An Improved Gauge Dual Descent Algorithm for Noisy Phase Retrieval

By

WILLIAM E. WRIGHT

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Zhaojun Bai, Chair

Michael P. Friedlander

Shiqian Ma

Committee in Charge

2019

This dissertation is dedicated to my daughter Julia,
whose love and kindness gives me meaning every day.

Contents

William E. Wright
November 2019
Mathematics

An Improved Gauge Dual Descent Algorithm for Noisy Phase Retrieval

Abstract

Phase retrieval is a common problem in signal processing, with applications in astronomy, x-ray imaging, electron microscopy, and coherent diffraction imaging (CDI). Many models and algorithms exist for phase retrieval, yet few are designed to handle noise without imposing additional assumptions like signal sparsity. One recent algorithm for noisy phase retrieval which requires no underlying assumptions is the Gauge Dual Descent (GDD) algorithm, which iteratively denoises the desired signal.

The GDD algorithm involves a sequence of eigenvalue problems which can be computationally expensive for large-scale signal recovery, requiring many matrix-vector products. Additionally, signal denoising progress tends to stall in the GDD algorithm prior to satisfying optimality-based termination conditions. To address these challenges, this dissertation provides the following contributions. First, we establish empirical termination conditions for the GDD algorithm based on primal and dual variable relative errors which suggest signal denoising progress has stalled. Next, we develop an adaptive strategy for handling the sequence of eigenvalue problems in the GDD algorithm. These contributions lead to the Improved Gauge Dual Descent (IGDD) algorithm. Numerical examples demonstrate that the IGDD algorithm requires $50 - 80\%$ fewer matrix-vector products than the GDD algorithm for a variety of problems with low-oversampling.

Acknowledgments

Writing this dissertation has been a long, difficult, amazing, exhausting, and ultimately rewarding task. I am grateful to have so many people to thank for their support along the way.

To my advisors: Michael Friedlander, thank for inspiring me with your engaging courses and supporting my research in my early graduate school years. My passion for optimization and numerical methods was shaped in no small part by your advising. And to Zhaojun Bai, special thanks for your support, commitment, and time spent over the past three years. When I was a fourth-year graduate student, you offered to advise me throughout the entire dissertation research and writing process. Your guidance helped me see the forest for the trees, keeping my eyes on the overall goals of this dissertation. Along the way, you taught me how to identify important research topics, organize research papers, and produce quality technical writing. I truly could not have done this without you. Thank you.

To my family: Thank you all for believing in me and supporting me throughout my graduate work. To my mother Sharon, my daughter Julia, and her mother Pattie, thank you all for listening to my challenges and offering encouraging words. I could not count the number of phone calls I had with you, discussing the latest challenge or worrying over the direction of my research. Your support meant everything to me.

And to my friends: Thank you all for your support and understanding. To my Davis friends from the *Anderson House*: our evenings spent discussing the day, watching movies, and playing games like *Pandemic Legacy: Season 2* and *Overcooked!* were exactly what I needed to maintain my sanity. To my West Coast friends: thank you for being there whenever I visited you in Paradise, Portland, San Diego, or San Francisco. Leaving the academic bubble of Davis to visit you was always a refreshing and rewarding change of pace. And, for anyone who wants to ask for the $1,000,000^{\text{th}}$ time if my dissertation is done, the answer is "yes".

CHAPTER 1

Introduction

Phase retrieval is the problem of recovering a signal from magnitude-only observations with little or no knowledge of the signal phase. Let $\mathbf{x}$ be a desired signal (or *true signal*) in $\mathbb{R}^n$ or $\mathbb{C}^n$ which has been observed with sensing (or sampling) vectors $a_i \in \mathbb{C}^n$, resulting in squared measurements $|\langle a_i, \mathbf{x}\rangle|^2 = \mathbf{b}_i \in \mathbb{R}$ for $i = 1, \ldots, m$. Also assume the *true observation* vector $\mathbf{b} = [\mathbf{b}_1, \ldots, \mathbf{b}_m]^T \in \mathbb{R}^m$ has been contaminated by possibly nontrivial noise $\eta = [\eta_1, \ldots, \eta_m]^T \in \mathbb{R}^m$, giving the observation vector $b = \mathbf{b} + \eta \in \mathbb{R}^m$. Then we define the *phase retrieval problem* as

$$\begin{aligned} &\text{find} \quad x \\ &\text{s.t.} \quad |\langle a_i, x\rangle|^2 = b_i = \mathbf{b}_i + \eta_i \quad 1 \leq i \leq m, \end{aligned} \tag{1.0.1}$$

where x is an approximation of the desired signal $\mathbf{x}$. Note that if b is not identically zero then the solution $\mathbf{x}$ will not be unique since we may change the sign or phase of $\mathbf{x}$ to generate additional solutions. If $\eta_i = 0$ for all i, then (1.0.1) is the noiseless phase retrieval problem (from [**24**] and [**14**] among many others). In this dissertation however, we are primarily concerned with nontrivial noise and will refer to (1.0.1) with $||\eta||_2 > 0$ as *noisy phase retrieval*. Additionally, we are concerned with noise η which has a Gaussian distribution, as discussed in [**12**] and [**27**].

Each sensing vector $a_i \in \mathbb{C}^n$ is typically the conjugate of the ith row of the $n-$dimensional discrete Fourier transformation (DFT) matrix F [**9**, Chapter 11]. This gives the constraint $|Fx|^2 = b$ (where the square operator is applied element-wise). Often the number of observations $m = nL$ is oversampled by a factor of L to promote uniqueness of a signal solution and convergence of a given algorithm.

The domain of the constraint in (1.0.1) is a high-dimensional torus, and thus phase retrieval is inherently nonconvex. When deciding how to handle a particular phase retrieval problem, this nonconvexity presents a unique challenge in terms of choosing an appropriate algorithm.

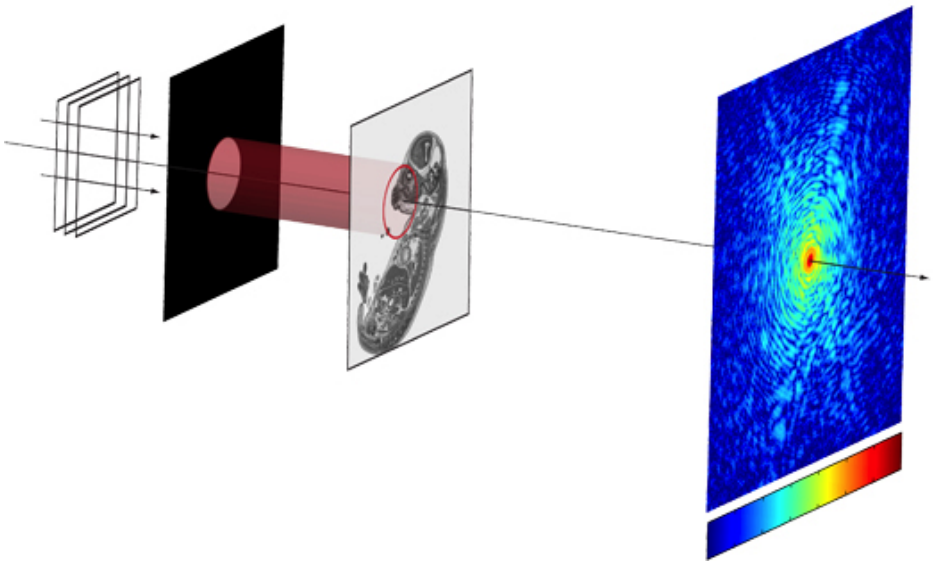

FIGURE 1.1. Depiction of coherent diffractive imaging. Coherent waves (left) are projected at an image (center) which causes a diffraction pattern that is measured by a detector (right). Image from [**34**].

Phase retrieval has a broad range of applications across the sciences, many of which fall into the general category of coherent diffraction imaging (CDI) [**42**]. This section provides a brief overview of CDI and closes with a description of the phase retrieval experimental models used in this dissertation (as well as [**12**], [**14**], and [**27**]). More specific applications of phase retrieval can be found in astronomy [**25**], diffraction and array imaging [**10**] [**17**], microscopy [**43**], optics [**67**], and x-ray crystallography [**35**], [**44**] (for a recent benchmark set of crystallography problems, see [**23**]). For a comprehensive introduction to optical phase retrieval and an overview of recent theory and methods, see the survey [**60**].

CDI is a method for reconstructing 2- or 3-dimensional nano-structures (e.g., nanotubes, nanocrystals, proteins). In this process, highly coherent waves (e.g., x-rays, electrons, photons) are projected at a given object. The resulting diffraction creates a pattern of intensities which are measured with a detector, resulting in magnitude-only measurements. Figure 1.1 below depicts the CDI observation process.

Because CDI does not involve optical lenses, there is no optical aberration (blurring or distorting). Instead, the resolution depends on the limits of diffraction and dose. Many efficient methods

exist for handling low-noise phase retrieval (see Chapter 2 for an overview of some common methods). However, due to the nonconvexity of the constraints in (1.0.1), many low-noise methods lead to algorithms which are likely to diverge or converge to a suboptimal local minimum if there is modest noise or an insufficient number of observations. Thus accurate CDI typically requires minimal noise and multiple observations to recover a high-resolution solution.

When developing an experimental model, there are many methods for increasing the number of observations of a signal. Some options include rotating the position of the object, using a spatial light modulator to defocus the observations, and inserting phase plates, or *masks*, in line with the waves (see the survey [**21**] for a discussion of these methods). Our experimental models use the same masking method described in [**12**, Section 2], [**14**, Sections 4.2, 4.3], and [**27**, Section 5.1] (see Section 2.4 for an overview of these papers). This masking method involves placing a phase plate with a known structure oriented normal to the projected waves. The phase plate can be placed on either side of the object; in our experiments, the plate lies between the object and the detector. The mask is then shifted and multiple observations are collected.

Mathematically, the application of a phase plate to the phase problem (1.0.1) is equivalent to replacing the sensing operator $|Fx|^2$ with $|FC_jx|^2$, where the matrices $C_j \in \mathbb{C}^{n\times n}$ for $j = 1,\dots,L$ are diagonal with standard Gaussian distribution entries $C_j(i,i) \sim \mathcal{N}(0,1)$, representing the diffraction patterns of the shifted phase plate. This gives the observation constraint

$$\left|\begin{matrix} FC_1x \\ \vdots \\ FC_Lx \end{matrix}\right|^2 = b. \tag{1.0.2}$$

In certain cases only a limited number of observations can be collected. For instance, in x-ray imaging, overexposure of the incident waves to the object (living tissue) can be dangerous. Thus the number of observations L may be relatively small compare to the signal size n, again making signal recovery difficult for nonconvex methods.

In this dissertation, our experimental phase retrieval models are generated using the method established in [**14**] and extended in [**27**]. We begin with a true signal $\mathbf{x}$ which is either a vectorized image or a randomly generated vector with coordinates having complex standard Gaussian distribution $\mathbf{x}_i \sim \mathcal{N}(0,1)$. Next, we generate $j = 1, \ldots, L$ diagonal mask matrices $C_j \in \mathbb{C}^{n \times n}$ with diagonal entries $C_j(i,i) \sim \mathcal{N}(0,1)$. We then compute the product (1.0.2) to obtain the true observation vector $\mathbf{b}$. For noiseless phase retrieval, we set the observation vector to $b = \mathbf{b}$. If the phase retrieval problem requires a nonzero noise term η, we add η to the true observation to obtain the observation vector $b = \mathbf{b} + \eta$. Figure 1.2 below depicts the primary test image used in this dissertation. Here, an original image of size 128×128 is used to generate a noisy phase retrieval problem, and we see a few particular iterates returned by the Gauge Dual Descent (GDD) algorithm, the primary algorithm considered in this dissertation (Algorithm 3 in Section 4.2). For a complete explanation of our method for creating noisy phase retrieval problems, see Section 5.2.

