Conditions

In this final example, we simulate motion of the wavefront for an extreme case $d/r_0 = 45$. In this case, the wavefront is highly oscillatory and the observed images are severely blurred. This is illustrated in Figure 2.20 and 2.23. Figure 2.21 shows the relative errors of the reconstructed PSFs, using our FFH approach and the naïve approach, and the middle row in Figure 2.23 shows the reconstructed image. As can be seen from this example, it is essential to obtain accurate estimates of the PSFs when attempting to reconstruct extremely blurred images. In particular, if we attempt to use gradients measured by a telescope's wavefront sensor to reconstruct a single image frame, the highly oscillatory nature of the wavefront does not provide enough information to expect to get an accurate estimate of the high resolution gradients (and, hence, the corresponding wavefront phase and PSF) by simply inter-

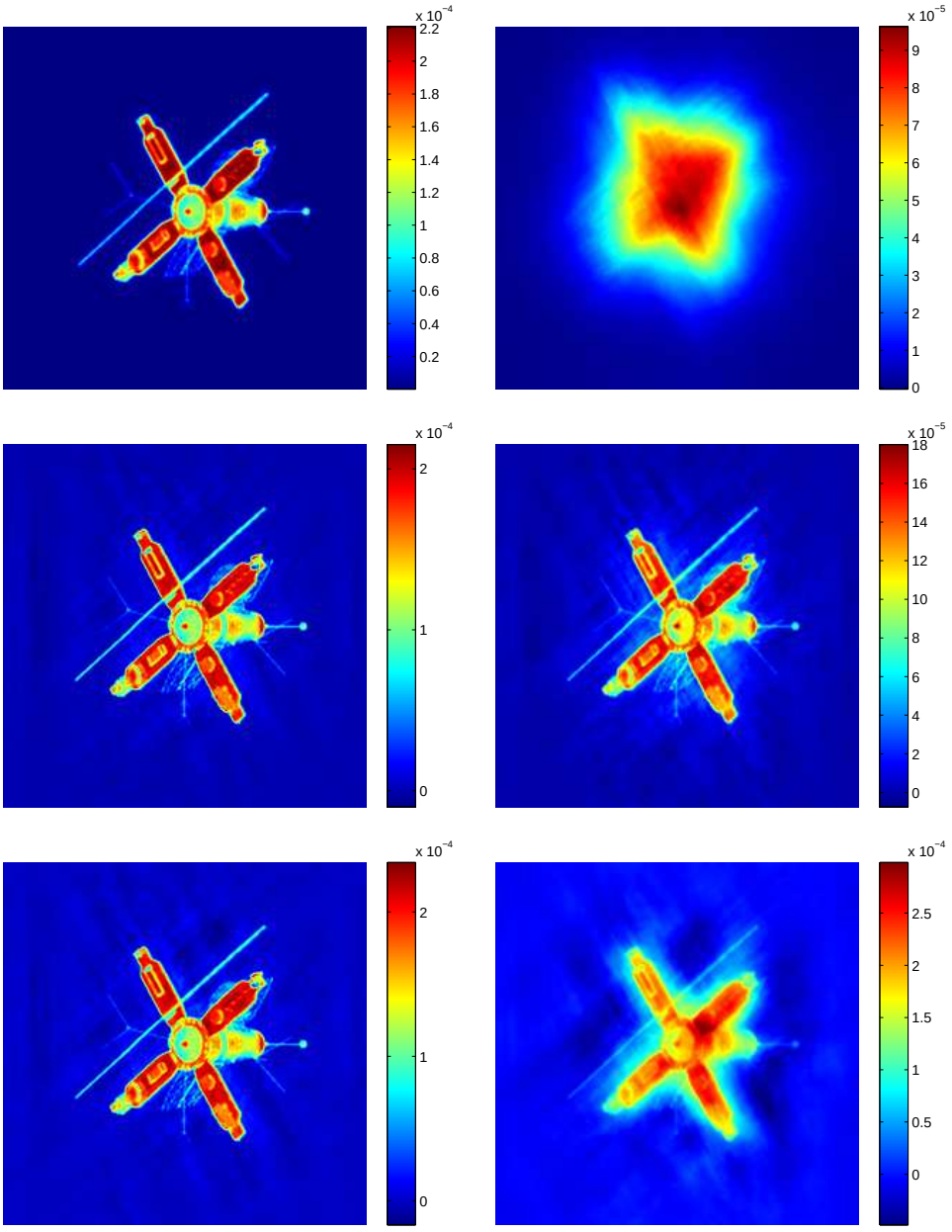

Figure 2.18: Comparison of reconstructed images ($d/r_0 = 20$). Form left to right, the top row shows the diffraction limited image and the first frame blurred image; the middle row shows the reconstructions by FFH model (relative error = 0.2134) and by naïve approach (relative error = 0.3761); the bottom row shows the reconstructions by FFH model with 1% noise wind profile (relative error = 0.1826), and with 5% noise wind profile (relative error = 0.5259).

polating the low resolution measurements to a high resolution grid. It is essential to obtain additional information about the gradients on the high resolution grid, such as we have proposed in this work with the FFH model of the wavefront.

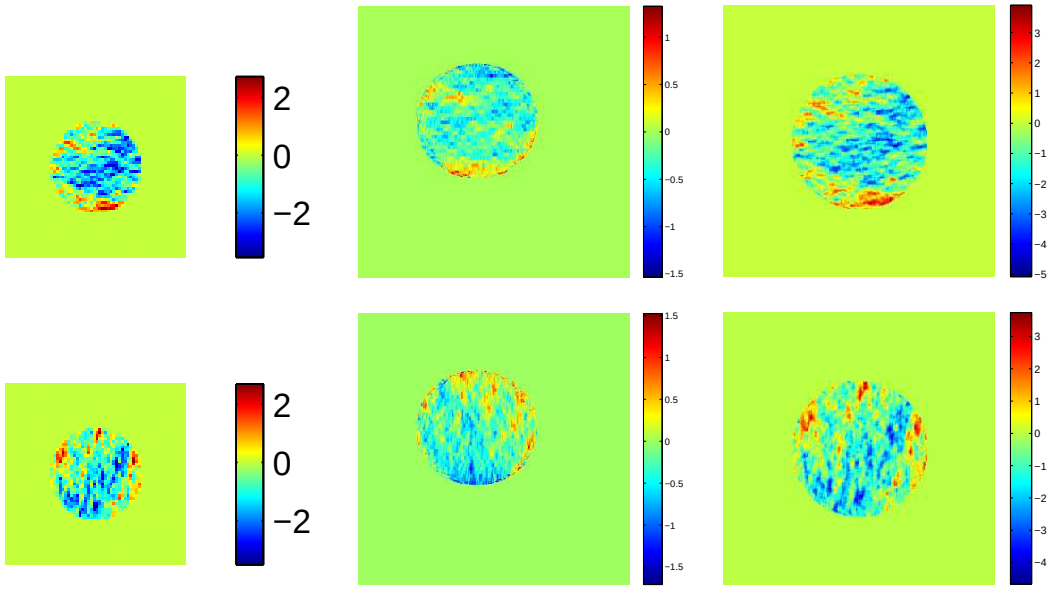

Figure 2.19: The measured and the reconstructed gradients on coarse and fine grids respectively ($d/r_0 = 45$). The top row shows the measured gradients, the reconstructed composite gradients of the 1st layers and the reconstructed gradients of the 1st frame along the horizontal direction. The bottom row shows the measured gradients, the reconstructed composite gradients of the 1st layer and the reconstructed gradients of the 1st frame along the vertical direction.

Figure 2.22 illustrates the relative errors of the computed PSFs. Note that the relative errors using multiple FFH model with 1% noise in the wind profile is much smaller than that using the naïve with the true wind profile. The corresponding reconstructed image is also improved in Figure 2.23.

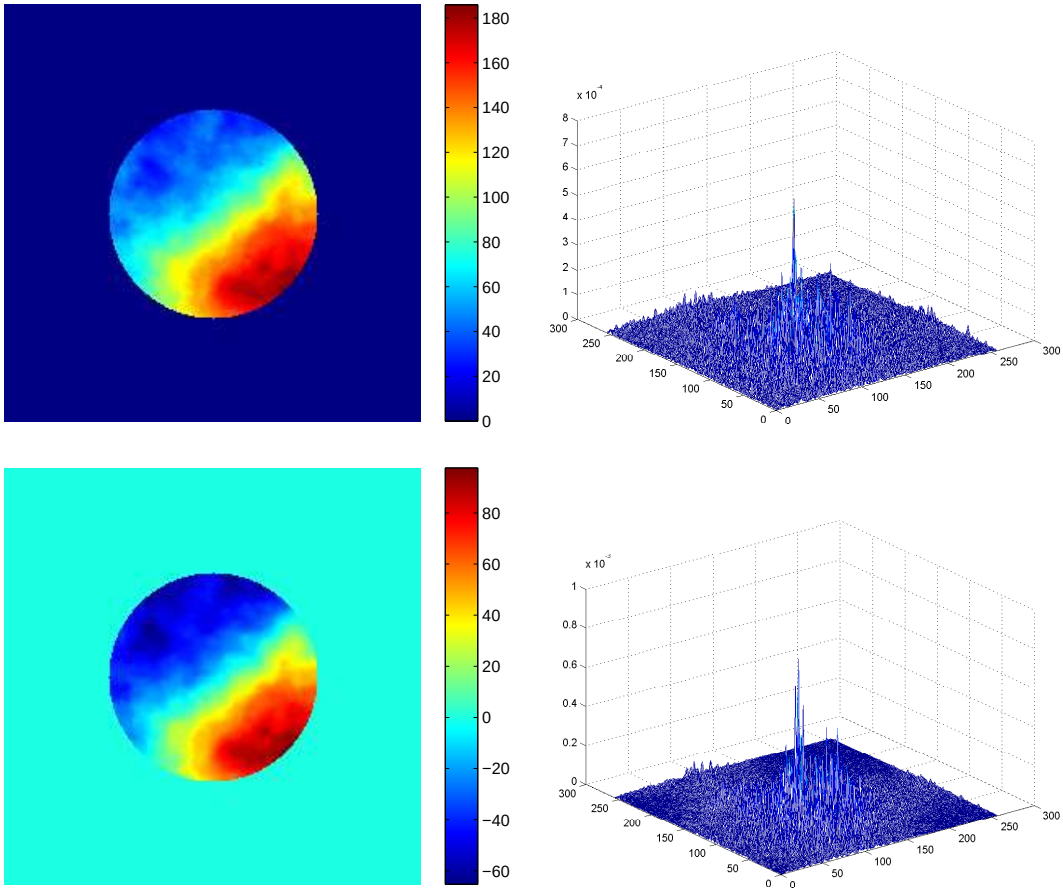

Figure 2.20: The true and the reconstructed phase and PSF of the 1st frame ($d/r_0 = 45$). The top row shows the true wavefront incident at the telescope and the corresponding blurring kernel. The bottom row shows the reconstructed wavefront and the corresponding blurring kernel.

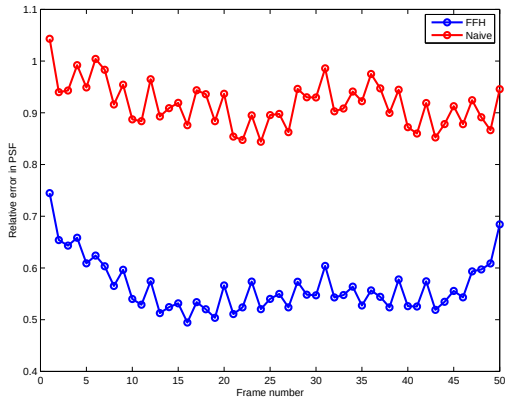

Figure 2.21: Plot of the relative errors of the reconstructed PSFs for each frame using two different approaches, the FFH and naïve approach, when $d/r_0 = 45$.

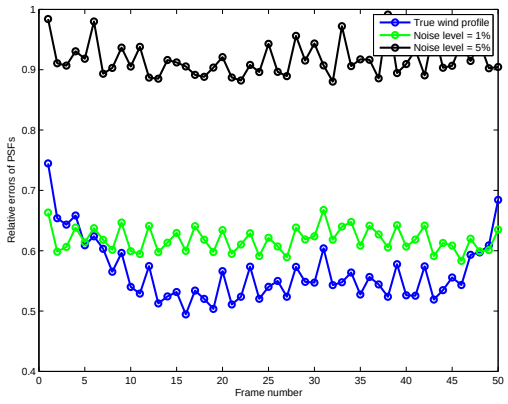

Figure 2.22: Plot of the relative errors of the reconstructed PSFs for different perturbed wind profiles, adding 0%, 1% and 5% noise respectively, by FFH approach when $d/r_0 = 45$.

2.5 Conclusions

The least squares problem (2.8) and the multi-frame deconvolution problem are well understood, and there are many approaches to solve these problems. The new contribution of this work is the FFH gradient reconstruction, which requires solving the least squares problem (2.7). The structure and sparsity of this problem depends on the wind velocity. For example, if the wind velocity results in uniform shifts that are an integer multiple of the (high resolution) pixel size, then there may be some structure (e.g., Toeplitz) that can be exploited when solving this least squares problem. However, for realistic problems, the wind velocity may result in nonuniform shifts, and the shifts are not likely to be integer multiples of the pixel size. Thus, there is no obvious general approach that exploits structure when solving this least squares problem.

However, we do exploit sparsity. In particular, in the experiments reported

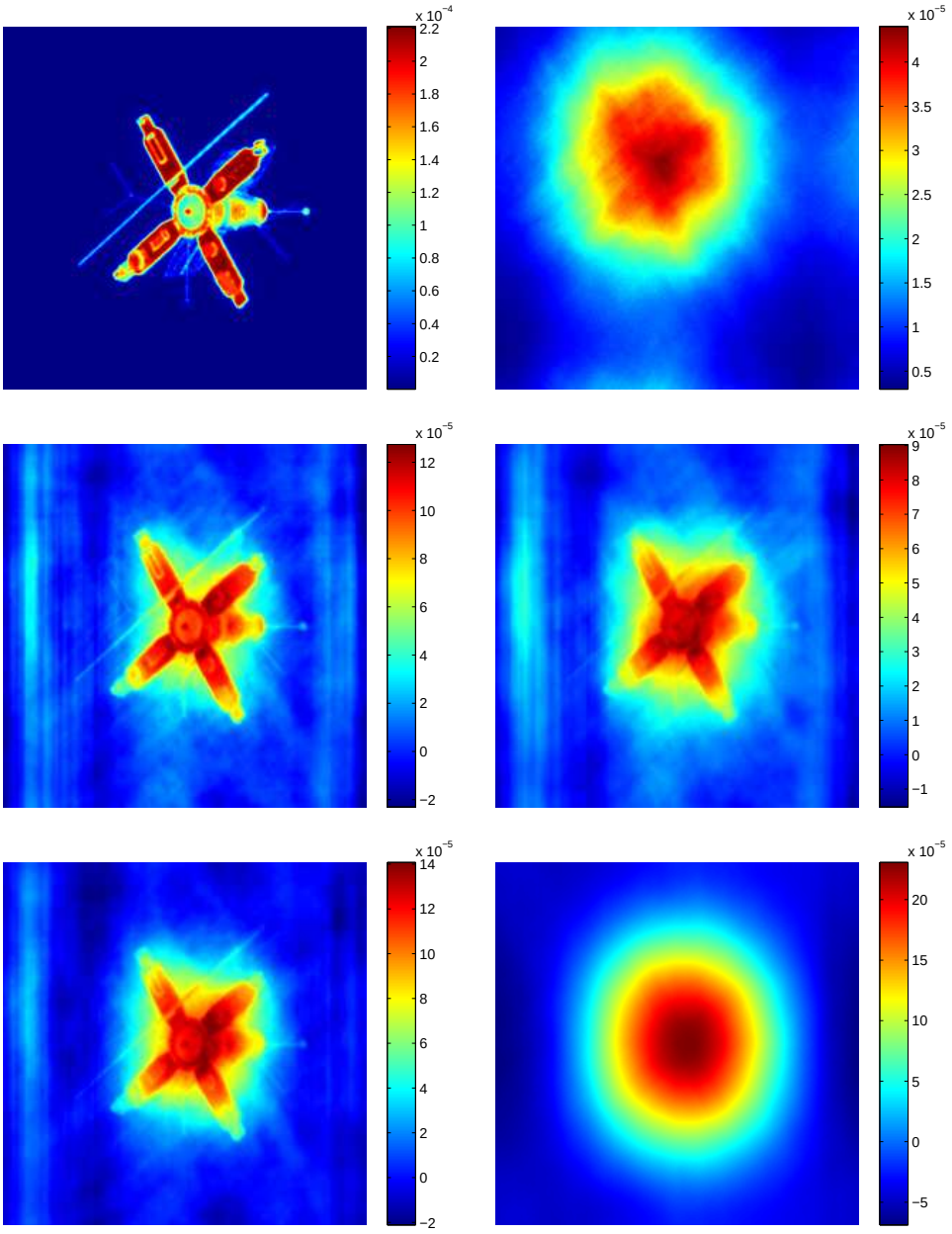

Figure 2.23: Comparison of reconstructed images ($d/r_0 = 45$). Form left to right, the top row shows the diffraction limited image and the first frame blurred image; the middle row shows the reconstructions by FFH model (relative error = 0.6052) and by naïve approach (relative error = 0.7304); the bottom row shows the reconstructions by FFH model with 1% noise wind profile (relative error = 0.6391), and with 5% noise wind profile (relative error = 1.1915).

in this work, the number of non-zeroes in the $204,800 \times 277,248$ coefficient matrix in Equation (2.7) is 2,616,204, or approximately 13 unknowns per row. The number of LSQR iterations needed to solve Equation (2.7) and (2.8) for each of the examples discussed in this section is reported in Table 2.2.

Table 2.2: This table displays the number of LSQR iterations needed to compute $x-$ and y-gradients, as well as the average number of iterations needed for the wavefront reconstruction of each frame.

$\frac{d}{r_0}$	ϕ_x	ϕ_y	ϕ (FFH)	ϕ (naïve)
5	169	174	666	47
20	153	163	663	47
45	154	156	665	48

Note that since the affine transformation depends on the wind velocities, the wind information is crucial for image reconstruction. Our results show that the multiple layer FFH model still improves the image reconstruction even with the inexact wind profile. Especially under the good and not extreme poor seeing conditions, with 5% or less noise, we still could deblur the images. Even under the extreme poor seeing conditions, with 1% noise, we improve the original observed images.

Chapter 3

Parallel Implementation

3.1 Motivation

One important feature of astronomical imaging problems is their large scale. The images themselves can be extremely large partly because large image collectors are built to compensate for the lack of brightness of distant and small space objects; for example, digitized photographic plates at the Space Telescope Science Institute can generate 14,000 by 14,000 16-bit pixel images. Moreover, imaging techniques like deconvolution often involve multiple frames of images: that is, commonly, hundreds or even thousands of frames are included to reconstruct the object. Many software packages have been developed to solve large scale problems: some of them focus on utilizing regularization, such as Regularization Tools [40] and RestoreTools [55]; some of them implement parallel computing, for example, ScaLAPACK [19] and PETSc [4].

In particular, parallel computing is an efficient and feasible way to solve large scale problems. Among a number of parallel software packages developed to solve large scale problems, many utilize iterative methods because of the simplicities and conveniences of iterative implementation. The one we will focus on in parallel is Trilinos described in the next section.

3.2 Overview of Trilinos

The Trilinos project [68, 46, 49] has been developed to assist scalable solver design and provide a good robust solver for application problems in C++. One characteristic of Trilinos is its two level design where package is the fundamental building block which contains the minimal dependencies within itself. Just as the meaning of Trilinos in Greek, "a string of pearls", each individual package is a gem on its own, but when combined, their contributions to the numerical software libraries is worth even more. Since roughly each package can be treated as an individual mathematical library and can be distributed separately, in the next few subsections, we focus on only some of the packages in Trilinos, including Epetra, EpetraExt, Belos and Teuchos.

3.2.1 Epetra

Epetra, along with Tpetra and Jpetra, is one implementation of the Petra Object Model. Specifically, Petra class provides a foundation for the development of all Trilinos solvers. It defines and executes the data structure of matrices, vectors and graphs for abstract interfaces in parallel implementation. Specifically, parallel data redistribution requires the identification of data packets. In Epetra, to facilitate the redistribution and provide a generic analysis capability, an element *global ID* (GID) is associated with each packet of a distributed object according to a map object (i.e., ElementSpace Object). Four map objects are defined for a distributed object (e.g., a matrix or a graph) [45]:

- **RowMap** lists the GIDs that will be managed on each processor. Typically the processor owns part or all of the data associated with the rows indicated in RowMap.
- **ColMap** is the same as RowMap except that it deals with columns.

- **DomainMap** lists the distribution of GIDs associated with vectors in the domain of the matrix; in other words it defines the layout of the domain object (i.e., vector $\boldsymbol{x}$ in $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$). These GIDs must be uniquely associated with a processor.

- **RangeMap** is the same as DomainMap, except that it deals with the range space (i.e., vector $\boldsymbol{y}$ in $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$).

For example, if a matrix is $m \times n$ with a linear distribution on p processors, each processor owns m/p rows of the matrix which are indicated in a one-to-one RowMap object. The same matrix can also be distributed according to a two dimensional partition where each processor owns m/p rows and n/p columns. Note in the latter case, in general neither RowMap nor ColMap is one-to-one. On the other hand, DomainMap and RangeMap must be one-to-one.

In addition, we use two Epetra classes to define sparse matrices and dense vectors for our problem:

- **Epetra_CrsMatrix** is for constructing and using real-valued double-precision sparse compressed row matrices. The data distribution of Epetra_CrsMatrix is described by all four map attributes listed above.

- **Epetra_MultiVector** is for constructing and using real-valued double-precision dense multi-vectors, vectors and matrices in parallel. The dimensions and distribution of the dense multi-vectors are determined by map attributes and the number of vectors.

In summary, data redistribution is simple with map objects, which enables users to concentrate on algorithms design.

3.2.2 EpetraExt

As a compensation of Epetra, EpetraExt supplies a series of extensions to further assist linear algebra operations. With Epetra focusing on its primary functionality, EpetraExt supports transformations, coloring, partitioning and I/O. To reuse the data generated and collected from other resources, what we are interested in EpetraExt is I/O support. Currently, in EpetraExt, several data structure conversions are provided, including

- **MATLABtriplet format** for writing and reading RowMatrix and Operator objects in Epetra.
- **Matrix Market format** for writing and reading BlockMap, Map, MultiVector, Vector, RowMatrix and Operator objects in Epetra.
- **HDF5 format** for writing and reading of BlockMap, Map, MultiVector, Vector, RowMatrix and Operator objects in Epetra.
- **XML-compatible writer and reader** for writing and reading Epetra objects in XML files.

The ones we use for the imaging problem are the first two: MATLABtriplet format conversion and Matrix Market format conversion.

3.2.3 Belos

As other Trilinos packages, Belos is named by a Greek term meaning "arrow". Belos [10], written using "generic" programming techniques, is the next generation of iterative solvers. By using TSF (a collection of abstract classes that provides an interface to perform solver operations), Belos package has no explicit dependence, and can be used with any linear algebra library that implements TSF abstract interfaces.

Belos contains a collection of different iterative methods for solving large sparse linear systems and least squares problems. Belos prefers algorithms that solve higher level problems since the library separates the algorithm from the implementations of the underlying linear algebra objects. Specifically, Belos contains a collection of standard Krylov methods such as CG, GMRES and Bi-CGSTAB, and their flexible and block variants.

In order to implement iterative methods effectively and efficiently, several classes are created to complete solvers. Typically an abstract solver manager class is needed to detail the functionality for each particular solver; an iterator class is utilized to specify iterations; and an abstract status test class is used to test convergence to stop iterating. In our project, we apply the LSQR solver in Belos to solve the large scale sparse least squares system.

3.2.4 Teuchos

As more and more work has been integrated in Trilinos as packages, Trilinos provides Teuchos, which is aimed to facilitate collection of the common tools, to leverage across all packages created by different developers. The goal of Teuchos is robustness and portability making dependency on Teuchos not a practical liability. Therefore, Teuchos classes are not required to be adopted by other packages, and only very few packages have essential dependence on it. There are several important functionalities provided by Teuchos classes we need:

- **RefCountPtr** is a reference-counted pointer for managing dynamically allocated memory that is safe under multiple inheritance [9].

- **ParameterList** is a parameter that can be used to tune how a package is used, or can provide information back to the user from a package.

- **Traits** provides detailed information about supported generic data types. Teuchos provides three types of traits: ScalarTraits, OrdinalTraits and PacketTraits.

- **Timers** defines uniform interface to wall-clock timers.

3.3 Detailed Implementation and Results

For the parallel implementation, we mainly focus on the system to reconstruct high resolution gradients by FFH as shown in Equation (2.7). The major task is to efficiently construct the motion matrix and define a corresponding matrix-vector multiplication for the iterative solver. Specifically, we create an Epetra_CrsMatrix pointer for each layer motion matrix $\left[(\boldsymbol{I} \otimes \boldsymbol{RW})\boldsymbol{A}_j\right]$. All pointers are pushed into a vector for later use. When implementing the matrix-vector multiplication for the LSQR iterative solver, we split the Epetra_MultiVector object into pieces for each layer motion matrix, and after multiplication, we reassemble vector pieces back to one vector. These operations utilize two functions defined in the Trilinos Teko package: one2many and many2one. Those two functions redistribute an Epetra_MultiVector object through Epetra map objects. For the low resolution gradient data (the left hand side in Equation (2.7)), we read data generated from MATLABin Matrix Market format, and use the Epetra_MultiVector objects to represent dense vectors.

This parallel implementation is run on a machine with 64 GB of RAM and 16 cores (eight Dual-Core AMD Opteron(tm) Processor 8220 CPUs). We illustrate the performance of this parallel implementation in Table 3.1, 3.2 and 3.3. For each problem, the program ran 10 times, and the average time is recorded. Specifically, Table 3.1 summaries the average time for solving a 50-frame problem, while Table 3.2 and 3.3 correspond to a 100-

frame and a 200-frame problem respectively. The setup time in the tables includes time for constructing matrices and reading in data; and the iterative time is the iterative phase of LSQR. The iteration numbers remain almost constant as the number of processors grows. The timings show relatively good scalability up to 8 processors. When using 16 processors, the setup time increases while the iterative time slightly decreases. This phenomenon may result from the fact that increasing the number of processors will bring in additional communication time among processors. Another potential reason is when increasing the number of processors, virtual memory may be needed because more local data is created and eventually the total amount exceeds the memory size.

Figure 3.1, 3.2 and 3.3 show the speedup of setup and iterative time with respect to the time using 1 processor as a baseline.

Table 3.1: Timing of 50-frame problem.

No. of proc	Setup time (sec)	Iterative time (sec)	Iter No.
1	51.8	374.5	173
2	32.1	209.4	174
4	20.3	126.8	174
8	14.2	77.6	174
16	15.7	72	174

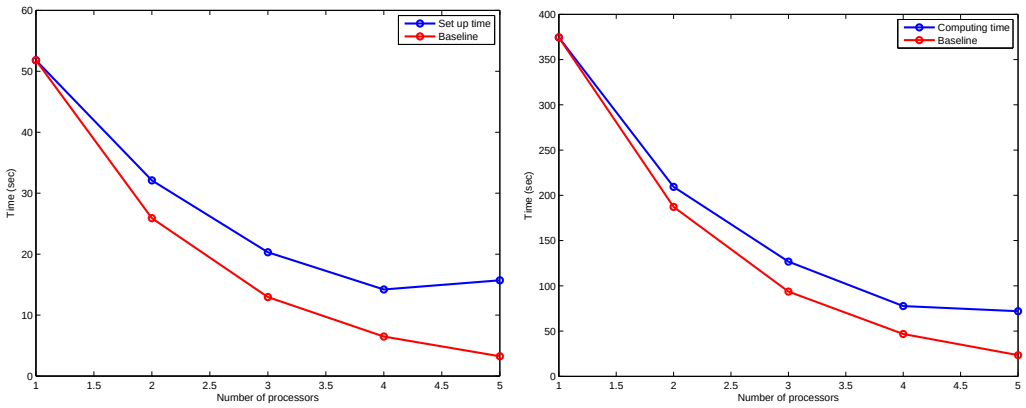

Figure 3.1: Parallel computing time vs. ideal scalability of a 50-frame problem based on one processor. The left figure is the comparison of the setup time; and the right figure is the comparison of the iterative time.

Table 3.2: Timing of 100-frame problem.

No. of proc	Setup time (sec)	Iterative time (sec)	Iter No.
1	131.0	862.2	163
2	80.3	475.7	163
4	50.1	270.7	163
8	34.8	175.1	163
16	39.1	154.7	163

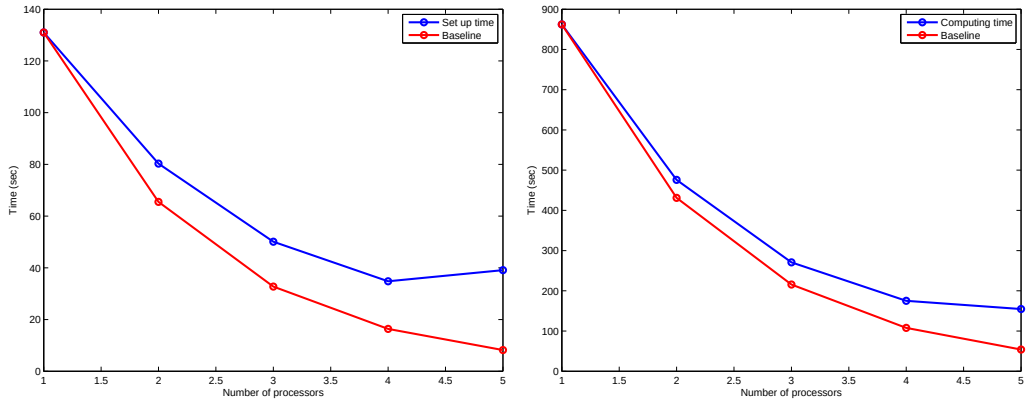

Figure 3.2: Parallel computing time vs. ideal scalability of a 100-frame problem based on one processor. The left figure is the comparison of the setup time; and the right figure is the comparison of the iterative time.