FIGURE 1.2. Results from the Gauge Dual Descent algorithm (Algorithm 3 in Section 4.2) applied to a noisy phase retrieval problem (1.0.1) with oversampling rate $L = m/n = 8$ and noise ratio $\epsilon_{\text{rel}} = ||\eta||_2/||b||_2 = 0.30$.

We now summarize the contributions and layout of this dissertation. As we will see in Chapter 2, a wide range of mathematical models and solution methods have been developed for phase retrieval. Yet few models exist for noisy phase retrieval without imposing additional restrictions such as signal sparsity. One recent noisy phase retrieval model which requires no underlying assumptions is the gauge dual of the PhaseLift model (PLGD), which we define in Section 3.2.

The PLGD model was first introduced and analyzed in [**27**], and is based on the PhaseLift model [**12**] in which a desired signal of n elements (e.g., pixels) is lifted into the space of $n \times n$ positive semidefinite matrices, creating a convex, large-scale recovery problem (see Section 2.4 for details). The PLGD model maintains the convergence guarantees of the convex PhaseLift

model, yet also allows for the development of efficient first-order (i.e., gradient-based) methods. To optimize the PLGD model we use the first-order method and the resulting algorithm proposed by [**27**], which we restate in Section 4.2 as the Gauge Dual Descent (GDD) algorithm. For noiseless phase retrieval, the authors of [**27**] demonstrate that the GDD algorithm is far more efficient than a previous algorithm for optimizing the PhaseLift model and returns signals with greater accuracy than the wflow algorithm [**14**] (see Section 2.4 for wflow details).

Yet in the case of noisy phase retrieval, PLGD models with Guassian noise (which we define in Section 5.2) face two significant challenges. Computationally, each evaluation of the PLGD objective function involves a large-scale eigenvalue problem which may require significant computational costs for large signals. Additionally, since PLGD models with Gaussian noise typically do not have a unique solution (see Section 5.3), first-order algorithms such as the GDD algorithm typically fail to converge.

This dissertation offers two main contributions for PLGD models with Gaussian noise. First, we address the convergence challenges of the GDD algorithm and establish new termination conditions which indicate that signal recovery progress has stagnated. Second, we develop a new strategy for handling the sequence of eigenvalue problems in the GDD algorithm. Applying these two modifications to the GDD algorithm decreases the computational costs of the GDD algorithm by $50 - 80\%$ for problems with minimal oversampling.

This dissertation is organized in the following manner. Chapter 2 provides a survey of phase retrieval methods. Chapter 3 presents the gauge duality theory necessary for developing, analyzing, and optimizing the PLGD model. Chapter 4 then presents the Gauge Dual Descent (GDD) algorithm, a first-order algorithm for the PLGD model, and examines the effectiveness of the GDD algorithm for noiseless phase retrieval.

Chapter 5 demonstrates that the GDD algorithm typically fails to converge for PLGD models with Gaussian noise. We then identify the cause of this behavior and establish new termination conditions for the GDD algorithm. Note that Appendix A demonstrates that the GDD algorithm is generally more accurate for PLGD models with Gaussian noise than the wflow algorithm of Section 2.4.

Chapter 6 develops a new, efficient strategy for solving the sequence of eigenvalue problems in the GDD algorithm which we define as the *evolving matrix eigenvalue problem* (EMEP). We first show that the EMEP is the computational bottleneck of the GDD algorithm. We see that the spectrum of these eigenvalue problems evolves in a predictable way, with the algebraically largest eigenvalues clustering for later EMEP iterates. This clustering causes later EMEP iterates to have more difficult eigenvalue problems. Next, we review the *implicitly restarted Arnoldi method* (IRAM), a common large-scale eigenvalue method and develop an efficient, adaptive strategy (Algorithm 8 in Section 6.4) for choosing IRAM parameters to handle the EMEP. We close Chapter 6 by demonstrating that Algorithm 8 effectively tracks the clustering of the algebraically largest eigenvalues from earlier to later EMEP iterates, thus selecting more desirable parameters for the IRAM.

Chapter 7 presents the Improved Gauge Dual Descent (IGDD) algorithm which applies the new termination conditions of Chapter 5 and new eigenvalue strategy (Algorithm 8) of Chapter 6 to the GDD algorithm. Chapter 7 then provides performance results for the IGDD algorithm. We see that the IGDD algorithm reduces computational costs for a variety of noisy phase retrieval problems. Chapter 8 concludes this dissertation with a summary of our contributions and suggestions for future work.

This dissertation uses the following notation. Additional notation and definitions specific to gauge duality are stated in Section 3.2.

The (i,j) entry of a matrix A is denoted $[A]_{i,j}$ or $A(i,j)$, and the i-th component of a vector a is denoted a_i or $[a]_i$. Vector norms are the standard p-norms. Matrix norms for $A \in \mathbb{C}^{m\times n}$ are Schatten p-norms, which apply the p-norm to the vector of singular values, i.e.,

$$||A||_p = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p(A) \right)^{1/p}. \tag{1.0.3}$$

The special case of $p = 2$ gives the Frobenius norm

$$||A||_F = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A) \right)^{1/2}. \tag{1.0.4}$$

Gauge duality is a duality based on multiplicative relations, and thus Schatten norms are essential to gauge duality and developing the PLGD model. In contrast, the EMEP requires measurements with the vector-induced 2-norm. Thus we define the generic *matrix norm* (without a numeric label) as

$$||A|| = \sigma_{\max}(A) = \sup_{||v||_2=1} ||Av||_2. \tag{1.0.5}$$

The standard basis vector is denoted e_i, where $[e_i]_i = 1$ and all other components are zero. Given a vector d in $\mathbb{R}^n$ or $\mathbb{C}^n$ with components $d_1, d_2, \ldots, d_n$, the *diagonal operator* is denoted $\text{Diag}(d)$ and defined as

$$\text{Diag}(d)_{i,j} = \text{Diag}(d_1, d_2, \ldots, d_n)_{i,j} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{else.} \end{cases} \tag{1.0.6}$$

Additionally, if A is a matrix in $\mathbb{R}^{n\times n}$ or $\mathbb{C}^{n\times n}$ then the *diagonal operator* is defined as

$$\text{diag}(A) = \begin{bmatrix} A(1,1) \\ A(2,2) \\ \vdots \\ A(n,n) \end{bmatrix}. \tag{1.0.7}$$

Given $\mathcal{S}$, a subset of a finite-dimensional Euclidean space $\mathcal{X}$, the *indicator* function of $\mathcal{S}$ is defined as

$$\delta_{\mathcal{S}}(x) = \begin{cases} 0 & x \in \mathcal{S} \\ +\infty & x \notin \mathcal{S}. \end{cases} \tag{1.0.8}$$

It is easily seen that if $\mathcal{S}$ is convex, then $\delta_{\mathcal{S}}$ will be convex. The indicator function is useful for tasks like embedding a domain constraint of an optimization model into the objective function and is used frequently in Chapter 3 for proving gauge duality results.

If $\mathcal{C}$ is a convex subset of a finite-dimensional Euclidean space, then the *normal cone* of $\mathcal{C}$ at $y_0 \in \mathcal{C}$ is defined as

$$N_{\mathcal{C}}(y_0) = \{g \in \mathcal{X} \mid \langle g, y - y_0\rangle \leq 0 \ \ \forall y \in \mathcal{C}\}. \tag{1.0.9}$$

By convention, if y_0 is not in $\mathcal{C}$, then $N_{\mathcal{C}}(y_0)$ is the empty set.

Given a subspace S of $\mathbb{R}^n$ or $\mathbb{C}^n$, the *orthogonal complement* of S is defined as

$$S^{\perp} = \{v \mid \langle v, w \rangle = 0 \text{ for all } w \in S\}. \tag{1.0.10}$$

Given a symmetric (or Hermitian) matrix A in $\mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$), its eigenvalues are ordered

$$\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A), \tag{1.0.11}$$

where $\lambda_1(A)$ or simply λ_1 is the algebraically largest eigenvalue of A, and $\lambda_n(A)$ or λ_n is the smallest eigenvalue. The *spectrum* of A is the set of all of its eigenvalues, $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. If S is a subspace of $\mathbb{R}^n$ (or $\mathbb{C}^n$) then (θ, u) is a *Ritz pair for A with respect to S* if

$$\langle w, (Au - \theta u) \rangle = 0 \qquad \forall w \in S. \tag{1.0.12}$$

Likewise, θ is a *Ritz value* and u the corresponding *Ritz vector for A with respect to S*.

For a pair of matrices $A, B \in \mathbb{C}^{n \times n}$, their inner product is induced by the trace

$$\langle A, B \rangle := \operatorname{tr}(A^* B) = \sum_{i=1}^{n} \sigma_i(A^* B). \tag{1.0.13}$$

Given $\mathcal{C}$, a convex subset of a finite-dimensional Euclidean space $\mathcal{X}$, we define projection of $x \in \mathcal{X}$ onto $\mathcal{C}$ as $\Pi_{\mathcal{C}}(x)$.

Given a linear operator $\mathcal{A} : \mathcal{X} \to \mathcal{Y}$ over finite-dimensional Euclidean spaces $\mathcal{X}$ and $\mathcal{Y}$, its *adjoint* $\mathcal{A}^* : \mathcal{Y} \to \mathcal{X}$ is defined as the operator which satisfies

$$\langle \mathcal{A}x, y \rangle = \langle x, \mathcal{A}^* y \rangle \text{ for all } x \in \mathcal{X}, y \in \mathcal{Y}. \tag{1.0.14}$$

Since $\mathcal{X}$ and $\mathcal{Y}$ are finite and $\mathcal{A}$ is linear, $\mathcal{A}$ is also continuous. Thus the Riesz representation theorem guarantees that there will exist a unique linear operator $\mathcal{A}^*$ [**50**, Section 6.2]. In this dissertation, we will be concerned specifically with linear operators $\mathcal{A} : \mathcal{H}^n \to \mathbb{R}^m$, where $\mathcal{H}^n$ is the set of $n \times n$ Hermitian matrices. It is easily shown that all such linear operators $\mathcal{A}$ will have the

form

$$\mathcal{A}(X) = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix}, \tag{1.0.15}$$

where each A_i is some matrix in $\mathcal{H}^n$. In this case, the adjoint of $\mathcal{A}$ is given by

$$\langle \mathcal{A}(X), y \rangle = \sum_{i=1}^{m} \langle A_i, X \rangle y_i = \sum_{i=1}^{m} \langle y_i A_i, X \rangle = \langle X, \sum_{i=1}^{m} y_i A_i \rangle = \langle X, \mathcal{A}^* y \rangle, \tag{1.0.16}$$

where $X \in \mathcal{H}^n$ and $y \in \mathbb{R}^m$. Thus we have $\mathcal{A}^* y = \sum_{i=1}^{m} y_i A_i$.