Table 3.3: Timing of 200-frame problem.

No. of proc	Setup time (sec)	Iterative time (sec)	Iter No.
1	388.6	2887	174
2	241.0	1520	175
4	150.4	814.6	174
8	102.8	500.5	176
16	118.8	461	176

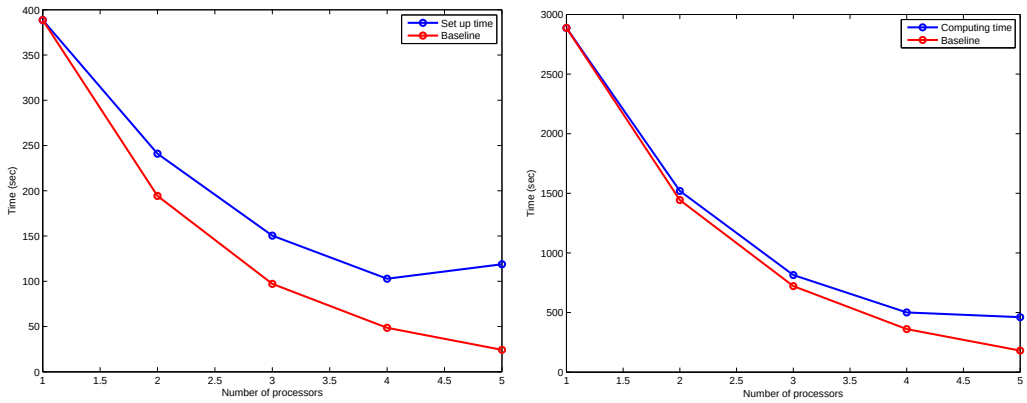

Figure 3.3: Parallel computing time vs. ideal scalability of a 200-frame problem based on one processor. The left figure is the comparison of the setup time; and the right figure is the comparison of the iterative time.

Chapter 4

Preconditioning

4.1 Preconditioning LS systems

A general linear least squares system takes the form:

$$\min_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2,\ \boldsymbol{A} \in \mathcal{R}^{m\times n},\ \boldsymbol{b} \in \mathcal{R}^m,$$

where the rectangular matrix $\boldsymbol{A}$ can be full rank or rank deficient. In the late nineteen sixties, basic numerical methods for solving least squares problems had been developed. For example, Golub in 1965 [32] proposed the QR decomposition using Householder transformations. Since then, great progress, in both direct and iterative methods, has been made to solve least squares problems.

When $\boldsymbol{A}$ is large and sparse, iterative methods can be applied to the normal equation $\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) = 0$. An important class of iterative methods, the Krylov subspace methods, seeks an approximation $\boldsymbol{x}_k$ in step k which minimizes a quadratic error functional such that $\boldsymbol{x}_k \in \boldsymbol{x}^{(0)} + \mathcal{K}_k(\boldsymbol{A}^T\boldsymbol{A}, \boldsymbol{r}^{(0)})$, $\boldsymbol{r}^{(0)} = \boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}x^{(0)})$, where $\mathcal{K}_k$ is the Krylov subspace,

$$\mathcal{K}_k(\boldsymbol{A}^T\boldsymbol{A}, \boldsymbol{r}^{(0)}) = \text{span}\{\boldsymbol{r}^{(0)}, \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{r}^{(0)}, (\boldsymbol{A}^T\boldsymbol{A})^2\boldsymbol{r}^{(0)}, \cdots, (\boldsymbol{A}^T\boldsymbol{A})^{k-1}\boldsymbol{r}^{(0)}\}\,.$$

The implementation can either be based on a CG algorithm or a Lanczos process. Among all iterative methods, the LSQR algorithm (see Algorithm 1), which is based on Lanczos bidiagonalization, and which was developed by Paige and Saunders [58], is one of the most popular approaches to solve least squares systems.

Algorithm 1 LSQR

Let $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ be an initial approximation

Compute $\beta_0 = \|\boldsymbol{b}\|_2$, $\boldsymbol{u}_0 = \boldsymbol{b}/\beta_0$, $\alpha_0 = \|\boldsymbol{A}^T\boldsymbol{u}_0\|_2$, $\boldsymbol{v}_0 = \boldsymbol{A}^T\boldsymbol{u}_0/\alpha_0$

Set $\boldsymbol{w}_0 = \boldsymbol{v}_0$, $\phi_0 = \beta_0$, $\bar{\rho}_0 = \alpha_0$

for $k = 0, 1, \ldots,$ **do**

- $\beta_{k+1} = \|\boldsymbol{A}\boldsymbol{v}_k - \alpha_k\boldsymbol{u}_k\|_2$
- $\boldsymbol{u}_{k+1} = (\boldsymbol{A}\boldsymbol{v}_k - \alpha_k\boldsymbol{u}_k)/\beta_{k+1}$
- $\boldsymbol{z}_k = \boldsymbol{A}^T\boldsymbol{u}_{k+1} - \beta_{k+1}\boldsymbol{v}_k$
- $\alpha_{k+1} = \|\boldsymbol{z}_k\|_2$
- $\boldsymbol{v}_{k+1} = \boldsymbol{z}_k/\alpha_{k+1}$
- $\rho_k = (\bar{\rho}_k^2 + \beta_{k+1}^2)^{1/2}$
- $c_k = \bar{\rho}_k/\rho_k$
- $s_k = \beta_{k+1}/\rho_k$
- $\theta_{k+1} = s_k\alpha_{k+1}$
- $\bar{\rho}_{k+1} = -c_k\alpha_{k+1}$
- $\phi_k = c_k\bar{\phi}_k$
- $\bar{\phi}_{k+1} = s_k\bar{\phi}_k$
- $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + (\phi_k/\rho_k)\boldsymbol{w}_k$
- $\boldsymbol{w}_{k+1} = \boldsymbol{v}_{k+1} - (\theta_{k+1}/\rho_k)\boldsymbol{w}_k$
- Test for convergence

end for

According to [58], LSQR is mathematically equivalent to applying the CG method to the normal equations (for example CGLS, the Conjugate Gradient

Least Squares method), but converges somewhat more quickly when $\boldsymbol{A}$ is ill-conditioned. However, when we need to solve a regularized LS system, such as the one in our FFH reconstruction problem, we may need to solve the system for many values of the regularization parameter. If the system is large scale with a small regularization parameter, the convergence of LSQR is slowed down. Therefore, we may need preconditioning to achieve a reasonable rate of convergence.

Researchers have come up with a lot of different ways to construct a preconditioner $\boldsymbol{P}$ for the CG method. Early ideas for preconditioning the CG method started in the nineteen fifties; see more details in [34]. More recent preconditioning techniques for iterative methods, including the CG method, are discussed in [67, 12]. In particular, we consider right preconditioning

$$\min_{\boldsymbol{y}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{P}^{-1}\boldsymbol{y}\|_2 \,, \;\; \boldsymbol{P}^{-1}\boldsymbol{y} = \boldsymbol{x} \,.$$

Performing LSQR with the right preconditioner, we obtain Algorithm 2.

4.2 General Techniques for Choosing Preconditioners

Although a preconditioner is chosen based on properties of the matrix $\boldsymbol{A}$, there are a few particularly important concerns that should be satisfied:

- $\boldsymbol{A}\boldsymbol{P}^{-1}$ should be close to the identity matrix or at least have a spectrum that is clustered.
- The operation $\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{y}$ should be cheap/easy to perform.

A good preconditioner $\boldsymbol{P}$ typically depends on the problem under consideration. Below, we describe three ways to construct preconditioners.

Algorithm 2 LSQR with right preconditioning

Find a preconditioner matrix $\boldsymbol{P}$

Let $\boldsymbol{y}^{(0)} = \boldsymbol{0}$ be an initial approximation

Compute $\beta_0 = \|\boldsymbol{b}\|_2$, $\boldsymbol{u}_0 = \boldsymbol{b}/\beta_0$, $\boldsymbol{q}_0 = \boldsymbol{A}^T\boldsymbol{u}_0$, $\alpha_0 = \|\boldsymbol{P}^{-T}\boldsymbol{u}_0\|_2$, $\boldsymbol{v}_0 = \boldsymbol{P}^{-T}\boldsymbol{q}_0/\alpha_0$

Set $\boldsymbol{w}_0 = \boldsymbol{v}_0$, $\phi_0 = \beta_0$, $\bar{\rho}_0 = \alpha_0$

for $k = 0, 1, \ldots,$ **do**

- $\boldsymbol{p}_k = \boldsymbol{P}^{-1}\boldsymbol{v}_k$
- $\beta_{k+1} = \|\boldsymbol{A}\boldsymbol{p}_k - \alpha_k\boldsymbol{u}_k\|_2$
- $\boldsymbol{u}_{k+1} = (\boldsymbol{A}\boldsymbol{p}_k - \alpha_k\boldsymbol{u}_k)/\beta_{k+1}$
- $\boldsymbol{q}_{k+1} = \boldsymbol{A}^T\boldsymbol{u}_{k+1}$
- $\boldsymbol{z}_k = \boldsymbol{P}^{-T}\boldsymbol{q}_{k+1} - \beta_{k+1}\boldsymbol{v}_k$
- $\alpha_{k+1} = \|\boldsymbol{z}_k\|_2$
- $\boldsymbol{v}_{k+1} = \boldsymbol{z}_k/\alpha_{k+1}$
- $\rho_k = (\bar{\rho}_k^2 + \beta_{k+1}^2)^{1/2}$
- $c_k = \bar{\rho}_k/\rho_k$
- $s_k = \beta_{k+1}/\rho_k$
- $\theta_{k+1} = s_k\alpha_{k+1}$
- $\bar{\rho}_{k+1} = -c_k\alpha_{k+1}$
- $\phi_k = c_k\bar{\phi}_k$
- $\bar{\phi}_{k+1} = s_k\bar{\phi}_k$
- $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + (\phi_k/\rho_k)\boldsymbol{w}_k$
- $\boldsymbol{w}_{k+1} = \boldsymbol{v}_{k+1} - (\theta_{k+1}/\rho_k)\boldsymbol{w}_k$
- if $|\bar{\phi}_{k+1}|$ is small enough then compute $\boldsymbol{x}_{k+1} = \boldsymbol{P}^{-1}\boldsymbol{y}_{k+1}$
- Test for convergence

end for

4.2.1 Classical Iterative Scheme as Preconditioners

Each linear stationary iterative method induces a preconditioner. To see this, for simplicity, assuming $\boldsymbol{A}$ is square and we consider the classical iterative methods as splitting methods

$$\boldsymbol{A} = \boldsymbol{G} - \boldsymbol{H}\,,$$

where $\boldsymbol{G}$ is nonsingular. Then

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \rightarrow \boldsymbol{G}\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{H}\boldsymbol{x} \rightarrow \boldsymbol{x} = \boldsymbol{G}^{-1}\boldsymbol{b} + \boldsymbol{G}^{-1}\boldsymbol{H}\boldsymbol{x}\,,$$

which has a fixed point form

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{G}^{-1}\boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{G}^{-1}\boldsymbol{b} \\ &= \boldsymbol{M}\boldsymbol{x}_k + \boldsymbol{N}\boldsymbol{b}\,, \text{ where } \boldsymbol{M} = \boldsymbol{G}^{-1}\boldsymbol{H} \text{ and } \boldsymbol{N} = \boldsymbol{G}^{-1}\,. \end{aligned} \tag{4.1}$$

The iteration matrix can be written as

$$\boldsymbol{M} = \boldsymbol{G}^{-1}\boldsymbol{H} = \boldsymbol{N}(\boldsymbol{G} - \boldsymbol{A}) = \boldsymbol{N}(\boldsymbol{N}^{-1} - \boldsymbol{A}) = \boldsymbol{I} - \boldsymbol{N}\boldsymbol{A}\,.$$

Convergent stationary iterations satisfy $\rho(\boldsymbol{M}) < 1$, i.e., the spectrum of $\boldsymbol{M}$ is small, or in other words, $\boldsymbol{N}\boldsymbol{A}$ is close to the identity matrix $\boldsymbol{I}$. Thus, $\boldsymbol{N}$ plays the role of an approximate inverse. Hence, the matrix $\boldsymbol{P} = \boldsymbol{N}^{-1}$ may be used as a preconditioner - it approximates $\boldsymbol{A}$ in some sense.

4.2.2 Incomplete Factorizations

Another very popular class of preconditioners is the incomplete factorization. Such type of preconditioning is also one of the historically earliest examples which made preconditioned CG become widely accepted. The idea was driven from the fact that by direct methods, computing LU decomposition of a sparse matrix $\boldsymbol{A}$ may result in considerable fill-in. In view of the fact that a

preconditioner just needs to be an approximation to $\boldsymbol{A}^{-1}$, one may consider computing an approximate factorization $\tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}} \approx \boldsymbol{A}$, where $\tilde{\boldsymbol{L}}$ and $\tilde{\boldsymbol{U}}$ should also be sparse. Then choosing $\boldsymbol{P} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{U}}$ leads to an efficient evaluation of $\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{y}$, since $\boldsymbol{z} = \tilde{\boldsymbol{L}}^{-1}\boldsymbol{y}$ and $\boldsymbol{x} = \tilde{\boldsymbol{U}}^{-1}\boldsymbol{z}$ can easily be realized by forward and backward substitution.

For *symmetric positive definite* (SPD) matrices $\boldsymbol{A}$, it is common to use the Cholesky factorization $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{A}$ instead of the LU-decomposition. This sparked the development of incomplete Cholesky factorization techniques.

Another type of preconditioner derives from the incomplete QR factorization, see [3, 17, 66]. With rigorous algorithm design, a robust preconditioner can be constructed by incomplete QR factorization.

4.2.3 Approximate inverse

Starting from the observation that the right preconditioner $\boldsymbol{P}$ should satisfy $\boldsymbol{A}\boldsymbol{P}^{-1} \approx \boldsymbol{I}$, approximate inverse preconditioners are directly aiming at a matrix $\boldsymbol{P}^{-1} = \boldsymbol{X}$ such that

$$\|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{I}\|$$

is small in some norm $\|\cdot\|$. The matrix-vector multiplication $\boldsymbol{X}\boldsymbol{y}$ is then taken as the action of the preconditioner. For this to be viable, the matrix $\boldsymbol{X}$ needs to be sparse. One possible way of generating $\boldsymbol{X}$ is to prescribe its sparsity pattern and then try to construct $\boldsymbol{X}$ as the minimizer of

$$\min\{\|\boldsymbol{A}\boldsymbol{X} - \boldsymbol{I}\|_F : \boldsymbol{X} \in \mathcal{R}^{n\times m}, \text{ with prescribed sparseity pattern}\}$$

where $\|\cdot\|_F$ is the Frobenius matrix norm. More information on approximate inverse preconditioning techniques can be found in [14, 15, 13, 16]

4.3 SSOR Preconditioner

In this section, we construct a preconditioner for the wavefront reconstruction problem by *symmetric successive over relaxation* (SSOR) iteration. Consider normal equations of a regularized least squares system

$$\min_{\boldsymbol{x}} \left(\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \alpha^2 \|\boldsymbol{x}\|_2^2 \right) . \tag{4.2}$$

In the following we throughout write the matrix of normal equations $\hat{\boldsymbol{b}} = \boldsymbol{A}^T\boldsymbol{b}$ and $\hat{\boldsymbol{A}} = \boldsymbol{A}^T\boldsymbol{A} + \alpha^2\boldsymbol{I} = \boldsymbol{D} + \boldsymbol{L} + \boldsymbol{L}^T$, where $\boldsymbol{D}$, $\boldsymbol{L}$, $\boldsymbol{L}^T$ are the diagonal, the strictly lower, and the strictly upper triangular part of $\boldsymbol{A}^T\boldsymbol{A} + \alpha^2\boldsymbol{I}$, respectively. Note that the diagonal matrix $\boldsymbol{D} > 0$.

Starting from the forward Gauss-Seidel iteration, we choose the splitting where $\boldsymbol{G} = \boldsymbol{D} + \boldsymbol{L}$ and $\boldsymbol{H} = -\boldsymbol{L}^T$. By Equation (4.1), we have

$$\boldsymbol{x}_{k+1} = (\boldsymbol{D} + \boldsymbol{L})^{-1}(-\boldsymbol{L}^T)\boldsymbol{x}_k + (\boldsymbol{D} + \boldsymbol{L})^{-1}\hat{\boldsymbol{b}} ,$$

where $\boldsymbol{N} = \boldsymbol{D} + \boldsymbol{L}$ can be used as a preconditioner. Since $\boldsymbol{D}$ is invertible, the matrix $\boldsymbol{D} + \boldsymbol{L}$ is lower triangular, so the equation $\boldsymbol{x} = (\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{y}$ is easy to solve by backsubstitution. Similarly, we could define the backward Gauss-Seidel iteration by splitting $\boldsymbol{G} = (\boldsymbol{D} + \boldsymbol{L}^T)$ and $\boldsymbol{H} = -\boldsymbol{L}$, then

$$\boldsymbol{x}_{k+1} = (\boldsymbol{D} + \boldsymbol{L}^T)^{-1}(-\boldsymbol{L})\boldsymbol{x}_k + (\boldsymbol{D} + \boldsymbol{L}^T)^{-1}\hat{\boldsymbol{b}} .$$

The so-called *successive over relaxation* (SOR) method is obtained by introducing a relaxation factor ω in the component formulation of the Gauss-Seidel method. For example based on forward Gauss-Seidel, define $\boldsymbol{G} = (\frac{1}{\omega}\boldsymbol{D} + \boldsymbol{L})$ and $\boldsymbol{H} = -((1 - \frac{1}{\omega})\boldsymbol{D} + \boldsymbol{L}^T)$, then

$$\boldsymbol{x}_{k+1} = (\boldsymbol{D} + \omega\boldsymbol{L})^{-1} \left[(1 - \omega)\boldsymbol{D} - \omega\boldsymbol{L}^T \right] \boldsymbol{x}_k + \omega(\boldsymbol{D} + \omega\boldsymbol{L})^{-1}\hat{\boldsymbol{b}} .$$

SSOR is a symmetric iteration, which aims to overcome the disadvantage of the Gauss-Seidel and SOR methods. By applying first a step of SOR based

on the forward Gauss-Seidel method

$$\boldsymbol{x}_{k+\frac{1}{2}} = \boldsymbol{x}_k + \omega(\boldsymbol{D} + \omega\boldsymbol{L})^{-1}(\hat{\boldsymbol{b}} - \hat{\boldsymbol{A}}\boldsymbol{x}_k)\,,$$

and then an SOR step based on the backward Gauss-Seidel method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k+\frac{1}{2}} + \omega(\boldsymbol{D} + \omega\boldsymbol{L}^T)^{-1}(\hat{\boldsymbol{b}} - \hat{\boldsymbol{A}}\boldsymbol{x}_{k+\frac{1}{2}})\,,$$

the iteration matrix of SSOR is symmetric if the original matrix is symmetric.

In particular, $\hat{\boldsymbol{A}} = \boldsymbol{A}^T\boldsymbol{A} + \alpha^2\boldsymbol{I}$ is symmetric, and the associated SSOR iteration matrix is $\boldsymbol{I} - \omega(2-\omega)(\boldsymbol{D}+\omega\boldsymbol{L}^T)^{-1}\boldsymbol{D}(\boldsymbol{D}+\omega\boldsymbol{L})^{-1}\hat{\boldsymbol{A}}$. Then we may define an approximate inverse matrix $\boldsymbol{N} = \omega(2-\omega)(\boldsymbol{D}+\omega\boldsymbol{L}^T)^{-1}\boldsymbol{D}(\boldsymbol{D}+\omega\boldsymbol{L})^{-1}$; or in other words, we now find a matrix

$$\hat{\boldsymbol{P}} = \frac{1}{\omega(2-\omega)}(\boldsymbol{D}+\omega\boldsymbol{L})\boldsymbol{D}^{-1}(\boldsymbol{D}+\omega\boldsymbol{L}^T)\,,$$

such that $(\boldsymbol{A}^T\boldsymbol{A} + \alpha^2\boldsymbol{I})\hat{\boldsymbol{P}}^{-1} \approx \boldsymbol{I}$. Note that we rewrite $\hat{\boldsymbol{P}}$ as

$$\begin{aligned}
\hat{\boldsymbol{P}} &= \frac{1}{\omega(2-\omega)}(\boldsymbol{D}+\omega\boldsymbol{L})\boldsymbol{D}^{-1}(\boldsymbol{D}+\omega\boldsymbol{L}^T) \\
&= \frac{1}{\omega(2-\omega)}(\boldsymbol{D}+\omega\boldsymbol{L})\boldsymbol{D}^{-1/2}\boldsymbol{D}^{-1/2}(\boldsymbol{D}+\omega\boldsymbol{L}^T) \\
&= \sqrt{\frac{1}{\omega(2-\omega)}}\left(\boldsymbol{D}^{-1/2}(\boldsymbol{D}+\omega\boldsymbol{L}^T)\right)^T \sqrt{\frac{1}{\omega(2-\omega)}}\left(\boldsymbol{D}^{-1/2}(\boldsymbol{D}+\omega\boldsymbol{L}^T)\right) \\
&= \boldsymbol{P}^T\boldsymbol{P}\,, \text{ where } \boldsymbol{P} = \sqrt{\frac{1}{\omega(2-\omega)}}\left(\boldsymbol{D}^{-1/2}(\boldsymbol{D}+\omega\boldsymbol{L}^T)\right)\,. \qquad (4.3)
\end{aligned}$$

Equation (4.3) gives a preconditioner $\boldsymbol{P}$ of the original regularized least squares system $\begin{bmatrix}\boldsymbol{A}\\ \alpha\boldsymbol{I}\end{bmatrix}$. We use this $\boldsymbol{P}$ as the right preconditioner for the LSQR algorithm.

4.4 Numerical Results

For the relaxation parameter ω, it is well known that $\omega \in (0, 2)$ satisfies the convergence requirement. According to an early report [44] and also [20], under relaxation is essential in image reconstruction because it provides better stability and convergence rate. In our work, based on numerous experiments, we choose $\omega = 0.08$.

We test this preconditioner under different seeing conditions using MATLAB. First we use parameters listed in Table 2.1 to build the problem. Table 4.1, 4.2 and 4.3 illustrate the vertical and horizontal reconstructed gradients (denoted as "$x-$" and "$y-$"), the number of iterations (represented as "IterNo"), the relative residual (represented as "Res"), the relative PSF errors (represented as "Err"), total LSQR iterative time (denoted as "Total time") and average iterative time per iteration (denoted as "Iter time") of the unpreconditioned and preconditioned systems with different d/r_0; Figure 4.1 provides the reconstructed images without and with a preconditioner; Figure 4.2 demonstrates relative residuals of the unpreconditioned and preconditioned LSQR. Based on our experiments, the number of iterations drops when using the SSOR type of preconditioner while achieving almost the same accuracy of the solution. Especially, the relative residual decays faster with an SSOR preconditioner. However, the total LSQR iterative time indicates that using the SSOR preconditioner results in longer overall computational time to compute reconstructions for this particular wind profile problem.

To show the effectiveness of preconditioning in the sense of computational time, we provide another example utilizing the SSOR preconditioner based on a new set of wind profile in Table 4.4. Table 4.5, 4.6 and 4.7 again summarize the unpreconditioned and preconditioned LSQR with different d/r_0; Figure 4.3 and Figure 4.4 provide the reconstructed images and the relative residuals of both unpreconditioned and preconditioned systems. Based on

Table 4.1: Numerical results of unpreconditioned and preconditioned systems corresponding to Table 2.1 when $d/r_0 = 5$.

No Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	169	0.0097	0.0245	37.89/0.2242
$y-$	174	0.0097	-	38.34/0.2203

SSOR Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	96	0.0096	0.0310	85.63/0.8920
$y-$	95	0.0096	-	85.68/0.9019

Table 4.2: Numerical results of unpreconditioned and preconditioned systems corresponding to Table 2.1 when $d/r_0 = 20$.

No Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	153	0.0097	0.1930	32.99/0.2156
$y-$	163	0.0097	-	34.86/0.2139

SSOR Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	85	0.0096	0.2129	75.08/0.8833
$y-$	92	0.0096	-	81.56/0.8865

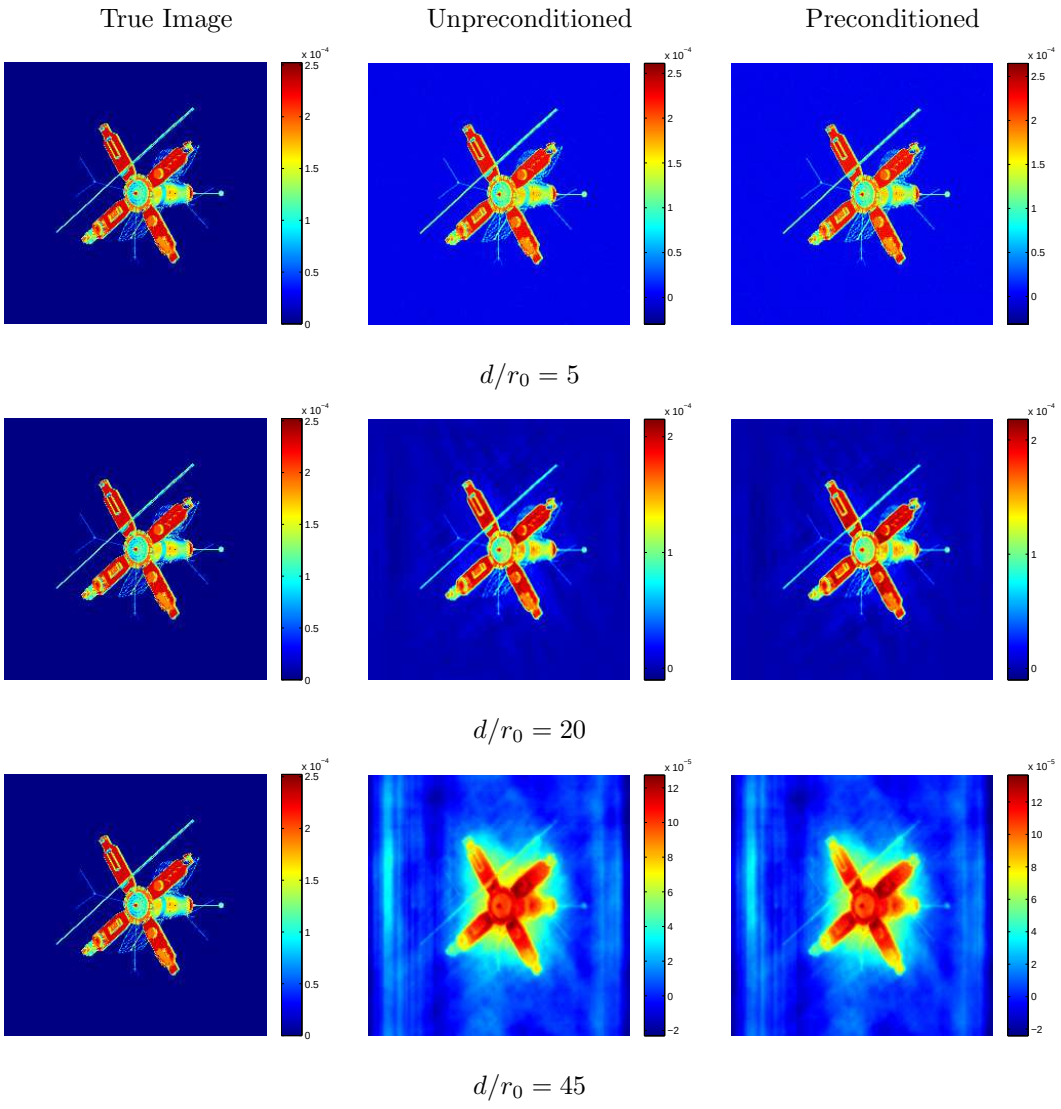

Figure 4.1: The comparisons of reconstructed images from unpreconditioned and preconditioned systems corresponding to Table 2.1. The columns from left to right correspond to the true object, the reconstructed images from the unpreconditioned and the preconditioned systems respectively; the first row shows the results in the case $d/r_0 = 5$, and the second and third rows are in the case $d/r_0 = 20$ and $d/r_0 = 45$.