Note that the PLGD objective function (defined in Section 3.2) is convex and generally non-differentiable, and thus we consider the subdifferential. Given a convex function $f : \mathcal{U} \to \mathbb{R}$ defined on an open, convex subset $\mathcal{U}$ of a finite-dimensional Euclidean space $\mathcal{X}$, the *subdifferential* of f at y_0 is defined as

$$\partial f(y_0) = \{ g \in \mathcal{X} \mid f(y) \geq f(y_0) + \langle g, y - y_0 \rangle \ \ \forall y \in \mathcal{U} \}, \tag{1.0.17}$$

and each element of $\partial f(y_0)$ is a *subgradient* of f.

The *Gaussian distribution* (or *normal distribution*) $\mathcal{N}(\mu, \sigma^2)$ is the distribution defined by the probability density function

$$f\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{1.0.18}$$

where μ is the mean and σ^2 the variance of the distribution. A real vector has Gaussian distribution $\nu \sim \mathcal{N}(\mu, \sigma^2)$ if all its elements have Gaussian distribution. Unless otherwise specified, the Gaussian distribution refers specifically to the *standard Gaussian distribution*, where $\mu = 0$ and $\sigma^2 = 1$. The *complex standard Gaussian distribution* is defined by the probability density function

$$f(z) = \frac{1}{\pi} e^{-|z|^2}. \tag{1.0.19}$$

CHAPTER 2

Noisy Phase Retrieval

2.1. Introduction

Given the variety of phase retrieval applications and the difficulty of solving the phase retrieval problem (1.0.1), a wide range of phase retrieval methods have been developed for both noisy and noiseless phase retrieval. In this chapter we review several of these methods, with particular attention toward methods for noisy phase retrieval. These methods are split into three classes: alternating projection methods (Section 2.2), structured optimization methods (Section 2.3) which rely on additional assumptions like sparsity of the desired signal $\mathbf{x}$, and unstructured optimization methods (Section 2.4) which only require an observation vector $b = \mathbf{b} + \eta$ and noise ratio $\epsilon_{\text{rel}} = ||\eta||_2/||b||_2$. These three classes were chosen to mirror the historical development of these methods (as the alternating projection methods preceded the others by a few decades) and emphasize the uniqueness of the three unstructured optimization methods discussed in Section 2.4. Note that the unstructured methods includes the Gauge Dual Descent (GDD) algorithm, a first-order algorithm which is discussed in Chapter 4 and central to this dissertation. Thus Section 2.4 offers greater detail for theoretical guarantees and numerical performance behavior.

For a recent survey of noiseless phase retrieval methods, see [**36**]. Also note Appendix A provides an additional comparison of models and algorithms for noisy phase retrieval. In Appendix A we show that the PLGD model first discussed in Section 2.4 is particularly well suited for developing a first-order optimization method, i.e., the GDD algorithm. Additionally, we demonstrate that the GDD algorithm is generally more accurate and robust to noise and low oversampling than the wflow algorithm also discussed in Section 2.4.

2.2. Alternating Projection Methods

We begin by discussing alternating projection methods for phase retrieval. These methods were established in the 1970s and 1980s as the first strategy for solving (1.0.1) and rely on prior

information about the signal, such as support constraints or positivity. Originally referred to as *error reduction algorithms*, the algorithms of this section were later identified as part of a larger class of nonconvex alternating projection methods [**40**]. These methods rely on projecting an iterate between the *object domain* (or time domain) in which we seek the desired signal $\mathbf{x}$, and the *frequency domain* in which we have the observation b.

We begin with two common alternating projection algorithms: the Gershberg-Saxton (GS) algorithm [**29**] and the hybrid input-output (HIO) algorithm [**24**], a variant of the GS algorithm which is still commonly used in practice. We then discuss recent alternating projection algorithms for noisy phase retrieval (oversampling smoothness (OSS) [**52**], error reduction (ER-) HIO and noise robust (NR-) HIO [**41**]). Note that these these alternating projection algorithms are usually not effective for noisy phase retrieval and typically require additional prior information.

2.2.1. Gershberg-Saxton Algorithm

Developed in 1972, the GS algorithm was the first alternating projection algorithm for phase retrieval and serves as an algorithmic foundation which later alternating projection algorithms would modify to improve the likelihood of convergence. Given the phase retrieval problem (1.0.1) with no oversampling (i.e., $L = 1$) and knowledge of the signal magnitude $c \in \mathbb{R}^n_+$ (i.e., $c_i = |\mathbf{x}_i|^2$ for all i), we may attempt to solve (1.0.1) with Algorithm 1.

Algorithm 1 Gershberg-Saxton (GS) algorithm

Input: Observation vector $b \in \mathbb{R}^n_+$, signal measurement vector $c \in \mathbb{R}^n_+$, DFT matrix $F \in \mathbb{C}^{n\times n}$.

Output: Approximate solution signal $x \in \mathbb{C}^n$.

Initialize: Choose a random signal $x_0 \in \mathbb{C}^n$, compute DFT $y_0 = Fx_0$, set $r_0 = |y_0|^2 - b$, $k = 0$.
while $||r_k||_2 >$ tol **do**
    Compute DFT of x_k and residual: $y_{k+1} = Fx_k$, $r_{k+1} = |y_{k+1}|^2 - b$.
    Impose observation magnitude constraints: $[y_{k+1}]_i = \frac{[y_{k+1}]_i}{|[y_{k+1}]_i|}\sqrt{b_i}$ for all $i = 1, \ldots, n$.
    Compute inverse DFT of y_{k+1}: $x_{k+1} = F^{-1}y_{k+1}$.
    Impose signal magnitude constraints: $[x_{k+1}]_i = \frac{[x_{k+1}]_i}{|[x_{k+1}]_i|}\sqrt{c_i}$ for all $i = 1, \ldots, n$.
    $k = k + 1$.
end while
Return: $x = x_k$.

During an iteration of the GS algorithm, knowledge of the signal and observation magnitudes, as well as any additional information, is applied when the iterate reaches that respective domain (steps 6 and 4, respectively). While the GS algorithm is notable as the first phase retrieval algorithm, this algorithm is also very likely to converge to a local rather than global minimum [**36**].

2.2.2. Hybrid Input-Output Algorithm

In [**24**], Fienup interpreted the GS algorithm as a nonlinear feedback control system (Figure 2.1), where the *System* component corresponds to the map $\widetilde{P}_{\text{freq}} = F^{-1}P_{\text{freq}}F$ (combining steps 3-5 of the GS algorithm, where P_{freq} is projection onto the frequency constraints in step 4). Since the GS algorithm interprets phase retrieval as an error-reduction problem, the system output $\widetilde{P}_{\text{freq}}(x_k)$ is viewed as the current candidate for the desired signal $\mathbf{x}$, and the signal magnitude constraints (GS algorithm, step 6) are imposed to arrive at a new input x_{k+1} which is considered the current approximation to the solution signal. By viewing phase retrieval as a nonlinear feedback problem, the input x_k is no longer treated as a signal approximation, and instead serves as feedback information for the system. Thus x_k need not satisfy the signal magnitude constraints.

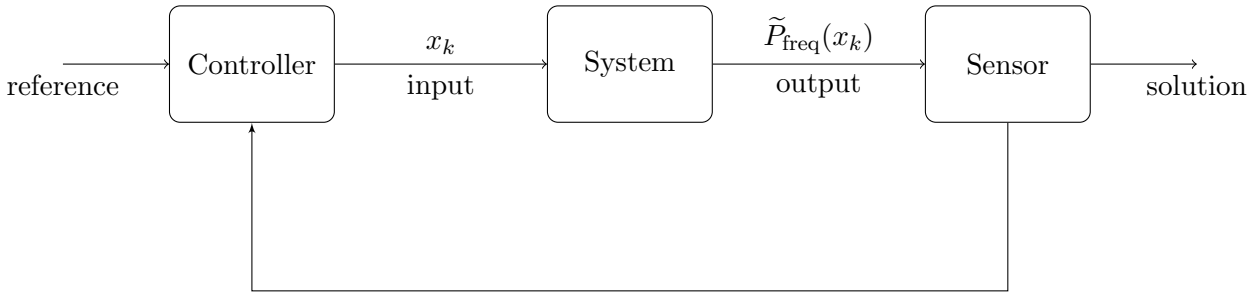

FIGURE 2.1. Phase retrieval as a nonlinear feedback control system

Fienup observed that a small change in the input will result in an output that is approximately a constant α times the change in the input, that is

$$\widetilde{P}_{\text{freq}}(x + \Delta x) - \widetilde{P}_{\text{freq}}(x) \approx \alpha \Delta x. \tag{2.2.1}$$

To force a change of Δx in the output $\widetilde{P}_{\text{freq}}(x)$, the input would logically be changed by $\beta\Delta x$, where $\beta = 1/\alpha$ from (2.2.1). This observation led Fienup to identify three new potential strategies for selecting an update (the *Controller* in Figure 2.1). The most successful of these strategies involves

replacing step 6 in the GS algorithm with the index-wise update

$$[x_{k+1}]_i = \begin{cases} [\widetilde{P}_{\text{freq}}(x_k)]_i & i \notin \mathcal{V} \\ [x_k]_i - \beta[\widetilde{P}_{\text{freq}}(x_k)]_i & i \in \mathcal{V}, \end{cases} \tag{2.2.2}$$

where $\mathcal{V}$ is the set of indices in which the update violates the object domain constraints. If we replace step 6 of the GS algorithm with (2.2.2), then we have the HIO algorithm. This simple corrective step (2.2.2) which HIO applies to the object domain makes the HIO algorithm much more likely than the GS algorithm to recover a signal from a low-noise observation [**36**]. Nevertheless, the HIO algorithm still has a tendency to converge to local minima and is not robust to noise. In practice, the user will often select a large number of random initial iterates to initialize the HIO algorithm and select the best resulting signal.

2.2.3. HIO-Type Methods

Recent developments in alternating projection algorithms for phase retrieval have focused on handling noisy observations. In [**52**], the authors modify the GS algorithm by taking the HIO update (2.2.2) and performing a Gaussian smoothing step in the frequency domain on the indices (pixels) violating the support constraint. Their oversampling smoothness (OSS) algorithm takes the update x_{k+1} from step 6 of the HIO algorithm and applies the additional update

$$[x_{k+1}]_i = \begin{cases} [x_{k+1}]_i & i \notin \mathcal{V} \\ \left[F^{-1}\left[F(x_{k+1}) \,.*\, \mathbf{w}(j,\alpha)\right]\right]_i & i \in \mathcal{V}, \end{cases} \tag{2.2.3}$$

where $j = 1, \ldots, N$ is the Fourier index, $.*$ is pointwise multiplication, and $\mathbf{w}(j,\alpha)$ is a normalized Gaussian function

$$\mathbf{w}(j,\alpha) = e^{-\frac{1}{2}\left(\frac{j}{\alpha}\right)^2}.$$

As $\alpha \to \infty$, the original HIO algorithm is recovered. The authors select α heuristically, with early $\alpha = \mathcal{O}(N)$ and later $\alpha = \mathcal{O}(1/N)$, causing the OSS algorithm to behave similarly to the HIO algorithm during early iterations while damping high frequency violations in later iterations. Their experiments [**52**, Section 3] apply Poisson noise with noise levels ranging from 0.05 to 0.25 (and measured in the 1-norm, but approximately equivalent to our definition $\epsilon_{\text{rel}} = ||\eta||_2/||b||_2$). Their

results suggest that the accuracy and consistency of the OSS algorithm is superior to HIO and two HIO variants: ER-HIO and NR-HIO of [**41**].