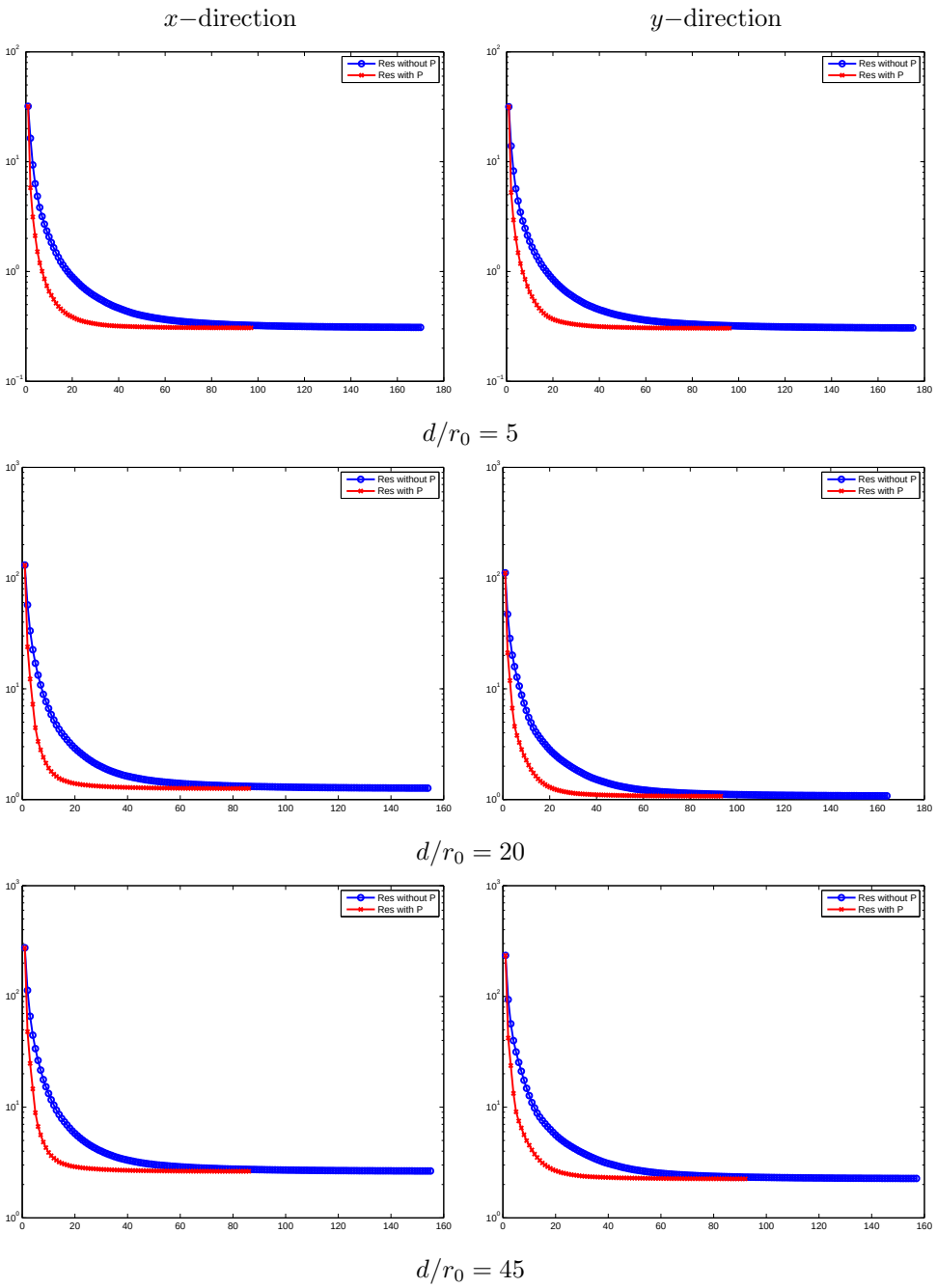

Figure 4.2: Residual vectors of unpreconditioned and preconditioned systems corresponding to Table 2.1. The left column is along the $x-$direction, and the right is $y-$direction. The vertical axis of each plot is log scale residual and the horizontal axis is iteration numbers. Rows correspond to $d/r_0 = 5, 20, 45$ respectively.

Table 4.3: Numerical results of unpreconditioned and preconditioned systems corresponding to Table 2.1 when $d/r_0 = 45$.

No Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	154	0.0097	0.4796	36.75/0.2386
$y-$	156	0.0097	-	37.91/0.2430

SSOR Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	85	0.0096	0.5264	77.49/0.9116
$y-$	91	0.0096	-	83.37/0.9162

the second set of results, both the number of iterations and the total LSQR iterative time are reduced by an SSOR preconditioner with the same solution accuracy. Moreover, comparisons of the relative residual also support that SSOR preconditioners are effective and efficient for this problem.

In summary, SSOR preconditioners reduce the iteration numbers for the image reconstruction problem. However, their efficiency in computational time actually depends on each particular problem, i.e., the wind profile.

Table 4.4: Wind velocities in pixel/frame of different turbulent layers.

layer	location (km)	x-velocity	y-velocity
1	0	0.0865	0.1498
2	11	0.8388	0.4843
3	15	0.3669	0.3669

Table 4.5: Numerical results of unpreconditioned and preconditioned systems corresponding to Table 4.4 when $d/r_0 = 5$.

No Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	846	0.0092	0.0062	236.65/0.2797
$y-$	826	0.0092	-	261.72/0.3169

SSOR Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	138	0.0093	0.0202	154.84/1.1220
$y-$	141	0.0093	-	175.32/1.2434

Table 4.6: Numerical results of unpreconditioned and preconditioned systems corresponding to Table 4.4 when $d/r_0 = 20$.

No Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	877	0.0092	0.0856	269.03/0.3068
$y-$	860	0.0092	-	264.57/0.3076

SSOR Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	138	0.0093	0.1077	161.17/1.1679
$y-$	141	0.0093	-	149.86/1.0628

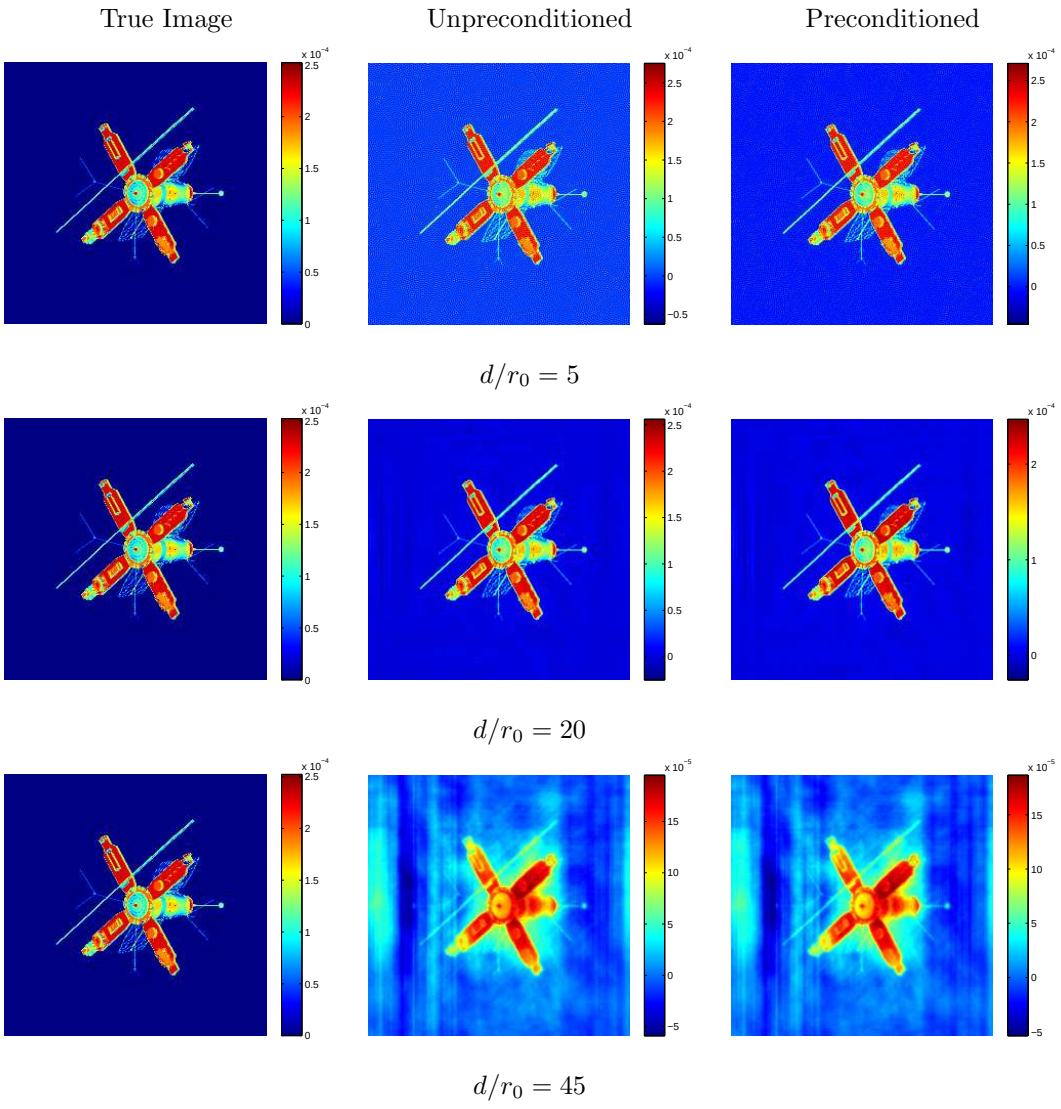

Figure 4.3: The comparisons of reconstructed images from unpreconditioned and preconditioned systems corresponding to Table 4.4. The columns from left to right correspond to the true object, the reconstructed images from the unpreconditioned and the preconditioned systems respectively; the first row shows the results in the case $d/r_0 = 5$, and the second and third rows are in the case $d/r_0 = 20$ and $d/r_0 = 45$.

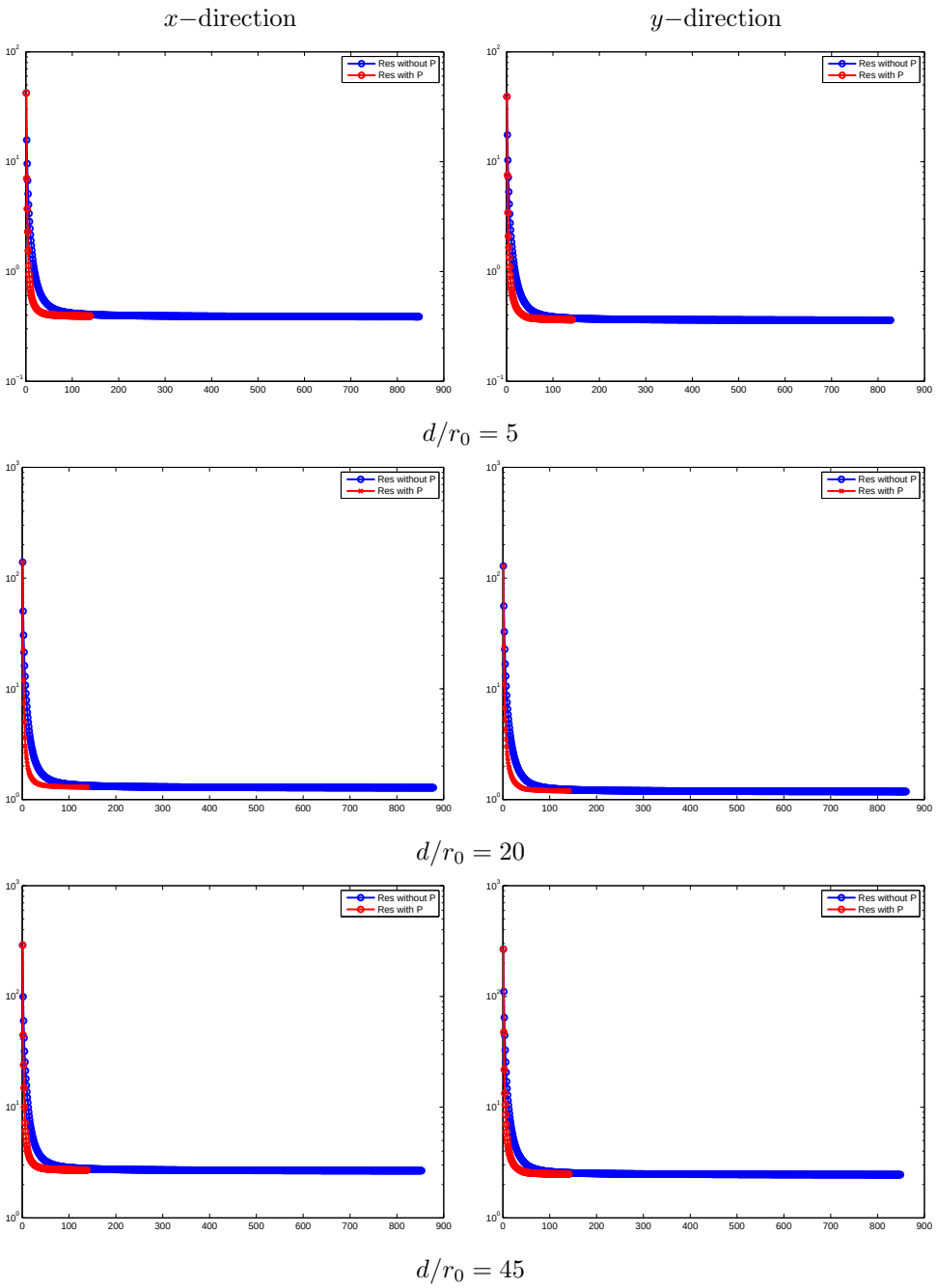

Figure 4.4: Residual vectors of unpreconditioned and preconditioned systems corresponding to Table 4.4. The left column is along the $x-$direction, and the right is $y-$direction. The vertical axis of each plot is log scale residual and the horizontal axis is iteration number. Rows correspond to $d/r_0 = 5, 20, 45$ respectively.

Table 4.7: Numerical results of unpreconditioned and preconditioned systems corresponding to Table 4.4 when $d/r_0 = 45$.

No Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	852	0.0092	0.3395	243.61/0.2859
$y-$	848	0.0092	-	238.14/0.2808

SSOR Preconditioner

Direction	IterNo.	Res	Err	Total time/Iter time
$x-$	137	0.0093	0.3310	162.48/1.1860
$y-$	140	0.0093	-	206.41/1.4744

4.5 Other Preconditioning Techniques

We also considered other preconditioning techniques, such as incomplete factorization and approximate inverse mentioned previously. However, it turns out those types of preconditioners do not efficiently improve the convergence rate or the computational performance. Here we summarize the preconditioning techniques we used for the regularized LS system (4.2):

- **Kronecker product approximation**: The basic idea of utilizing Kronecker product approximation is similar to the incomplete factorization; that is, find an approximate factorization of the linear system, and use the matrix or matrix product from the factorization as a preconditioner. Specifically, according to [54], a Kronecker product approximation of normal equations for (4.2) (with $\boldsymbol{A} \in \mathcal{R}^{(n_{\text{ssp}}m)\times(n_{\text{comp}}L)}$, where n_{ssp} and n_{comp} denote the dimensions of the downsampled and composite images;

m and L represent the number of frames and layers) can be written as:

$$\boldsymbol{A}^T\boldsymbol{A} + \alpha^2\boldsymbol{I} \approx \boldsymbol{K}_1 \otimes \boldsymbol{K}_2 \otimes \boldsymbol{K}_3 + \alpha^2\boldsymbol{I}$$

where $\boldsymbol{K}_1$, $\boldsymbol{K}_2 \in \mathcal{R}^{n_{\text{comp}} \times n_{\text{comp}}}$, $\boldsymbol{K}_2 \in \mathcal{R}^{L \times L}$. We further symmetrize $\boldsymbol{K}_1$, $\boldsymbol{K}_2$ and $\boldsymbol{K}_3$ by

$$\boldsymbol{K}_i = (\boldsymbol{K}_i + \boldsymbol{K}_i^T)/2 \ , \ i = 1\,, 2\,, 3\,,$$

so that an eigen-decomposition can be computed for each $\boldsymbol{K}_i$, and since the size of each matrix is relatively small, the decomposition is not too expensive to be computed. We assume $\boldsymbol{K}_i = \boldsymbol{V}_i\boldsymbol{E}_i\boldsymbol{V}_i^T$, where $\boldsymbol{V}_i$ is an orthogonal matrix and $\boldsymbol{E}_i$ is a diagonal matrix. By the properties of Kronecker product, we may find an approximation:

$$\begin{aligned}
\boldsymbol{A}^T\boldsymbol{A} + \alpha^2\boldsymbol{I} &\approx \boldsymbol{K}_1 \otimes \boldsymbol{K}_2 \otimes \boldsymbol{K}_3 + \alpha^2\boldsymbol{I} \\
&\approx (\boldsymbol{V}_1 \otimes \boldsymbol{V}_2 \otimes \boldsymbol{V}_3)(\boldsymbol{E}_1 \otimes \boldsymbol{E}_2 \otimes \boldsymbol{E}_3)(\boldsymbol{V}_1 \otimes \boldsymbol{V}_2 \otimes \boldsymbol{V}_3)^T + \alpha^2\boldsymbol{I} \\
&= \boldsymbol{V}\boldsymbol{E}\boldsymbol{V}^T + \alpha^2\boldsymbol{I} \\
&= \boldsymbol{V}(\boldsymbol{E} + \alpha^2\boldsymbol{I})\boldsymbol{V}^T \\
&= \boldsymbol{V}\boldsymbol{E}_\alpha^{1/2}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{E}_\alpha^{1/2}\boldsymbol{V}^T\,,
\end{aligned}$$

where $\boldsymbol{V} = \boldsymbol{V}_1 \otimes \boldsymbol{V}_2 \otimes \boldsymbol{V}_3$, $\boldsymbol{E} = \boldsymbol{E}_1 \otimes \boldsymbol{E}_2 \otimes \boldsymbol{E}_3$ and $\boldsymbol{E}_\alpha^{1/2}$ is a diagonal matrix with the square roots of the sums of the diagonal elements and α^2 on its main diagonal. Therefore, we may use $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{E}_\alpha^{1/2}\boldsymbol{V}^T$ as a preconditioner. Unfortunately, although this type of preconditioner has been shown to work well in other image processing applications, it performed poorly for our FFH reconstruction problem.

- **Incomplete Cholesky factorization**: We also implemented a preconditioner using incomplete Cholesky factorization. However, if we choose a preconditioner from a more "exact" factorization, which implies a better convergence rate, the computational time for the factorization increases dramatically, and exceeds the total computational

time without a preconditioner. On the other hand, if we choose a less "exact" factorization, the computational time for factorization decreases, but due to poor convergence rate, the total computational time still overruns that without a preconditioner.

- **Approximate inverse**: According to [25], we construct an approximate inverse type preconditioner for Equation (4.2). However, both the iteration number and the computational time increase dramatically.

Chapter 5

Wavefront Screen Simulation

The efficiency and accuracy of an algorithm should be verified not only in a theoretical way, but also in a practical way by testing. Although testing should always be based on the actual data, good simulations are still needed to test different algorithms in a controlled manner and under a wide variety of conditions, especially in the case of lacking realistic testing data sets. A lot of excellent work has been done to simulate the atmospheric turbulence [52, 50, 64]. This chapter mainly concerns one approach, in [69], of generating simulation of moving wavefronts of the atmospheric turbulence for testing in the previous chapters.

This chapter is organized in the following way: first, we briefly go through some classical models in optics; then, simulate the moving wavefronts for both single layer and multi-layered cases; finally, some simulation results, which are used in the tests of the previous chapters, and open questions are presented.

5.1 Optics Models

The key to simulate the turbulent process is to generate the wavefront. In the simplest situation, the atmosphere above the telescope is treated as a single layer, so that the turbulence moves along a uniform wind velocity. We

call this simple assumption *the single layer case*. For more realistic situation, the atmosphere above the telescope is treated as several layers, each layer moving with its own velocity. We call the second situation the multi-layered case. Next, we discuss data generation of both cases.

First, we introduce the modified von Karman *power spectrum density* (PSD) model. In this model, PSD is defined as

$$\Phi(\kappa) = 0.033 C_n^2 \frac{\exp(-\frac{\kappa^2}{\kappa_m^2})}{(\kappa^2 + \kappa_0^2)^{11/6}}\,, \tag{5.1}$$

where C_n^2 is the structure constant of the index of refraction fluctuations, $\kappa = 2\pi(f_x\hat{\mathbf{i}} + f_y\hat{\mathbf{j}})$ is angular spatial frequency in rad/m (f_x and f_y are angular spatial frequency in cycle/m, and κ^2 is element wise squaring), and $\kappa_m = 5.92/l_0$ and $\kappa_0 = 2\pi/L_0$ (l_0 and L_0 are the inner and outer scale respectively; and the modified von Karman PSD model allows $l_0 = 0$ and $L_0 = \text{inf}$). The modified von Karman model is the simplest PSD model that includes effects of both inner and outer scales.

The Fried parameter r_0 can also be used to represent PSD in place of C_n^2. That is

$$\Phi(\kappa) = 0.49 r_0^{-5/3} \frac{\exp(\frac{-\kappa^2}{\kappa_m^2})}{(\kappa^2 + \kappa_0^2))^{11/6}}\,. \tag{5.2}$$

Consequently, if we use the ordinary frequency in cycle/m other than in rad/m, by the modified von Karman model, we obtain

$$\Phi(f) = 0.023 r_0^{-5/3} \frac{\exp(\frac{-f^2}{f_m^2})}{(f^2 + f_0^2))^{11/6}}\,, \tag{5.3}$$

where $f = (f_x\hat{\mathbf{i}} + f_y\hat{\mathbf{j}})$, $f_m = 5.92/(2\pi l_0)$ and $f_0 = 1/L_0$. For simulation, we use (5.3) to generate the wavefront. The reason we choose r_0 instead of C_n^2 to represent PSD is because the Fried parameter r_0 is closely related to the image quality which is discussed in Section 2.1.

5.1.1 Sinlge Layer Wavefront Screen Generation

Let $\phi(x, y)$ again denote the wavefront, which is Fourier-transformable. Then the phase is represented as the Fourier series

$$\phi(x,y) = \sum_{l=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{l,n} \exp[i2\pi(f_{x_l}\hat{\mathbf{i}} + f_{y_n}\hat{\mathbf{j}})] \,, \tag{5.4}$$

where f_{x_l}, f_{y_n} are the discrete horizontal and vertical directed spatial frequencies, and the $c_{l,n}$ are the Fourier series coefficients [69].

By the central-limit theorem, we assume that the Fourier coefficients $c_{l,n}$ have a Gaussian distribution because of independent random atmospheric fluctuations along the optical path. In general we assume that the real and imaginary parts of Fourier coefficients have zero mean and equal variances, and the cross-covariances are zero. Moreover, to obtain real-valued ϕ, the Fourier coefficients in (5.4) are Hermitian symmetric, i.e. $c_{l,n} = \bar{c}_{-l,-n}$. For the zero frequency term ($l = n = 0$), we assume that $c_{0,0}$ is a real, zero mean Gaussian random variable. Then the inverse Fourier transform of $c_{l,n}$ are all real, and the expectation of the squared wavefront module satisfies

$$E\left[|\phi|^2\right] = E\left[\sum_{l,n=-\infty}^{\infty} |c_{l,n}|^2\right] . \tag{5.5}$$

The total power in the wavefront written using both Parseval's theorem and the definition of power spectral density gives us the relationship between wavefront $\phi(x, y)$ and PSD:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\phi(x,y)|^2 dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Phi(f_x, f_y) df_x df_y \,. \tag{5.6}$$

Therefore, (5.6) is true if we assume

$$\begin{aligned} E\left[|c_{l,n}|^2\right] &= \Phi(f_{x_l}, f_{y_n}) \triangle f_{x_l} \triangle f_{y_n} \\ &= \frac{1}{h_x h_y} \Phi(f_{x_l}, f_{y_n}) \,, \end{aligned} \tag{5.7}$$

where $\triangle f_{x_l} = 1/h_x$ and $\triangle f_{y_n} = 1/h_y$, and h_x, h_y are horizontal and vertical grid sizes respectively.

Next, in order to produce realizations of the Fourier coefficients for wavefronts, we need to generate the PSD Φ by (5.3). Then taking the square root of the variance given in (5.7) produces the random draws of the Fourier series in Equation (5.4). Finally, the inverse Fourier transform is applied to the Fourier coefficients $c_{l,n}$ to generate the wavefront.

5.1.2 Multi-Layered Wavefront Screen Generation

In [65], it is stated that for the multi-layered model, the C_n^2 profile can be split into a finite number of layers. Each layer is characterized by a turbulence strength that is approximately constant within the layer. The notation $C_{n_j}^2$ is used to represent the structure constant for the jth layer. The notations z_j and Δz_j are used to represent the altitude and thickness of the jth layer. The motivation for breaking the turbulence region into layers is the desire to investigate turbulence subregions that have approximately homogeneous statistics. The effect of field propagation through the entire turbulence region (assuming the indices of refractive fluctuations for each layer are statistically independent) is the sum of the effects of that through all subregions. This layering of the turbulence greatly simplifies the calculations to follow.

The values of the weight of the jth layer, $C_{n_j}^2$, and the altitude, z_j, are chosen in such a way that the zeroth through seventh-order moment, m, of the continuous model match the layered model:

$$\int_0^{\Delta z} z^m C_n^2(z) dz = \sum_{j=1}^{L} z_j^m C_{n_j}^2 \Delta z_j \,, \tag{5.8}$$

where $0 \leq m \leq 7$, Δz is the length of the propagation path through the turbulence, and L is the number of wavefront layers being used.

According to [69], since previously we used the Fried parameter r_0 to generate the PSD, we need the following equation for the plane-wave source:

$$r_{0,pw} = \left[0.423k^2 \int_0^{\Delta z} C_n^2(z)dz\right]^{-3/5}, \tag{5.9}$$

where $k = 2\pi/\lambda$ (λ is the wave length) is the wave number. Moreover, we need to define the log-amplitude variance

$$\sigma^2_{\chi,pw} = 0.563k^{7/6}\Delta z^{5/6} \int_0^{\Delta z} C_n^2(z)(1 - \frac{z}{\Delta z})^{5/6}dz\,. \tag{5.10}$$

Then by (5.8), the Fried parameter $r_{0,pw}$ and the log-amplitude variance $\sigma^2_{\chi,pw}$ of the layered model match the parameters of the bulk turbulence being modeled, and

$$\begin{aligned} r_{0,pw} &= (0.423k^2 \sum_{j=1}^{L} C^2_{n_J}\Delta z_j)^{-3/5}\,, \\ \sigma^2_{\chi,pw} &= 0.563k^{7/6}\Delta z^{5/6} \sum_{j=1}^{L} C^2_{n_j} \left(1 - \tfrac{z_j}{\Delta z}\right)^{5/6} \Delta z_j\,. \end{aligned} \tag{5.11}$$

By grouping the equations in (5.11), we can define the Fried parameter of the jth layer by

$$r_{0_j} = \left[0.423k^2 C^2_{n_j}\Delta z_j\right]^{-3/5}. \tag{5.12}$$

That is

$$\begin{aligned} r_{0,pw} &= (\sum_{j=1}^{L} r_{0_j}^{-5/3})^{-3/5}\,, \\ \sigma^2_{\chi,pw} &= 1.33k^{-5/6}\Delta z^{5/6} \sum_{j=1}^{L} r_{0_j}^{-5/3}(1 - \frac{z_j}{\Delta z})^{5/6}\,. \end{aligned} \tag{5.13}$$

The above equations can be written as a linear system

$$\begin{pmatrix} r_{0,pw}^{-5/3} \\ \frac{\sigma^2_{\chi,pw}}{1.33}(\frac{k}{\Delta z})^{5/6} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ (1 - \frac{z_1}{\Delta z})^{5/6} & (1 - \frac{z_2}{\Delta z})^{5/6} & \cdots & (1 - \frac{z_L}{\Delta z})^{5/6} \end{pmatrix} \begin{pmatrix} r_{0_1}^{-5/3} \\ r_{0_2}^{-5/3} \\ \vdots \\ r_{0_L}^{-5/3} \end{pmatrix}. \tag{5.14}$$

From (5.14), we can build up an underdetermined LS problem, which requires finding $r_{0_1}, r_{0_2}, \ldots, r_{0_L}$ to minimize

$$\left\| \begin{pmatrix} r_{0,pw}^{-5/3} \\ \frac{\sigma_{\chi,pw}^2}{1.33}(\frac{k}{\Delta z})^{5/6} \end{pmatrix} - \begin{pmatrix} 1 & 1 & \cdots & 1 \\ (1-\frac{z_1}{\Delta z})^{5/6} & (1-\frac{z_2}{\Delta z})^{5/6} & \cdots & (1-\frac{z_L}{\Delta z})^{5/6} \end{pmatrix} \begin{pmatrix} r_{0_1}^{-5/3} \\ r_{0_2}^{-5/3} \\ \vdots \\ r_{0_L}^{-5/3} \end{pmatrix} \right\|_2^2 . \quad (5.15)$$

Then we can solve for r_{0_j} and generate the wavefront for each layer.

The least squares problem (5.15) can be solved in different ways. Since the scale of the linear system is small, we apply the naïve normal equation approach to solve this least squares problem. The solution is with the smallest 2-norm, and possibly has negative values. In our project, we also consider a constrained optimization approach, for which the solution is forced to be nonnegative. Thus, the solution is more physically realistic.

5.2 Simulation Results

Using the above idea, we generate the single layer wavefront. Parameters are set as in Table 5.1, and the generated wavefront and the associated PSF in Figure 5.1.