While HIO-type methods are computationally efficient and still commonly used in practice [**36**], they also require special parameter tuning (e.g., β in HIO and α, β in OSS) and prior information about the desired signal. Additionally, none of these methods are wholly robust to noise or guaranteed to converge, and require large batches of random initializations in practice to generate an adequate solution. To overcome these deficiencies, recent phase retrieval methods have largely avoided the alternating projection framework, instead casting (1.0.1) as a structured optimization problem.

2.3. Structured Optimization Methods

Next we discuss structured optimization methods for noisy phase retrieval. In the past decade, a wide range of methods have been crafted to take advantage of the structure of particular phase retrieval problems. The survey in this section highlights typical methods and summarizes a few theoretical guarantees for exact recovery or error bounds. The methods in this section are grouped in terms of the structural property each method seeks to exploit. The *compressive phase retrieval* methods assume sparsity in the desired signal $\mathbf{x}$, and include [**61**], the GrEedy Sparse PhAse Retrieval (GESPAR) method [**59**], and a thresholded wflow algorithm [**11**]. The *robust phase retrieval* methods assume sparsity in either the observation b [**38**] or the noise η [**37**]. The *supervised phase retrieval* methods require an approximate solution signal $\hat{x}$ for initialization, and include [**30**] and [**2**].

2.3.1. Compressive Phase Retrieval Methods

We begin by summarizing a few recent compressive phase retrieval methods and theoretical results. The authors of [**22**] consider the theoretical recovery guarantees for compressive phase retrieval. They prove signal uniqueness and stable recovery (a stronger property than invertibility) for a variety of sparsity assumptions. For instance, if the signal $\mathbf{x}$ is k-sparse and observation b is noiseless, then $\mathcal{O}(k\log(n/k))$ observations are necessary for signal uniqueness. If $\mathbf{x}$ is k-sparse and b is noisy, then $\mathcal{O}(k\log(n/k)\log(k))$ observations are necessary to guarantee stable recovery (which gives $\mathcal{O}(n\log(n))$ observations if $\mathbf{x}$ dense).

Many methods have been considered for handling compressive phase retrieval. In [**61**], the authors consider sparse signals with noisy observations and proceed by lifting the signal and sensing vectors (for details on lifting and the PhaseLift model, see Section 2.4). They construct a rank minimization problem similar to the PhaseLift rank minimization model (see Section 2.4), but with an additional mixed $1,2$-norm constraint

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n} X_{i,j}^2\right)^{1/2} \leq \zeta$$

which promotes row-sparsity of the lifted solution matrix X. The resulting algorithm also includes a thresholding step on the spectrum of X to enforce low-rankness in the solution. The authors provide comparative numerical results indicating the addition of this constraint/thresholding strategy in their optimization method decreases reconstruction error as the noise ratio increases. However, the overall model is nonconvex, the iterations are expensive (each requiring the inversion of a lifted matrix), and no theory is provided to guarantee convergence or signal recovery quality.

The GESPAR method of [**59**] assumes $\mathbf{x}$ is k-sparse and b is noisy, and constructs an algorithm which maintains an active set S of indices which converges to the appropriate k-sized set of active indices in the solution signal. This local search method does not require matrix lifting, making it efficient for large-scale phase retrieval. Numerical results [**59**, Section 5] indicate that as sparsity increases, GESPAR has a higher recovery probability than a PhaseLift algorithm and a sparse variant of the HIO algorithm.

A thresholded version of the wflow algorithm (see Section 2.4) is considered in [**11**]. Here, the authors assume that $\mathbf{x}$ is sparse and b is noisy, and add the penalty $\tau||x||_1$ to the wflow objective function. The resulting wflow-type model

$$\min_{x} \quad \tfrac{1}{2m}\sum_{i=1}^{m}\left(|a_i^* x|^2 - b_i\right)^2 + \tau||x||_1, \tag{2.3.1}$$

is a phase retrieval analog to the basis pursuit denoising model [**18**]. The authors show that for the proper choice of τ, their thresholded wflow algorithm has the minimax optimal rate of convergence, $\mathcal{O}(k\log(n)/m)$.

Another recent method for compressive phase retrieval considers unstructured noise η in the observation, and interprets the recovery of a k-sparse signal $\mathbf{x}$ as a covariance maximization problem between the observations b_i and the sensing values $|a_i^* x|^2$ over the appropriate k-dimensional subspace [**70**]. The authors show that only $\mathcal{O}(k)$ measurements are required for their algorithm to converge within ϵ error in a runtime of $\mathcal{O}(nk\log(1/\epsilon))$ iterations.

2.3.2. Robust Phase Retrieval Methods

Next we consider robust phase retrieval methods, where we find sparsity in the observation b rather than the signal $\mathbf{x}$. The authors of [**37**] assume the noise η is sparse. Whereas the minimization of the 2-norm is optimal for fitting against Gaussian noise, the 1-norm is more robust for distributions with heavier tails. Thus the authors minimize the objective function $\sum_{i=1}^{m} \left||a_i^* x|^2 - b_i\right|$ using an alternating direction method of multipliers (ADMM) algorithm. Their numerical results demonstrate that their algorithm achieves better accuracy than the wflow algorithm when recovering from an observation with Gaussian noise and 10% outliers.

In contrast, the authors of [**38**] consider phase retrieval when the observation vector b is itself sparse and noisy. They develop an HIO-type algorithm with noise suppression in the frequency domain and phase/amplitude filtering in the object domain. Numerical results show this algorithm achieves better accuracy under the given sparsity assumptions than the wflow algorithm and a filtered version of the GS algorithm.

2.3.3. Supervised Phase Retrieval Methods

The final set of structured optimization methods we consider are the supervised phase retrieval methods. Rather than assuming sparsity of the signal or observations, the authors of [**30**] and [**2**] assume the existence of an approximation $\hat{x}$ to the desired signal $\mathbf{x}$. In this supervised phase retrieval paradigm, the authors define the PhaseMax model

$$\begin{aligned} \max_{x} \quad & \mathrm{Re}\langle \hat{x}, x\rangle \\ \text{s.t.} \quad & |\langle a_i, x\rangle|^2 \le b_i, \quad i = 1, \dots, m, \end{aligned} \tag{2.3.2}$$

where $\mathrm{Re}(\cdot)$ maps a complex number to its real component. The PhaseMax model (2.3.2) has the benefits of being convex without requiring signal lifting. The dual to this problem is the widely-studied basis pursuit problem which has several efficient solution techniques [**18**], [**15**]. The authors

of [**30**] prove that the probability of exact recovery is dependent on the angle between $\mathbf{x}$ and $\hat{x}$. Their numerical results demonstrate that this probability quickly approaches 1 as oversampling increases.

2.4. Unstructured Optimization Methods

The final class of methods we examine are unstructured optimization methods. Unlike the methods discussed in Section 2.3, these methods place no assumptions on the signal $\mathbf{x}$, the sensing vectors a_i, or the observation $b = \mathbf{b} + \eta$ other than knowledge of the noise ratio $\epsilon_{\text{rel}} = ||\eta||_2/||b||_2$.

This section begins with an explanation of matrix lifting for the signal and sensing vectors. Next we discuss the PhaseLift model [**12**] and the wflow algorithm [**14**]. Theoretical and numerical results are included to highlight the effectiveness and robustness of these models and methods (for complete theoretical results, see [**14**], [**65**] for wflow and [**13**], [**16**] for PhaseLift). This discussion of PhaseLift and wflow leads us to the PhaseLift gauge dual model [**27**], another unstructured optimization model which is the subject of Chapters 3 and 4.

2.4.1. PhaseLift

The PhaseLift model was first introduced in [**12**], with additional theoretical results established in [**16**]. This model is based on the concept of matrix lifting. First we define the linear operator $\mathcal{A}$ which allows us to lift the nonlinear phase retrieval observation constraint from (1.0.1) into a higher-dimensional linear constraint. If we lift the n-dimensional signal x and sensing vectors a_i into rank-one Hermitian matrices $X = xx^*, A_i = a_i a_i^* \in \mathcal{H}^n$, then the sensing operator $\mathcal{A}$ is defined coordinate-wise to satisfy $[\mathcal{A}(X)]_i = \langle A_i, X\rangle = \text{tr}(a_i a_i^* xx^*) = |\langle a_i, x\rangle|^2$. Thus $\mathcal{A} : \mathcal{H}^n \to \mathbb{R}^m$ and its adjoint $\mathcal{A}^* : \mathbb{R}^m \to \mathcal{H}^n$ are defined as

$$\mathcal{A}(xx^*) := \begin{bmatrix} \langle a_1 a_1^*, xx^*\rangle \\ \vdots \\ \langle a_m a_m^*, xx^*\rangle \end{bmatrix} \quad \text{and} \quad \mathcal{A}^* y := \sum_{i=1}^m y_i a_i a_i^*. \tag{2.4.1}$$

In particular, if the experimental model is the one discussed in Chapter 1 then we have

$$\mathcal{A}(xx^*) = \begin{vmatrix} FC_1x \\ \vdots \\ FC_Lx \end{vmatrix}^2 = \text{diag}\left[\begin{pmatrix} FC_1 \\ \vdots \\ FC_L \end{pmatrix} (xx^*) \begin{pmatrix} FC_1 \\ \vdots \\ FC_L \end{pmatrix}^* \right], \tag{2.4.2}$$

$$\mathcal{A}^*y = \sum_{j=1}^{L} [FC_j]^* \text{Diag}(y_j) FC_j.$$

The lifted rank-one matrix $X = xx^* \in \mathcal{H}^n$ and sensing operator $\mathcal{A}$ are then used to lift the nonlinear phase retrieval constraints $|\langle a_i, x\rangle|^2 = b_i$ for $i = 1, \ldots, m$ into the linear constraint $\mathcal{A}(X) = b$. Thus (1.0.1) is equivalent to the left-most problem in the sequence of problems

$$\begin{array}{ll} \text{find} & x \\ \text{s.t.} & \mathcal{A}(xx^*) = b \end{array} \iff \begin{array}{ll} \text{find} & X \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \\ & \text{rank}(X) = 1 \end{array} \iff \begin{array}{ll} \min\limits_{X \in \mathcal{H}^n} & \text{rank}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0. \end{array} \tag{2.4.3}$$

If a rank-one solution $X = xx^*$ exists then this matrix satisfies the lifted model in the middle of (2.4.3), and the left implication holds. Likewise, the rank minimization model at right in (2.4.3) will have a rank-one solution, and the right implication holds.