Table 5.1: Parameter settings for single layer simulation

Dimensions	Diameter of pupil	Structure constant	Fried parameter	d/r_0
N	d	C_n^2	r_0	d/r_0
800	3.7(m)	2.1901e-18	0.7400	5.0001

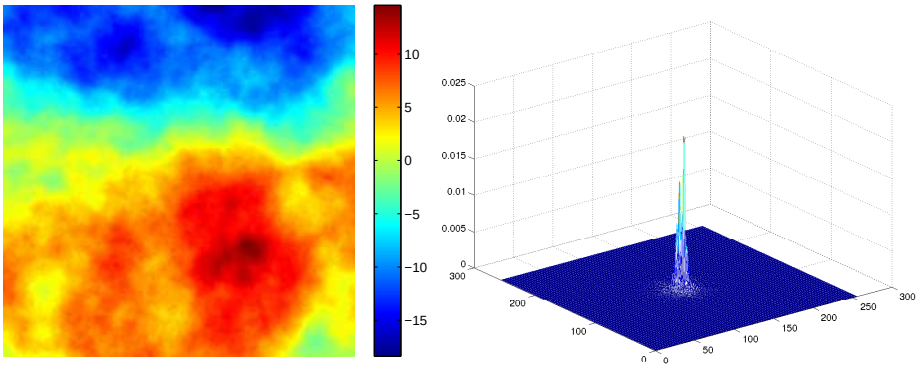

Figure 5.1: Simulated wavefronts and associated PSF in single layer case. On the left is the generated single layer wavefronts; on the right is the generated single layer PSF.

For multi-layered case, solving a LS problem with constraints [69], we could split the single layer into three wavefront screens. The corresponding r_{0_j} for each layer are listed in Table 5.2. The generated wavefront and PSF are in Figure 5.2.

Table 5.2: Multi-layered Fried parameters (by constrained LS)

Layer	1	2	3
r_0	r_{0_1}	r_{0_2}	r_{0_3}
0.7400	50.0000	0.9025	1.5856

We also list r_{0_j} by normal equations in Table 5.3, and the generated wavefront and PSF are in Figure 5.3.

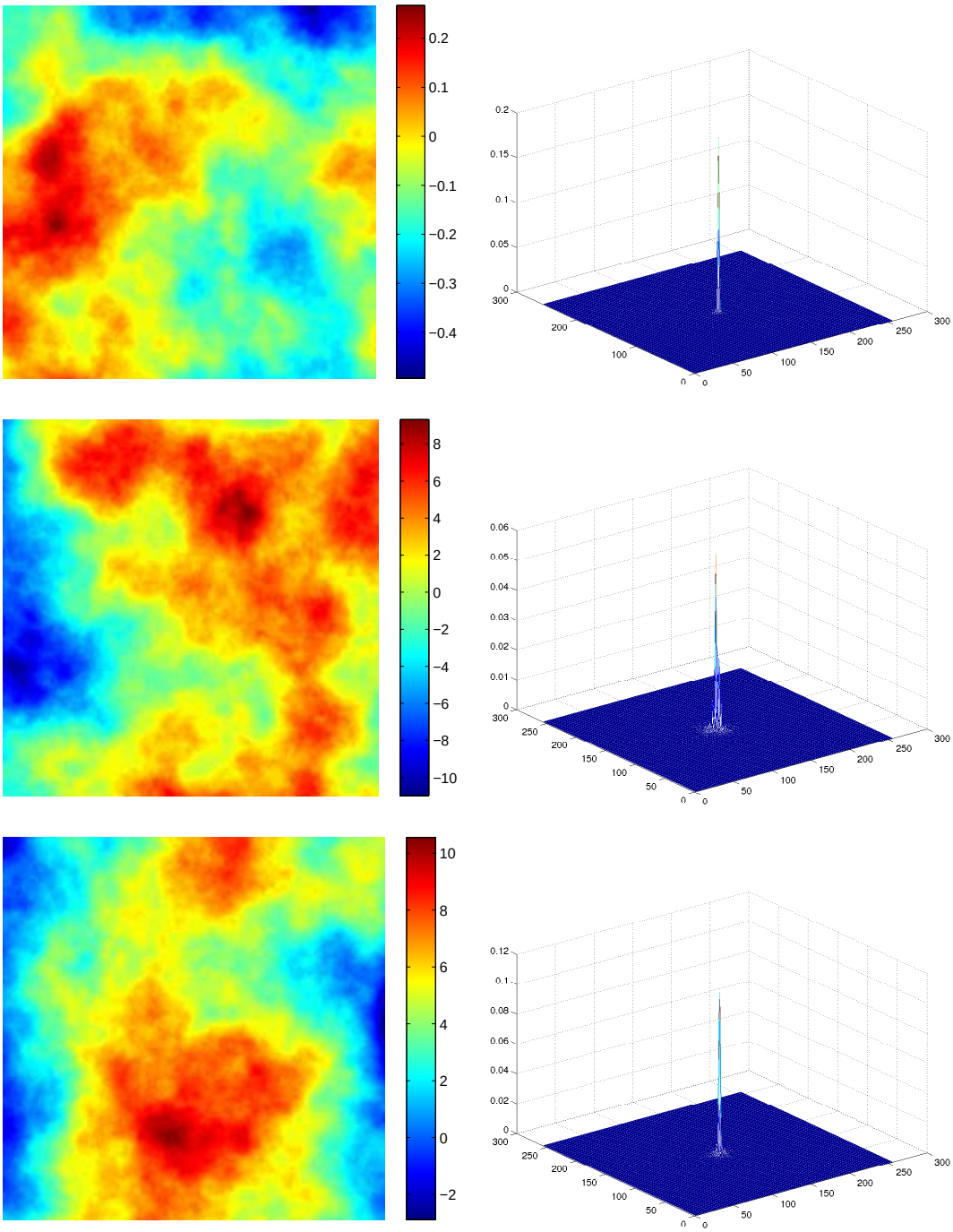

Figure 5.2: Simulated wavefronts and PSFs in multi-layered case. From top to bottom are the 1-st, the 2-nd and the 3-rd layer wavefronts (left) and PSFs (right) by solving a constrained LS problem.

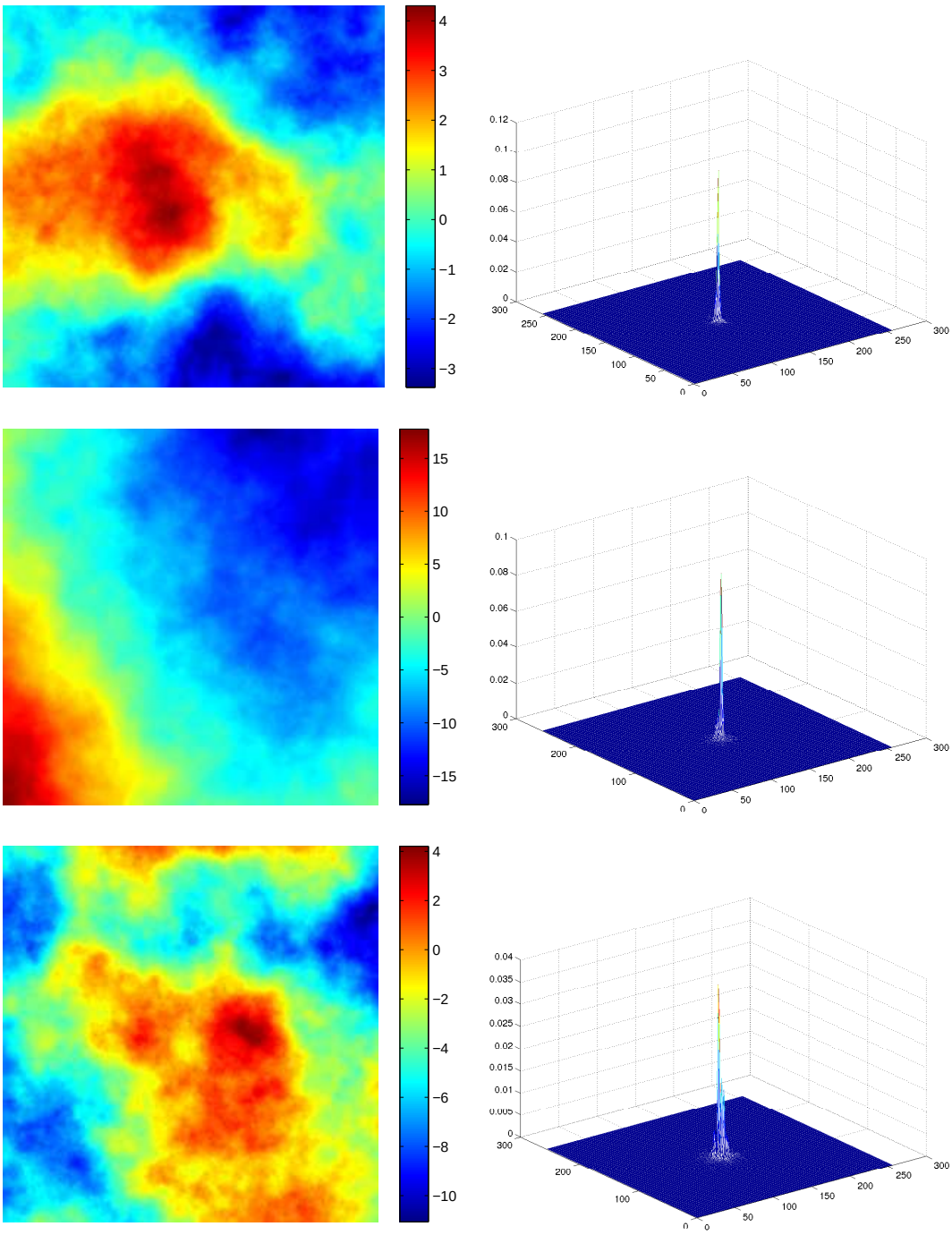

Figure 5.3: Simulated wavefronts and PSFs in multi-layered case. From top to bottom are the 1-st, the 2-nd and the 3-rd layer wavefronts (left) and PSFs (right) by solving normal equations.

Table 5.3: Multi-layered Fried parameters (by normal equations)

Layer	1	2	3
r_0	r_{0_1}	r_{0_2}	r_{0_3}
0.7400	3.6440	1.2558	1.1011

5.3 Remarks and Future Directions

There are still some open questions and remarks we need to list here for future study.

- Large ratio d/r_0 implies more turbulence involved. If we set the structure constant $C_n^2 = 1.7810e-16$, then the corresponding Fried parameter $r_0 = 0.0529$. Again we assume the diameter of the observation aperture $d = 3.7$, then $d/r_0 = 70$. In this case, the generated PSFs are in Figure 5.4

- If we split the generated wavefront in the extreme case $d/r_0 = 70$ into three layers by the constrained minimization approach, then we obtain the PSFs on the left of Figure 5.5. It seems that the split PSFs are much better than we expect or in the single layer case. Possibly, it is because of the constraints.

- If we split the generated wavefront in the extreme case $d/r_0 = 70$ into three layers by normal equation approach, then we obtain the PSFs on right of Figure 5.5, which are more consistent with the single layer generation.

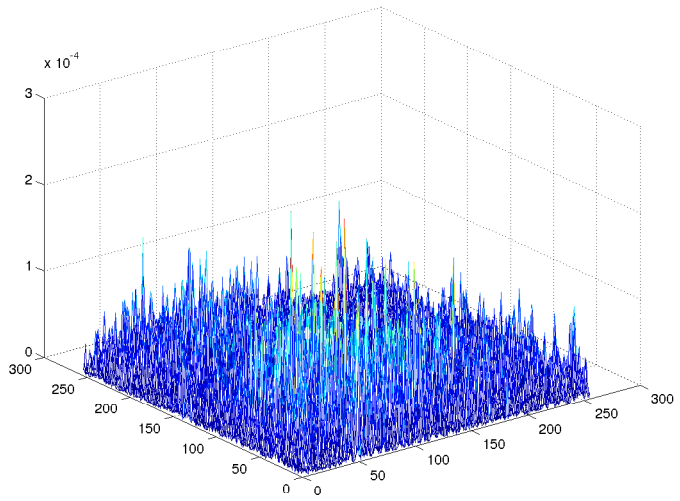

Figure 5.4: PSF in single layer case when $d/r_0 = 70$.

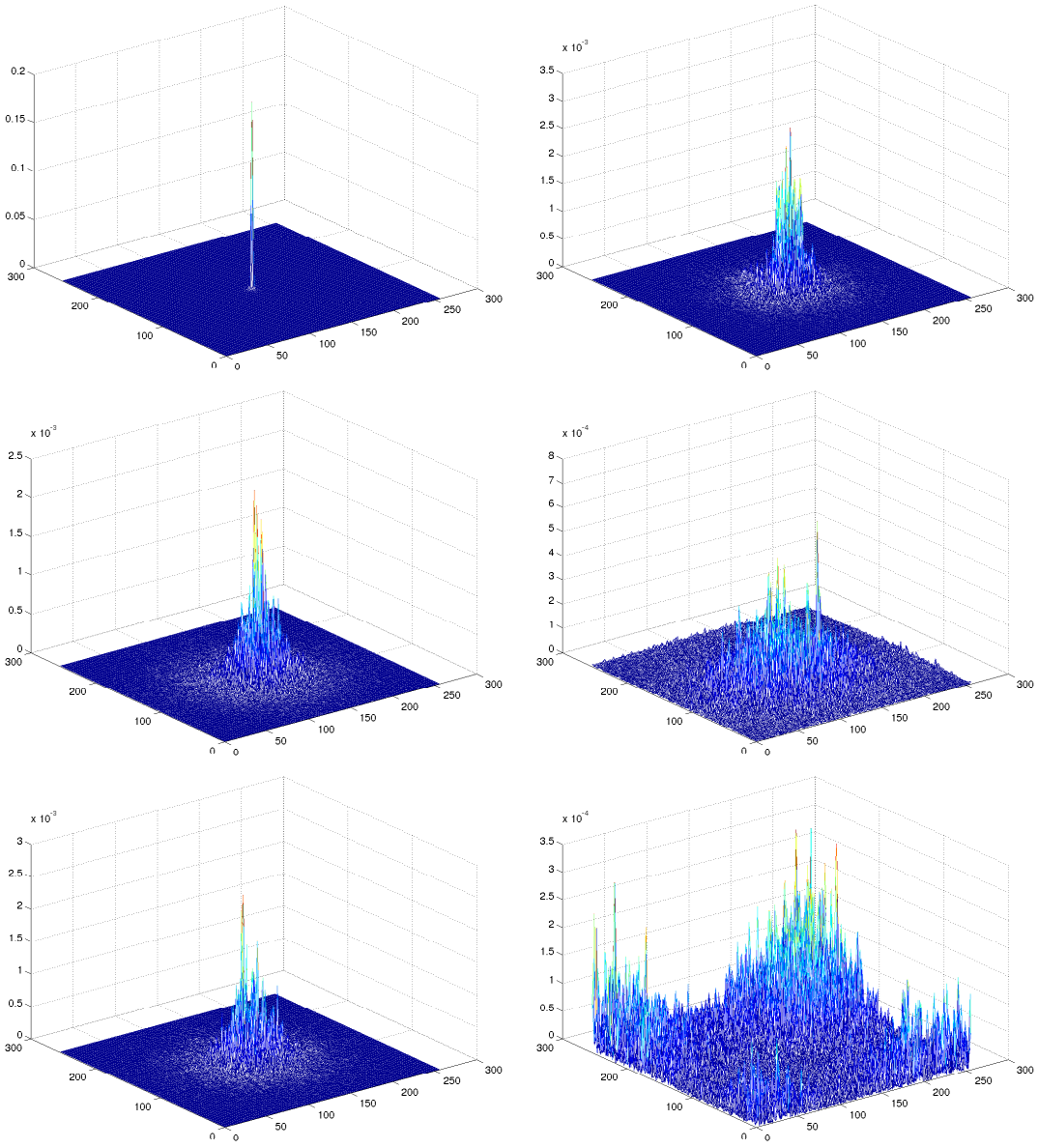

Figure 5.5: PSFs in multi-layered case when $d/r_0 = 70$. The left column are the three layer generated PSFs by solving a constrained LS problem, and the right column are the three layer generated PSFs by solving normal equations.