The resulting rank minimization problem at right of (2.4.3) is NP hard, as it includes the cardinality minimization problem as a special case (see [**45**] or [**49**]). Thus the next step is to relax the discrete, nonconvex objective function rank(X). To generalize the right-most model in (2.4.3) for noisy phase retrieval, a norm bound is also applied to the sensing constraint. This gives the semidefinite program which defines the PhaseLift model [**12**], [**16**]

$$\begin{array}{lll} & \min\limits_{X \in \mathcal{H}^n} & ||X||_1 = \sum\limits_{i=1}^{n} \sigma_i(X) \\ \text{(PhaseLift)} & \text{s.t.} & ||\mathcal{A}(X) - b||_2 \leq \epsilon \\ & & X \succeq 0. \end{array} \tag{2.4.4}$$

Here the objective function $||X||_1$ refers to the Schatten p-norm (which is expressed as $\sum_{i=1}^{n} \sigma_i(X)$, tr$(X)$, or $\langle I, X\rangle$ in various literature). While we can evaluate the function $||X||_1$ by computing

the trace of X, evaluating the subdifferential of this function requires a (partial) SVD. Also note that the term $\epsilon = ||\eta||_2$ measures the total noise, whereas $\epsilon_{\text{rel}} = ||\eta||_2/||b||_2$ discussed in Chapter 1 measures the noise ratio (or relative noise). We choose the Schatten p-norm to highlight the fact that (2.4.4) is the semidefinite analog to the celebrated basis pursuit denoising problem [**18**], [**15**],

$$\begin{aligned} \min_x \quad & ||x||_1 \\ \text{s.t.} \quad & ||Ax - b||_2 \leq \epsilon, \end{aligned} \tag{2.4.5}$$

where minimizing $||x||_1$ serves as a convex relaxation to minimizing the discrete, nonconvex function $||x||_0 = \text{nnz}(x)$.

The next two theorems quantify the relaxation of the PhaseLift model (2.4.4) from the phase retrieval model (1.0.1), establishing exact and approximate recovery guarantees for the PhaseLift model [**13**], [**16**]. Theorem 2.4.1 applies to the noiseless case (where $\epsilon = 0$ in the PhaseLift model), establishing a probability of equivalence between (1.0.1) and (2.4.4).

THEOREM 2.4.1. *Consider an arbitrary signal* $\mathbf{x}$ *in* $\mathbb{R}^n$ *or* $\mathbb{C}^n$ *and assume the sensing vectors* a_i *are independent, uniformly distributed on the unit sphere of* $\mathbb{R}^n$ *or* $\mathbb{C}^n$*. Suppose that the number of measurements obeys* $m \geq c_0 n$*, where* c_0 *is a sufficiently large constant. Then in both the real and complex cases, the solution to (2.4.4) is exact with high probability in the sense that the noiseless PhaseLift problem (i.e., (2.4.4) with* $\epsilon = 0$*) has a unique solution obeying*

$$X = \mathbf{x}\mathbf{x}^*.$$

This holds with probability at least $1 - \mathcal{O}(e^{-\gamma m})$*, where* γ *is a positive absolute constant.*

PROOF. See [**13**, Section 2]. □

The assumptions of Theorem 2.4.1 are met by the experimental models used in [**12**], [**16**], [**27**], and this dissertation (see Chapter 1, where the masks $C_i \in \mathbb{C}^{n \times n}$ from (1.0.2) have diagonal entries chosen with Gaussian distribution). Thus, if L is the oversampling rate (i.e., $m = nL$) then Theorem 2.4.1 shows that only $L = \mathcal{O}(1)$ masks are required to guarantee exact signal recover (up to global phase) with high probability.

Theorem 2.4.2 applies to the noisy phase retrieval case ($\epsilon > 0$), where there is no guarantee that the PhaseLift (2.4.4) solution matrix X is low rank. Thus the solution signal x is set to an appropriate rescaling of the eigenvector v_1 corresponding to the algebraically largest eigenvalue $\lambda_1 = \lambda_1(X)$, giving

$$x = \sqrt{\lambda_1} v_1. \tag{2.4.6}$$

THEOREM 2.4.2. *Consider an arbitrary signal* $\mathbf{x}$ *in* $\mathbb{R}^n$ *or* $\mathbb{C}^n$ *and assume the sensing vectors* a_i *are independent, uniformly distributed on the unit sphere of* $\mathbb{R}^n$ *or* $\mathbb{C}^n$*. Also suppose that the number of measurements obeys* $m \geq C_0 n \log n$*, where* C_0 *is a sufficiently large constant. Then the PhaseLift (2.4.4) solution* X *obeys*

$$||X - \mathbf{x}\mathbf{x}^*||_F \leq C_0 \epsilon, \tag{2.4.7}$$

for some positive constant C_0*. Additionally, we have*

$$||x - e^{i\theta}\mathbf{x}||_2 \leq C_0 \min(||\mathbf{x}||_2, \epsilon/||\mathbf{x}||_2), \tag{2.4.8}$$

where x *is defined as in (2.4.6) and* $\theta \in [0, 2\pi]$*. Both of these estimates hold with probability at least* $1 - \mathcal{O}(e^{-\gamma \frac{m}{n}})$*, where* γ *is a positive absolute constant.*

PROOF. See [**16**, Section 6]. □

The strength of the PhaseLift model lies in its convexity and generality (no signal or observation assumptions are necessary). However, in practice the objective function $||X||_1$ is difficult to optimize, as evaluation of the subdifferential $\partial||X||_1$ requires a (partial) SVD of X.

2.4.2. Wflow

Like the PhaseLift model, the recent Wirtinger flow (wflow) model [**14**] also seeks to solve the phase retrieval problem (1.0.1) without any assumptions on the structure or sparsity of the signal or observation. However, unlike the PhaseLift model, the wflow model does not involve matrix lifting and instead frames (1.0.1) as the least-squares problem

$$\min_x \quad \tfrac{1}{2m} \sum_{i=1}^{m} \left(|a_i^* x|^2 - b_i\right)^2 = \tfrac{1}{2m} ||\mathcal{A}(xx^*) - b||_2^2. \tag{2.4.9}$$

While the wflow model (2.4.9) is nonconvex, the authors of [**14**] develop an efficient gradient descent-like algorithm (the wflow algorithm) which has a provable guarantee for exact recovery when initialized appropriately. Initialization of the wflow algorithm involves a rescaling of the eigenvector v_1 corresponding to the algebraically largest eigenvalue of the matrix $\mathcal{A}^*b$, which is itself the sum of the outer products of the measurement vectors, scaled with the observation magnitudes

$$\mathcal{A}^*b := \sum_{i=1}^{m} b_i a_i a_i^*. \tag{2.4.10}$$

The authors motivate this choice of initialization by noting that if the sensing vectors a_i are i.i.d. with standard Gaussian distribution, and $||\mathbf{x}||_2 = 1$, then the expected value of the outer product sum $\mathcal{A}^*b$ will be

$$\mathbb{E}\left[\frac{1}{m}\mathcal{A}^*b\right] = I + 2\mathbf{x}\mathbf{x}^*. \tag{2.4.11}$$

Thus for large m, (2.4.10) has a high probability of returning a vector v_1 closely aligned with the solution $\mathbf{x}$. Intuitively, this initialization can also be seen as maximizing $\langle \mathcal{A}(vv^*), b\rangle = \langle vv^*, \mathcal{A}^*b\rangle = \langle v, [\mathcal{A}^*b]v\rangle$. Hence selecting the eigenvector v_1 corresponding to the algebraically largest eigenvalue of $\mathcal{A}^*b$ should promote $\mathcal{A}(v_1v_1^*)$ being collinear with b.

With this initialization, the authors recommend a gradient descent-like method with a preset stepsize strategy. Let $f(x) = \frac{1}{2m}||\mathcal{A}(xx^*) - b||_2^2$, and the residual $r = \frac{1}{m}[\mathcal{A}(xx^*) - b]$. The mapping $f : x \to \frac{1}{2m}||\mathcal{A}(xx^*) - b||_2^2$ from $\mathbb{C}^n$ to $\mathbb{R}$ is not holomorphic, and thus not complex-differentiable. As a result, the authors appeal to Wirtinger derivatives [**14**, Section 6] for the descent direction

$$\begin{aligned} \nabla f(x) &= \frac{1}{m}[\mathcal{A}^*(\mathcal{A}(xx^*) - b)]x \\ &= \frac{1}{m}[\mathcal{A}^*r]x \\ &= \frac{1}{m}\left[\sum_{j=1}^{L} C_j^* F^* \mathrm{Diag}(r_j) F C_j\right] x. \end{aligned} \tag{2.4.12}$$

The stepsize is then chosen using the heuristics [**14**, Section 2]

$$\mu_k = \min\{1 - e^{-k/k_0}, \mu_{\max}\} \qquad k_0 = 330,\ \mu_{\max} = 0.4. \tag{2.4.13}$$

This descent direction and stepsize choice lead to the wflow algorithm.

Algorithm 2 wflow algorithm

Input: Sensing operator $\mathcal{A}$ (2.4.1) with sensing vectors a_i for $i = 1, 2, \ldots, m$, observation vector $b \in \mathbb{R}^m_+$.

Output: Approximate solution signal x.

1: *Initialize:* Compute the algebraically largest eigenpair (λ_1, v_1) of $\mathcal{A}^*b$, set $x_0 = \alpha v_1$ where $\alpha^2 = n \sum_{i=1}^m b_i / \sum_{i=1}^m ||a_i||_2^2$, set $r_0 = \frac{1}{m}\left[\mathcal{A}(x_0 x_0^*) - b\right]$ and $k = 0$.

2: **while** $||r_k||_2 >$ tol **do**

3: Compute Wirtinger derivative: $\nabla f(x_k)$ from (2.4.12).

4: Compute stepsize: μ_{k+1} from (2.4.13) and set $\alpha_k = \frac{\mu_{k+1}}{||x_0||_2^2}$.

5: Compute signal update: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.

6: Compute residual: $r_{k+1} = \frac{1}{m}\left[\mathcal{A}(x_{k+1}x_{k+1}^*) - b\right]$.

7: $k = k + 1$.

8: **end while**

9: *Return:* $x = x_k$.

In [**14**, Section 2.3], it is observed that the wflow algorithm can be interpreted as a stochastic gradient scheme, where $\nabla f(x)$ is an unbiased estimate of an ideal gradient $\nabla F(x)$, with

$$F(x) = x^* \left(I - \mathbf{x}\mathbf{x}^*\right) x - \frac{3}{4}\left(||x||_2^2 - 1\right)^2. \tag{2.4.14}$$

Interestingly, we observe that the wflow algorithm can also be interpreted as an alternating projection HIO-type algorithm, where computation of the Wirtinger derivative (2.4.12) corresponds to steps 3-5 of the GS algorithm and the signal update $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ corresponds to step 6 of the GS algorithm. More precisely, in the last line of (2.4.12) we see x_k is first mapped to the frequency domain (Algorithm 1, step 3). This vector is then damped coordinate-wise by the corresponding residual values $\mathrm{Diag}(r_j)$ for $j = 1, \ldots, L$ (Algorithm 1, step 4). Finally, the observation is mapped back to the object domain (Algorithm 1, step 5) and damped by α_k to generate the signal update $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$ (similar to the HIO update (2.2.2) which replaces Algorithm 1, step 6).

Yet unlike other alternating projection algorithms, the wflow algorithm does not require additional frequency domain information or heuristics. The wflow algorithm replaces the frequency domain damping in Algorithm 1, step 4 with a damping which uses the observation residual $r = \mathcal{A}(xx^*) - b$ as an exact, real-time measurement of the frequency domain violations of x. In this view, $\mathrm{Diag}(r_j)$ in (2.4.12) provides a heuristic-free frequency domain damping. Consequentially, the wflow algorithm merges the computational simplicity of an alternating projection method (Section 2.2) with the broad utility of an unstructured optimization method. Theorem 2.4.3 provides an exact recovery probability for Algorithm 2 applied to noiseless phase retrieval problems.