Chapter 6

Conclusions

6.1 Concluding Remarks

In this thesis, we described how practical deblurring of images distorted by atmospheric turbulence requires solving three large scale least squares problems:

- Reconstructing high resolution wavefront phase gradients from low resolution measurements obtained from a wavefront sensor on the telescope.
- Reconstructing wavefront phases from wavefront phase gradients.
- Reconstructing the image using an atmospheric blurring model that depends on the wavefront phase.

The main contribution of this work is to describe a mathematical model, based on a frozen flow hypothesis, that allows for reconstruction of more accurate high resolution gradients than existing approaches. The frozen flow hypothesis captures the inherent temporal correlations present in wavefronts in consecutive frames of data. Exploiting these correlations can lead to more

accurate PSF estimations, as illustrated by the numerical experiments in Chapter 2. We also showed that the FFH model can be formulated as a sparse least squares problem that can be efficiently solved with iterative methods such as LSQR.

We have also investigated the effectiveness of the FFH model for inexact systems based on noisy wind velocities. Our experiments demonstrate that even with up to 5% error in the wind vectors, the multi-layered FFH model still improves the reconstructed image quality.

A parallel implementation of the essential FFH model has been developed situations when a large number of frames and layers is used in the model (thus resulting in extremely large linear systems). Numerical experiments for parallel programs show good scalability up to 16 processors.

We considered several various preconditioning approaches for LSQR. We found that with the same regularization parameter, SSOR preconditioners reduce the number of iterations. In the case when the iteration number decreases dramatically, the total computational time can also be reduced by this type of preconditioner.

There are several open issues not addressed in this work. For example, we do not have an automatic approach for choosing regularization parameters, and in this regard, hybrid methods (see, for example, [24] and the references therein) might be useful. In addition, it might be worth investigating alternatives to Tikhonov regularization. We also mention that the frozen flow model is only valid for short time periods. For long time periods, it might be necessary to partition the frames into sub-time periods over which the frozen flow is valid. Currently we assume the wind speeds are known to high accuracy. Therefore, another issue worth investigation, if there is uncertainty in the wind speeds, is to reformulate the FFH reconstruction as a separable nonlinear least squares problem, where the wind speeds are considered as unknowns. Finally, because we obtain only an approximation of the wavefront

phases, we only have an approximation of the corresponding PSFs. Further improvement of the PSFs and reconstructed image might be possible using multi-frame blind deconvolution algorithms. In this case, our approach can be used to obtain an initial guess for the PSFs.

6.2 Contributions

This dissertation presents the following contributions we have made:

- We have proposed an efficient algorithm to compute high resolution gradient approximations from low resolution gradient measurements obtained from a wavefront sensor on a telescope.
- We have incorporated the general rigid motion model for the gradient reconstruction problem, which allows both linear and nonlinear motions of the atmospheric turbulence from different air layers.
- We have implemented this algorithm in MATLAB, and provided a parallel implementation utilizing the Trilinos framework, which can be used for extremely large scale realistic problems in astronomical imaging.
- We have analyzed the efficiency of our algorithm under different seeing conditions, and the sensitivity of the solution with respect to the noisy parameters (i.e., wind velocities).
- We have demonstrated that an SSOR preconditioner, with severe underrelaxation, can be effective at reducing the number of iterations and total computational time of the gradient reconstruction problem.

Bibliography

[1] H. C. Andrews and B. R. Hunt. *Digital Image Restoration*. Prentice-Hall, Inc, Upper Saddle River, NJ, US, 1977.

[2] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, United Kingdom, 1996.

[3] Z. Z. Bai, I. S. Duff, and A. J. Wathen. A class of incomplete orthogonal factorization methods. I: Methods and theories. *BIT Numerical Mathematics*, 41(1):53–70, 2001.

[4] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. Petsc Web Page, May 2013.

[5] M. R. Banham and A. K. Katsaggelos. Digital image restoration. *Signal Processing Magazine, IEEE*, 14(2):24–41, 1997.

[6] J. M. Bardsley. An analysis of methods for wavefront reconstruction from gradient measurements in adaptive optics. *Int. J. of Pure and Applied Mathematics*, 42:71–81, 2008.

[7] J. M. Bardsley. Wavefront reconstruction methods for adaptive optics systems on ground-based telescopes. *SIAM J. Matrix Anal.Appl.*, 30(1):67–83, 2008.

[8] J. M. Bardsley, S. Knepper, and J. G. Nagy. Structured linear algebra problems in adaptve optics imaging. *Comp. Math*, 35(2-4):103–117, 2011.

[9] R. A. Bartlett. Teuchos:: RCP beginners guide. Technical Report SAND2004-3268, Sandia National Laboratories, 2004.

[10] E. Bavier, M. Hoemmen, S. Rajamanickam, and H. Thornquist. Amesos2 and Belos: Direct and iterative solvers for large sparse linear systems. *Scientific Programming*, 20(3):241–255, 2012.

[11] J. M. Beckers. Adaptive optics for astronomy: Principles, performance, and applications. *Annu. Rev. Astron. Astrophs.*, 31:13–62, 1993.

[12] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[13] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM J. Sci. Comput.*, 22(4):1318–1332, 2000.

[14] M. Benzi and M. Tůma. Numerical experiments with two approximate inverse preconditioners. *BIT Numerical Mathematics*, 38(2):234–241, 1998.

[15] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2):305–340, 1999.

[16] M. Benzi and M. Tůma. A robust incomplete factorization preconditioner for positive definite matrices. *Numerical Linear Algebra with Applications*, 10(5-6):385–400, 2003.

[17] M. Benzi and M. Tůma. A robust preconditioner with low memory requirements for large sparse least squares problems. *SIAM J. Sci. Comput.*, 25(2):499–512, 2003.

[18] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, PA, US, 1996.

[19] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I.Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK User's Guide*. SIAM, Philadelphia, PA, US, 1997.

[20] J. M. Blackledge. *Digital Image Processing: Mathematical and Computational Methods*. Horwood Publishing, Chichester, West Sussex, UK, 2005.

[21] C. Charalambous, F. K. Ghaddar, and K. Kouris. Two iterative image restoration algorithms with applications to nuclear medicine. *Medical Imaging, IEEE Transactions on*, 11(1):2–8, 1992.

[22] Q. Chu, S. Jefferies, and J. G. Nagy. Iterative wavefront reconstruction for astronomical imaging. Technical Report Math/CS TR-2012-011, Department of Mathematics and Computer Science, Emory University, 2012. To appear *SIAM J. Sci. Comput.*

[23] J. Chung, S. Knepper, and J. G. Nagy. Large-scale inverse problems in imaging. In O. Scherzer, editor, *Handbook of Mathematical Methods in Imaging*, pages 43–86. Springer, 2011.

[24] J. Chung, J. G. Nagy, and D. P. O'Leary. A weighted GCV method for Lanczos hybrid regularization. *Elec. Trans. Numer. Anal.*, 28:149–167, 2008.

[25] X. Cui and K. Hayami. Generalized approximate inverse preconditioners for least squares problems. *Japan Journal of Industrial and Applied Mathematics*, 26(1):1–14, 2009.

[26] T. A. Davis. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. SIAM, Philadelphia, PA, US, 2006.

[27] H. W. Engl and W. Grever. Using the L-curve for determining optimal regularization parameters. *Numerische Mathematik*, 69(1):25–31, 1994.

[28] D. L. Fried. Least-square fitting a wave-front distortion estimate to an array of phase-difference measurements. *J. Opt. Soc. Am*, 67(3), 1977.

[29] E. Gendron and P. Léna. Single layer atmospheric turbulence demonstrated by adaptive optics observations, astrophysics and space science. *Astrophysics and Space Science*, 239(2):221–228, 1996.

[30] E. Gershnik and E. N. Ribak. Light propagation through multilayer atmosperic turbulence. *Optics Communications*, 142(1):99–105, 1997.

[31] L. Gilles. Order-N sparse minimum-variance open-loop reconstructor for extreme adaptive optics. *Optics Letters*, 28(20):1927–1929, 2003.

[32] G. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, 7(3):206–216, 1965.

[33] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, US, 3rd edition, 1996.

[34] G. H. Golub and D. P. O'Leary. Some history of the conjugate gradient and Lanczos algorithms: 1948-1976. *SIAM Review*, 31(1):50–102, 1989.

[35] J. W. Goodman. *Introduction to Fourier Optics*. MaGraw-Hill, Greenwood Village, CO, US, 2nd edition, 1996.

[36] C. W. Groetsch. *The Theory of Tikhonov Regularization for Fredholm Integral Equations of the First Kind*. Pitman Advanced Pub. Program, Boston, MA, US, 1984.

[37] J. Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton University Bulletin*, 13:49–52, 1902.

[38] P. C. Hansen. The discrete Picard condition for discrete ill-posed problems. *BIT Numerical Mathematics*, 30(4):658–672, 1990.

[39] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, Philadelphia, PA, US, 1998.

[40] P. C. Hansen. Regularization tools version 4.0 for MATLAB 7.3. *Numerical Algorithms*, 46(2):189–194, 2007.

[41] P. C. Hansen. *Discrete Inverse Problems: Insight and Algorithms*. SIAM, Philadelphia, PA, US, 2010.

[42] P. C. Hansen, J. G. Nagy, and D. P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering*. SIAM, Philadelphia, PA, US, 2006.

[43] J. W. Hardy. *Adaptive Optics for Astronomical Telescopes*. Oxford University Press, New York, NY, US, 1998.

[44] G. T. Herman, A. Lent, and S. W. Rowland. Art: Mathematics and applications: A report on the mathematical foundations and on the applicability to real data of the algebraic reconstruction techniques. *Journal of Theoretical Biology*, 42(1):1–32, 1973.

[45] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring,

A. Williams, and K. S. Stanley. An overview of the Trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, September 2005.

[46] M. A. Heroux and J. M. Willenbring. Trilinos users guide. Technical Report SAND2003-2952, Sandia National Laboratories, 2003.

[47] R. H. Hudgin. Wave-front reconstruction for compensated imaging. *J. Opt. Soc. Am*, 67(3):375–378, 1977.

[48] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, US, 1989.

[49] Sandia National Laboratories. The Trilinos project, August 2013.

[50] R. G. Lane, A. Glindemann, and J. C. Dainty. Simulation of a Kolmogorov phase screen. *Waves in Random Media*, 2(3):209–224, 1992.

[51] K. J. Lee, N. G. Papadakis, D. C. Barber, I. D. Wilkinson, P. D. Griffiths, and M. N. Paley. A method of generalized projections (MGP) ghost correction algorithm for interleaved epi. *Medical Imaging, IEEE Transactions on*, 23(7):839–848, 2004.

[52] B. L. McGlamery. Computer simulation studies of compensation of turbulence degraded images. In L. Wilson, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 76, page 225, 1976.

[53] J. Nagy, Q. Chu, S. Knepper, and S. Jefferies. Fast psf reconstruction using the frozen flow hypothesis. In *Signal Recovery and Synthesis*. Optical Society of America, 2011.

[54] J. G. Nagy, M. K. Ng, and L. Perrone. Kronecker product approximations for image restoration with reflexive boundary conditions. *SIAM J. Matrix Anal.Appl.*, 25(3):829–841, 2003.

[55] J. G. Nagy, K. Palmer, and L. Perrone. Iterative methods for image deblurring: a MATLAB object-oriented approach. *Numerical Algorithms*, 36(1):73–93, 2004.

[56] J. Núñez. Image reconstruction and restoration in astronomy. *International Journal of Imaging Systems and Technology*, 6(4):295–296, 1995.

[57] H. Nyquist. Certain topics in telegraph transmission theory. *Trans. AIEE*, 47(2):617–644, 1928.

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

[59] D. L. Phillips. A technique for the numerical solution of certain integral equations of the first kind. *Journal of the ACM (JACM)*, 9(1):84–97, 1962.

[60] L. A. Poyneer, D. T. Gavel, and J. M. Brase. Fast wave-front reconstruction in large adaptive optics systems with use of the Fourier transform. *J. Opt. Soc. Am. A.*, 19(10):2100–2111, 2002.

[61] L. A. Poyneer, M. van Dam, and J. P. Véran. Experimental verification of the fozen flow atmospheric turbulence assumption with use of astronomical adaptive optics telemetry. *J. Opt. Soc. Am. A*, 26(4):833–846, 2009.

[62] H. Ren and R. Dekany. Fast wave-front reconstruction by solving the Sylvester equation with the alternating direction implicit method. *Opt. Express*, 12(14):3279–3296, 2004.

[63] F. Roddier. The effect of atmospheric turbulence in optical astronomy. *Progress in Optics*, 19:281–376, 1981.

[64] N. A. Roddier. Atmospheric wavefront simulation using Zernike polynomials. *Optical Engineering*, 29(10):1174–1180, 1990.

[65] M. C. Roggemann and B. M. Welsh. *Imaging through Turbulence*. CRC-Press, Boca Raton, FL, US, 1st edition, 1996.

[66] Y. Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *Journal of Computational and Applied Mathematics*, 24(1):89–105, 1988.

[67] Y. Saad and H. A. van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):1–33, 2000.

[68] M. Sala, M. A. Heroux, and D. M. Day. Trilinos tutorial. Technical Report SAND2004-2189, Sandia National Laboratories, 2004.

[69] J. D. Schmidt. *Numerical Simulation of Optical Wave Propagation with Examples in MATLAB*. SPIE Press, Bellingham, WA, USA, 2010.

[70] M. Schöck and E. J. Spillar. Method for a quantitative investigation of the frozen flow hypothesis. *J. Opt. Soc. Am. A*, 17(9):1650–1658, 2000.

[71] A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, volume 5, pages 1035–1038, 1963.

[72] A. N. Tikhonov, A. V. Goncharsky, V. V. Stepanov, and A. G. Yagola. *Numerical Methods for the Solution of Ill-Posed Problems*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.

[73] R. K. Tyson. *Principles of Adaptive Optics*. Academic Press, San Diego, CA, US, 1997.

[74] C. R. Vogel. Non-convergence of the L-curve regularization parameter selection method. *Inverse Problems*, 12(4):535, 1996.

[75] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, PA, US, 2002.