THEOREM 2.4.3. *Consider an arbitrary signal* $\mathbf{x}$ *in* $\mathbb{C}^n$ *and assume the sensing vectors* a_i *are independent, uniformly distributed on the unit sphere of* $\mathbb{C}^n$*. Suppose that the number of measurements obeys* $m \geq c_0 n \log(n)$*, where* c_0 *is a sufficiently large constant. Then the wflow initial estimate* x_0 *normalized to have squared Euclidean norm equal to* $(\sum_{i=1}^m b_i)/m$ *has*

$$\mathrm{dist}(x_0, \mathbf{x}) := \min_{\theta \in [0,2\pi]} ||x_0 - e^{i\theta}\mathbf{x}||_2 \leq \frac{1}{8}||x||_2 \tag{2.4.15}$$

with probability at least $1 - 10e^{-\gamma n} - 8/n^2$*, where* γ *is a positive absolute constant. Additionally, assume the stepsize is a bounded constant* $\mu_k = \mu \leq c_1/n$ *for some fixed numerical constant* c_1*. Then there is an event of probability at least* $1 - 13e^{-\gamma n} - me^{-1.5m} - 8/n^2$*, such that on this event, starting from any initial solution* x_0 *obeying (2.4.15), we have*

$$\mathrm{dist}(x_k, \mathbf{x}) \leq \frac{1}{8}||\mathbf{x}||_2 \left(1 - \frac{\mu}{4}\right)^{k/2}. \tag{2.4.16}$$

PROOF. See [**14**, Section 7]. □

Computationally, the wflow algorithm is very efficient at recovering the signal of an oversampled noiseless observation. Numerical experiments in [**14**, Section 4] demonstrate the effectiveness of the wflow algorithm for 1-D random signals, 2-D natural images, and 3-D molecular structures.

More generally, it was proved in [**65**] that if the observation vectors a_i are independent, uniformly distributed on the unit sphere of $\mathbb{C}^n$ and there are $m \geq c_0 n \log^3(n)$ measurements, then with probability at least $1 - c_0/m$ the function $f(x) = \frac{1}{2m}||\mathcal{A}(xx^*) - b||_2^2$ has the following properties:

- f has no spurious local minima,

- all global minima are equal to $\mathbf{x}$ up to some phase constant: $x = e^{i\theta}\mathbf{x}$, and
- f has negative directional curvature at all saddle points.

Thus when solving (2.4.9), algorithms such as the wflow algorithm do not require specialized initialization, and are likewise guaranteed to find a global minimum given sufficient oversampling.

However, the wflow algorithm can diverge when significant noise exists or the oversampling rate is too low (see Appendix A). In contrast, the Gauge Dual Descent (GDD) algorithm of Chapter 4 is guaranteed to converge and thus well suited for unstructured noisy phase retrieval problems.

CHAPTER 3

Gauge Duality Theory

3.1. Introduction

This chapter presents the gauge duality theory necessary for optimizing the PhaseLift gauge dual (PLGD) model. Section 3.2 introduces the PhaseLift primal (PLP) and PLGD pair of models along with relevant definitions and a brief history of gauge duality theory. In Section 3.3 we prove the gauge duality theorem (Theorem 3.3.1) for a more general primal-gauge dual model pair and establish weak duality, strong duality, and optimality conditions for this pair. Section 3.4 then returns to the PLP-PLGD pair, establishing key properties for optimizing the PLGD model and recovering a signal x from a given dual variable y.

All of the gauge properties and gauge duality results in this chapter were previously established in [**51**], [**26**], [**28**], and [**27**]. Gauge functions were first analyzed in [**51**]. Gauge duality was then introduced in [**26**], where the author focused on quadratic programming applications. In [**28**], the authors developed a broad set of antipolar calculus results for the analysis of gauge duality. These results were then used in [**27**] to develop the Gauge Dual Descent (GDD) algorithm (Algorithm 3 in Section 4.2), a first-order algorithm for the PLP-PLGD pair.

Our contribution is to provide a self-contained, comprehensive treatment of gauge duality theory for the PLP-PLGD pair. Prior to this treatment, the results in this chapter were spread throughout the original texts ([**51**], [**28**], and [**27**]), occasionally with differing notation or style. In contrast, Section 3.2 provides a single notation and style which is used in Sections 3.3 and 3.4 to develop gauge duality theory for the PLP-PLGD pair. The results of this chapter provide the theoretical foundation for developing the GDD algorithm for optimizing the PLGD model in Chapter 4.

3.2. Primal-Gauge Dual Models and Background

We begin this section by stating the PhaseLift primal (PLP) and PhaseLift gauge dual (PLGD) pair of models and reviewing key definitions. We then compare gauge duality with Lagrange duality and provide a brief history of gauge duality theory.

3.2.1. Models and Definitions

The PhaseLift primal semidefinite program (restated from (2.4.4) in Section 2.4) and its gauge dual are

$$\begin{array}{llll} & \min\limits_{X} & ||X||_1 = \sum\limits_{i=1}^{n} \sigma_i(X) & \\ \text{(PLP)} & \text{s.t.} & ||\mathcal{A}(X) - b||_2 \leq \epsilon & \\ & & X \succeq 0 & \end{array} \qquad \begin{array}{lll} \min\limits_{y} & \lambda_1(\mathcal{A}^* y) & \\ \text{s.t.} & \langle b, y \rangle - \epsilon ||y||_2 \geq 1, & \text{(PLGD)} \end{array} \tag{3.2.1}$$

where the linear operator $\mathcal{A}$ and its adjoint $\mathcal{A}^*$ are defined in (2.4.1) and $\lambda_1(\mathcal{A}^* y)$ is a function of $y \in \mathbb{R}^m$ which returns the algebraically largest eigenvalue of $\mathcal{A}^* y$. Note that in Section 3.4 we discuss how an approximate signal x may be recovered from PLGD solution y.

This section places the PLP-PLGD pair (3.2.1) in the context of gauge duality theory. We begin by discussing definitions and basic properties which are relevant to gauge duality theory. We then take a moment to highlight some parallels between gauge duality and Lagrange duality before closing with a summary of major developments in gauge duality theory. Along the way, we present three additional, more general primal-gauge dual models which have the PLP-PLGD pair (3.2.1) as a special case. Two of these models are then used in Section 3.3 to develop the theory which establishes the gauge duality of the PLP-PLGD pair.

The PLP-PLGD pair (3.2.1) is an example of a more general primal-gauge dual pair which we define as the *nonlinearly-constrained* pair

$$\begin{array}{lll} & \min\limits_{x \in \mathcal{X}} & \kappa(x) \\ \text{(P-nonlin)} & \text{s.t.} & x \in \mathcal{C} \end{array} \qquad \begin{array}{lll} \min\limits_{z \in \mathcal{X}} & \kappa^\circ(z) & \\ \text{s.t.} & z \in \mathcal{C}'. & \text{(GD-nonlin)} \end{array} \tag{3.2.2}$$

Here, $\mathcal{C}$ and $\mathcal{C}'$ are subsets of $\mathcal{X}$, a finite-dimensional Euclidean space. The set $\mathcal{C}'$ is the *antipolar* of $\mathcal{C}$, defined as

$$\mathcal{C}' = \{z \mid \langle x, z \rangle \geq 1 \ \forall x \in \mathcal{C}\}. \tag{3.2.3}$$

This is in contrast to the *polar* of $\mathcal{C}$, which is defined as

$$\mathcal{C}^\circ = \{z \mid \langle x, z\rangle \leq 1 \ \forall x \in \mathcal{C}\}. \tag{3.2.4}$$

Although we refer to the primal-gauge dual pair (3.2.2) as nonlinear, we are primarily concerned with closed, convex sets $\mathcal{C}$ which do not contain the origin (e.g., the PLP (3.2.1) constraint set when $\epsilon < ||b||_2$). Note that gauge dual models with linear constraints which do not include the origin are a subset of the models described by (3.2.2).

The functions $\kappa, \kappa^\circ : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ in (3.2.2) are *gauge* functions, meaning they are convex, nonnegative, positively homogeneous ($\kappa(\alpha x) = \alpha\kappa(x)$ for all $\alpha > 0$), and vanish at the origin. Gauge functions generalize the notion of norms, allowing for flexibility in modeling the phase retrieval problem. In the PLP model for instance, the Schatten 1-norm $\kappa(X) := ||X||_1$ and vector 2-norm $\rho(y) := ||y||_2$ are both gauges.

Given a gauge function κ, the *polar* of this function is the function κ° that most tightly satisfies the inequality

$$\langle x, z\rangle \leq \kappa(x)\kappa^\circ(z) \quad \forall x \in \text{dom } \kappa, \ \forall z \in \text{dom } \kappa^\circ. \tag{3.2.5}$$

Equivalently, the polar may be defined as [**51**, Section 15]

$$\kappa^\circ(z) = \inf\{\mu > 0 \mid \langle x, z\rangle \leq \mu\kappa(x) \ \forall x\}. \tag{3.2.6}$$

Note that the polar is a generalization of the *dual norm*, which is defined as

$$||z||_* = \sup_x \{\langle x, z\rangle \mid ||x|| \leq 1\}. \tag{3.2.7}$$

The *preimage* of a linear operator $A : \mathcal{X} \to \mathcal{Y}$ over the set $\mathcal{S} \subseteq \mathcal{Y}$ is defined as

$$A^{-1}\mathcal{S} = \{x \in \mathcal{X} \mid Ax \in \mathcal{S}\}. \tag{3.2.8}$$

The *closure* of a set $\mathcal{S} \subseteq \mathcal{X}$ is denoted $\mathrm{cl}(\mathcal{S})$. The *affine hull* of $\mathcal{S}$ is the set of all affine combinations of elements of $\mathcal{S}$, or

$$\mathrm{aff}(\mathcal{S}) = \left\{ \sum_{i=1}^{k} \alpha_i x_i \;\middle|\; k > 0,\ x_i \in \mathcal{S},\ \alpha_i \in \mathbb{R},\ \sum_{i=1}^{k} \alpha_i = 1 \right\}. \tag{3.2.9}$$

The *relative interior* of $\mathcal{S}$, denoted $\mathrm{ri}(\mathcal{S})$, is the interior within the affine hull of $\mathcal{S}$, i.e.,

$$\mathrm{ri}(\mathcal{S}) = \{x \in \mathcal{S} \mid \exists\ \epsilon > 0,\ B_\epsilon(x) \cap \mathrm{aff}(\mathcal{S}) \subseteq \mathcal{S}\}, \tag{3.2.10}$$

where $B_\epsilon(x)$ is a ball of radius ϵ centered at x. The *support* function $\sigma_{\mathcal{C}}$ of a nonempty convex set $\mathcal{C}$ is defined as

$$\sigma_{\mathcal{C}}(z) = \sup_{x \in \mathcal{C}} \langle x, z \rangle. \tag{3.2.11}$$

And the *Minkowski* function $\gamma_{\mathcal{C}}$ is defined as

$$\gamma_{\mathcal{C}}(x) = \inf \{\lambda \geq 0 \mid x \in \lambda \mathcal{C}\}. \tag{3.2.12}$$

If there is no λ such that $\lambda x \in \mathcal{C}$, then $\gamma_{\mathcal{C}}(x) = +\infty$. Note that any gauge κ is a Minkowski function $\gamma_{\mathcal{C}}$ for $\mathcal{C} = \{x \mid \kappa(x) \leq 1\}$ [**51**, Section 15]. Given a function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$, the *epigraph* of f is defined as

$$\mathrm{epi}(f) = \{(x, \tau) \mid f(x) \leq \tau\}. \tag{3.2.13}$$

Note that f is convex if and only if $\mathrm{epi}(f)$ is convex [**51**, Section 7]. Thus the function f is said to be *closed* if $\mathrm{epi}(f)$ is closed. Additionally, f is closed if and only if it is lower-semicontinuous (that is, $\liminf_{x \to x_0} f(x) \geq f(x_0)$ for all x_0 in $\mathrm{dom}(f)$) [**51**, Section 7]. Also, f is *proper* if the domain of f, $\mathrm{dom}(f) = \{x \mid f(x) < +\infty\}$ is nonempty.

If κ is a closed gauge, then its polar may also be expressed as [**51**, Section 15]

$$\kappa^\circ(z) = \sup_x \{\langle x, z \rangle \mid \kappa(x) \leq 1\}, \tag{3.2.14}$$

again highlighting the fact that the polar function is a generalization of the dual norm (3.2.7). Additionally, if κ is also positive everywhere except at the origin then its polar may also be defined

as [**51**, Section 15]

$$\kappa^\circ(z) = \sup_x \left\{ \frac{\langle x, z \rangle}{\kappa(x)} \right\}. \tag{3.2.15}$$

3.2.2. Development of Gauge Duality and Relation to Lagrange Duality

Given the gauge duality notation discussed above, we take a moment to contrast gauge duality with the much more common Lagrange duality. Whereas gauge duality involves multiplicative duality transformations, Lagrange duality is additive in nature. The reader may see [**8**, Chapter 5] for a comprehensive introduction to Lagrange duality, and [**51**, Section 28] or [**6**, Chapter 2] for a treatment of Lagrange duality theory.

Given P-nonlin, the primal model in (3.2.2), we see that its gauge dual GD-nonlin can be stated simply using the polar function κ° and the antipolar set $\mathcal{C}'$. Similarly, the Lagrange dual of P-nonlin can be described using the appropriate function and set transformations. Given a function $f : \mathcal{X} \to \mathbb{R} \cup \{\pm\infty\}$, the *convex conjugate* f^* is defined as the function that most tightly satisfies the inequality

$$\langle x, z \rangle \leq f(x) + f^*(z) \quad \forall x \in \text{dom } f, \ \forall z \in \text{dom } f^*. \tag{3.2.16}$$

Equivalently, the convex conjugate may be defined as [**51**, Section 12]

$$f^*(z) = \sup_x \{ \langle x, z \rangle - f(x) \}. \tag{3.2.17}$$

We see that (3.2.16) and (3.2.17) are the additive analogs of the polar function definitions (3.2.5) and (3.2.15), respectively. Additionally, for a set $\mathcal{S} \subseteq \mathcal{X}$, the *dual cone* is defined as

$$\mathcal{S}^* = \{ z \mid \langle x, z \rangle \leq 0 \ \forall x \in \mathcal{S} \}. \tag{3.2.18}$$

Given the convex conjugate κ^* and the dual cone $\mathcal{C}^*$, the Lagrange dual D-nonlin and gauge dual GD-nonlin both have the simple forms

$$\begin{array}{llll} & \max\limits_z & -\kappa^*(z) & \\ \text{(D-nonlin)} & \text{s.t.} & z \in \mathcal{C}^* & \end{array} \qquad \begin{array}{llll} & \min\limits_{z \in \mathcal{X}} & \kappa^\circ(z) & \\ \text{(GD-nonlin)} & \text{s.t.} & z \in \mathcal{C}'. & \end{array} \tag{3.2.19}$$

Note that Lagrange duality applies for any function $f : \mathcal{X} \to \mathbb{R} \cup \{\pm\infty\}$ and set $\mathcal{S} \subseteq \mathcal{X}$. Thus Lagrange duality has been studied thoroughly and applied extensively (e.g., see [**8**, Chapter 5]). In contrast, gauge duality places specific restrictions on the objective function and constraint set. As a result, the development of gauge duality theory has a brief, sporadic history.

In 1970, Rockafellar thoroughly analyzed gauge functions and their polars [**51**, Part III]. The concept of gauge duality was then introduced by Freund in 1987 [**26**]. This seminal work focused primarily on the *linearly-constrained* primal and gauge dual pair

$$\begin{array}{rlll} & \min\limits_{x\in\mathcal{X}} & \kappa(x) & \\ \text{(P-lin)} & \text{s.t.} & Ax = b & \end{array} \qquad \begin{array}{rlll} & \min\limits_{y\in\mathcal{Y}} & \kappa^\circ(A^*y) & \\ \text{(GD-lin)} & \text{s.t.} & \langle b, y\rangle = 1. & \end{array} \tag{3.2.20}$$

Here $A : \mathcal{X} \to \mathcal{Y}$ is a linear operator over finite-dimensional Euclidean spaces. As with Lagrange duality, if the primal constraint is replaced with $Ax \geq b$ then the gauge dual also includes the constraint $y \geq 0$. Freund develops strong duality and optimality conditions for the linear pair (3.2.20) based on polarity relationships for sets and gauge functions. His work also establishes these conditions for the nonlinear pair (3.2.2). In this case, his work requires $\mathcal{X}$ and $\mathcal{Y}$ to be *ray-like* (meaning that for all $x, y \in \mathcal{X}$ we have $x + \alpha y \in \mathcal{X}$ for all $\alpha \geq 0$). The addition of the ray-like property to $\mathcal{X}$ (along with closed, convex, and containing the origin) guarantees $\mathcal{X}'' = \mathcal{X}$.

Gauge duality theory was revisited in [**28**], where the authors consider the *inequality-constrained* primal and gauge dual pair

$$\begin{array}{rlll} & \min\limits_{x\in\mathcal{X}} & \kappa(x) & \\ \text{(P-ineq)} & \text{s.t.} & \rho(Ax - b) \leq \epsilon & \end{array} \qquad \begin{array}{rlll} & \min\limits_{y\in\mathcal{Y}} & \kappa^\circ(A^*y) & \\ \text{(GD-ineq)} & \text{s.t.} & \langle b, y\rangle - \epsilon\rho^\circ(y) \geq 1. & \end{array} \tag{3.2.21}$$

As we will see in Section 3.4, the PLP-PLGD pair (3.2.1) pair is recovered from P-GD-ineq (3.2.21) when we set $\kappa(X) = ||X||_1 + \delta_{(\cdot)\geq 0}(X)$ and $\rho(y) = ||y||_2$. The authors of [**28**] develop an antipolar calculus to determine the antipolar for sets like $\{y \mid \rho(Ax - b) \leq \epsilon\}$ and use this calculus rather than polarity relations to derive the pair P-GD-ineq (3.2.21) and establish conditions such as those for strong duality. Additionally, this antipolar calculus allows the authors to drop the requirement that the sets $\mathcal{X}$ and $\mathcal{Y}$ are ray-like.

In contrast to the antipolar calculus framework, the authors of [**1**] develop gauge duality through a perturbation framework. This even broader setting frames gauge duality as a product of Fenchel-Rockafellar duality, allowing the consideration of gauge duality for general nonnegative convex functions.

3.3. The Gauge Duality Theorem and Optimality Conditions

In this section we develop gauge duality theory for the inequality-constrained primal and gauge dual (P-GD-ineq) pair (3.2.21). Section 3.3.1 establishes a set of supporting propositions which are used in Section 3.3.2 to prove the Gauge Duality Theorem (Theorem 3.3.1). We follow a two-track proof strategy depicted in Figure 3.1, showing that the gauge duality of P-GD-ineq (3.2.21) is a consequence of the duality of both the nonlinear (3.2.2) and linear (3.2.20) pairs.

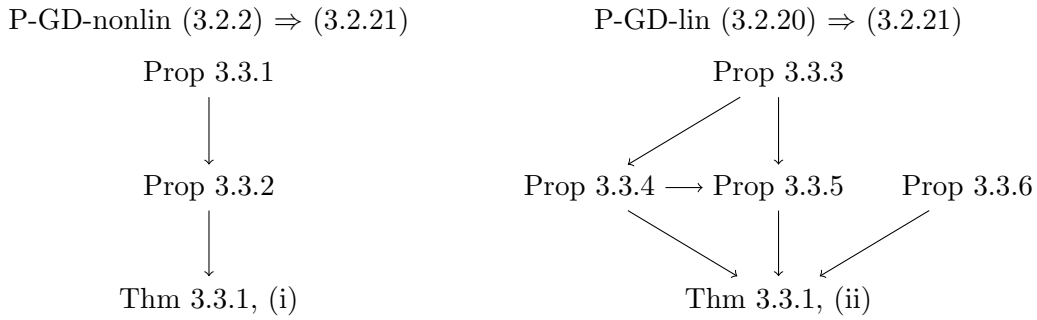

FIGURE 3.1. Dependency chart for proofs (i) and (ii) of Theorem 3.3.1 which establish the gauge duality of P-GD-ineq (3.2.21).

Finally, Section 3.3.3 establishes weak duality, strong duality, and optimality conditions for P-GD-ineq (3.2.21). In Section 3.4 we will return to the PhaseLift model (3.2.1), where the optimality conditions in this section provide a method for recovery of a primal signal x from a dual variable y.

The results in this section are based on [**51**], [**26**], and especially [**28**], and rely on the analysis of polarity relations rather than perturbation analysis as discussed in [**1**]. In particular, the two-track proof strategy depicted in Figure 3.1 was first developed in [**28**] as part of a larger treatment of antipolar calculus and the authors referred to [**51**] for more elementary results. This section provides a self-contained development of gauge duality for P-GD-ineq (3.2.21).

3.3.1. Supporting Propositions

Before presenting the Gauge Duality Theorem (Theorem 3.3.1 of Section 3.3.2), we present a set of supporting propositions. The first two propositions below are used in Theorem 3.3.1, (i) to show P-GD-ineq (3.2.21) is an instance of the nonlinear pair (3.2.2). Since P-GD-ineq (3.2.21) allows for the constraint set $\mathcal{C}$ to be any closed, convex set not containing the origin, we must simply establish the antipolar of the constraint set $\mathcal{C} = \{x \,|\rho(Ax - b) \leq \epsilon\}$. We begin by showing how linear and polar transformations on $\mathcal{C}$ commute under certain assumptions.

PROPOSITION 3.3.1. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite-dimensional Euclidean spaces, $\mathcal{C} \subseteq \mathcal{X}$ a closed, convex set which does not contain the origin, $A : \mathcal{X} \to \mathcal{Y}$ a linear operator, and $A^* : \mathcal{Y} \to \mathcal{X}$ its adjoint. Then*

$$(A\mathcal{C})' \;=\; (A^*)^{-1}\,\mathcal{C}'. \tag{3.3.1}$$

Additionally, assume $\mathcal{C}$ is polyhedral or $\mathrm{ri}(\mathcal{C}) \cap \mathrm{range}(A) \neq \emptyset$, and $A^{-1}\mathcal{C}$ is not empty. Then $(A^{-1}\mathcal{C})'$ is nonempty and the following set equality holds

$$\left(A^{-1}\mathcal{C}\right)' \;=\; A^*\mathcal{C}'. \tag{3.3.2}$$

PROOF. The first result is proved in [**28**, Proposition 3.3] and the second in [**28**, Proposition 3.4, 3.5]. □

The previous propositions allows us to construct the antipolar of the constraint set $\mathcal{C} = \{x \mid \rho(Ax - b) \leq \epsilon\}$ in Proposition 3.3.2.

PROPOSITION 3.3.2. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite-dimensional Euclidean spaces, $A : \mathcal{X} \to \mathcal{Y}$ a linear operator, and $A^* : \mathcal{Y} \to \mathcal{X}$ its adjoint. Let $\mathcal{C} = \{x \mid \rho(Ax - b) \leq \epsilon\}$ with $0 < \epsilon < \rho(b)$. Also assume $\mathrm{ri}(\mathcal{C}) \cap \mathrm{range}(A)$ and $A^{-1}\mathcal{C}$ are not empty. Then*

$$\mathcal{C}' = \{A^*y \mid \langle b, y\rangle - \epsilon\rho^\circ(y) \geq 1\}. \tag{3.3.3}$$

PROOF. Since the arguments of ρ lie in $\mathcal{Y}$, we first consider the antipolar of $\mathcal{D} = A\mathcal{C} \subseteq \mathcal{Y}$ to establish the constraint $\langle b, y\rangle - \epsilon\rho^\circ(y) \geq 1$. Note that $y \in \mathcal{D}'$ is equivalent to $\langle Ax, y\rangle \geq 1$ for all

$Ax \in A\mathcal{C}$. This is again equivalent to

$$\langle b - Ax, y\rangle \leq \langle b, y\rangle - 1 \ \forall x \text{ such that } \rho(Ax - b) \leq \epsilon. \tag{3.3.4}$$

Next apply definition (3.2.14) for the antipolar ρ° and take the supremum over $u = \frac{b-Ax}{\epsilon}$, giving

$$\begin{aligned} \epsilon\rho^\circ(y) &= \epsilon \sup_u \left\{ \langle u, y\rangle \mid \rho(u) \leq 1,\ u = \tfrac{b-Ax}{\epsilon} \right\} \\ &= \sup_x \left\{ \langle b - Ax, y\rangle \mid \rho\left(\tfrac{b-Ax}{\epsilon}\right) \leq 1 \right\} \\ &= \sup_x \left\{ \langle b - Ax, y\rangle \mid \rho(b - Ax) \leq \epsilon \right\}, \end{aligned} \tag{3.3.5}$$

where the last equality uses the postive homogeneity of the gauge ρ. Using equation (3.3.5), we see that equation (3.3.4) is equivalent to the desired constraint $\epsilon\rho^\circ(y) \leq \langle b, y\rangle - 1$. Thus $\mathcal{D}' = \{y \mid \langle b, y\rangle - \epsilon\rho^\circ(y) \geq 1\}$.

Finally, note that $\mathcal{C}$ and $\mathcal{C}'$ both lie in $\mathcal{X}$, while $\mathcal{D} = A\mathcal{C}$ and $\mathcal{D}'$ lie in $\mathcal{Y}$. Then by Proposition 3.3.1, the antipolar $\mathcal{C}'$ has the form

$$\begin{aligned} \mathcal{C}' = \left(A^{-1}\mathcal{D}\right)' &= A^*\mathcal{D}' \\ &= \left\{A^*y \mid \langle b, y\rangle - \epsilon\rho^\circ(y) \geq 1\right\}. \end{aligned} \tag{3.3.6}$$

□

The next four propositions allow us to derive the pair P-GD-ineq (3.2.21) from the linearly-constrained pair (3.2.20) by transforming the P-ineq model into a linearly-constrained gauge model of the form P-lin (see the dependency map in Figure 3.1 for reference). This process uses an indicator function to embed the constraint set $\mathcal{C} = \{x \mid \rho(Ax - b) \leq \epsilon\}$ into the primal objective function, resulting in a P-lin model (3.2.20). Finally, we determine the gauge dual of the resulting model using polar relations and properties established in Section 3.2.

The first two propositions show that we may discuss the polar of a sum of gauges in terms of sets induced by Minkowski functions. This allows us to determine the polar of a sum of gauges in Proposition 3.3.5.

PROPOSITION 3.3.3. *Let $\mathcal{C} \subseteq \mathcal{X}$ be a closed, convex set containing the origin and $\kappa = \gamma_{\mathcal{C}}$ the Minkowski function induced by $\mathcal{C}$. Then κ is a gauge, $\mathcal{C} = \{x \mid \kappa(x) \leq 1\}$, and $\mathcal{C}$ is the unique closed, convex set containing the origin such that $\kappa = \gamma_{\mathcal{C}}$.*

PROOF. To verify that κ is a gauge, first note that positive homogeneity and $\kappa(0) = 0$ are direct results of κ being a Minkowski function. To show convexity, let $x, y \in \mathcal{X}$ and $0 \leq \alpha \leq 1$. Then $x \in \kappa(x)\mathcal{C}$ means $\alpha x \in \alpha\kappa(x)\mathcal{C}$, and likewise $(1-\alpha)y \in (1-\alpha)\kappa(y)\mathcal{C}$. Thus $\alpha x + (1-\alpha)y \in (\alpha\kappa(x) + (1-\alpha)\kappa(y))\mathcal{C}$. Then by the infimum of the Minkowski function, $\kappa(\alpha x + (1-\alpha)y) \leq \alpha\kappa(x) + (1-\alpha)\kappa(y)$ and κ is convex.

Additionally, $\kappa(x) \leq 1$ is equivalent to $x \in \mathcal{C}$, and thus $\mathcal{C} = \{x \mid \kappa(x) \leq 1\}$.

Finally, assume there is some closed, convex set $\mathcal{D} \subseteq \mathcal{X}$ such that $\kappa = \gamma_{\mathcal{D}}$. Then $\kappa(x) = \gamma_{\mathcal{C}}(x) = \gamma_{\mathcal{D}}(x) \leq 1$ is equivalent to x being in both $\mathcal{C}$ and $\mathcal{D}$, since both sets are closed and convex. Likewise, $\kappa(x) > 1$ indicates x is in neither set. Thus $\mathcal{C} = \mathcal{D}$.

□

PROPOSITION 3.3.4. *Let κ_1 and κ_2 be gauges. Let $\kappa(x_1, x_2) = \kappa_1(x_1) + \kappa_2(x_2)$, $\mathcal{C}_1 = \{z_1 \mid \kappa^\circ(z_1) \leq 1\}$, $\mathcal{C}_2 = \{z_2 \mid \kappa^\circ(z_2) \leq 1\}$, and $\mathcal{C} = \{(z_1, z_2) \mid \kappa^\circ(z_1, z_2) \leq 1\}$. Then κ and κ° are gauges, $\kappa^\circ = \gamma_{\mathcal{C}}$, and $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$.*

PROOF. Since κ is the sum of gauges, it is also convex, nonnegative, positively homogeneous, and zero at the origin, and thus a gauge. Then κ° is also a gauge and by Proposition 3.3.3, $\mathcal{C}$ is the unique set such that $\kappa = \gamma_{\mathcal{C}}$. If $(z_1, z_2) \in \mathcal{C}$, then $\langle x_1, z_1\rangle + \langle x_2, z_2\rangle \leq \kappa(x_1, x_2)\kappa^\circ(z_1, z_2) \leq \kappa_1(x_1) + \kappa_2(x_2)$ for all $x_1 \in \operatorname{dom} \kappa_1$ and $x_2 \in \operatorname{dom} \kappa_2$. In particular, if $x_2 = 0$ then $\langle x_1, z_1\rangle \leq \kappa_1(x_1)$ for all $x_1 \in \operatorname{dom} \kappa_1$, indicating $z_1 \in \mathcal{C}_1$. Similarly $z_2 \in \mathcal{C}_2$ and thus $\mathcal{C}_1 \times \mathcal{C}_2 \subseteq \mathcal{C}$. For the reverse inclusion, $(z_1, z_2) \in \mathcal{C}_1 \times \mathcal{C}_2$ means $\langle x_1, z_1\rangle \leq \kappa_1(x_1)$ for all $x_1 \in \operatorname{dom} \kappa_1$ and $\langle x_2, z_2\rangle \leq \kappa_2(x_2)$ for all $x_2 \in \operatorname{dom} \kappa_2$. Adding these inequalities, we have $\langle x_1, z_1\rangle + \langle x_2, z_2\rangle \leq \kappa_1(x_1) + \kappa_2(x_2)$ for all $x_1 \in \operatorname{dom} \kappa_1$ and $x_2 \in \operatorname{dom} \kappa_2$, and thus $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$. □

Given the two propositions above, we now show that the polar of a sum of gauges is the max of the set of polars.

PROPOSITION 3.3.5. *Let κ_1 and κ_2 be gauges. Then the sum $\kappa(x_1, x_2) := \kappa_1(x_1) + \kappa_2(x_2)$ is a guage and has the polar*

$$\kappa^\circ(z_1, z_2) = \max\{\kappa^\circ(z_1),\ \kappa_2^\circ(z_2)\}. \tag{3.3.7}$$

PROOF. Proposition 3.3.4 shows κ and its polar κ° are gauges. Setting $\mathcal{C}_1 = \{z_1 \mid \kappa^\circ(z_1) \leq 1\}$, $\mathcal{C}_2 = \{z_2 \mid \kappa^\circ(z_2) \leq 1\}$, and $\mathcal{C} = \{(z_1, z_2) \mid \kappa^\circ(z_1, z_2) \leq 1\}$, we may express the gauge polars as Minkowski functions $\kappa_1^\circ = \gamma_{\mathcal{C}_1}$, $\kappa_2^\circ = \gamma_{\mathcal{C}_2}$, and $\kappa^\circ = \gamma_{\mathcal{C}}$. Since $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}$ are closed, convex sets containing the origin, Proposition 3.3.3 tells us these are the unique sets defining κ_1°, κ_2°, and κ°. Additionally, Proposition 3.3.4 implies $\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2$ and $\kappa^\circ(z_1, z_2) = \gamma_{\mathcal{C}}(z_1, z_2) = \gamma_{\mathcal{C}_1 \times \mathcal{C}_2}(z_1, z_2)$.

Then we have

$$\begin{aligned}
\kappa^\circ(z_1, z_2) &= \gamma_{\mathcal{C}_1 \times \mathcal{C}_2}(z_1, z_2) \\
&= \inf\{\lambda \geq 0 \mid z_1 \in \lambda\mathcal{C}_1,\ z_2 \in \lambda\mathcal{C}_2\} \\
&= \max\{\inf\{\lambda \geq 0 \mid z_1 \in \lambda\mathcal{C}_1\},\ \inf\{\lambda \geq 0 \mid z_2 \in \lambda\mathcal{C}_2\}\} \\
&= \max\{\gamma_{\mathcal{C}_1}(z_1),\ \gamma_{\mathcal{C}_1}(z_2)\} \\
&= \max\{\kappa_1^\circ(z_1),\ \kappa_2^\circ(z_2)\}.
\end{aligned} \tag{3.3.8}$$

□

The following proposition establishes two equalities which allow us to compute the polar of an objective function which includes an indicator function. This strategy allows us to embed the inequality constraint $\rho(Ax - b) \leq \epsilon$ in the model P-ineq (3.2.21) into the objective function of (3.2.20) in the proof of Theorem 3.3.1, (ii).

PROPOSITION 3.3.6. *Let ρ be a gauge. Then for all $y \in \mathcal{X}$ and $\tau \geq 0$ the following equalities hold.*

(i) $(\delta_{\operatorname{epi}\rho})^\circ(y, \tau) = \delta_{(\operatorname{epi}\rho)^\circ}(y, \tau)$,

(ii) $\delta_{(\operatorname{epi}\rho)^\circ}(y, \tau) = \delta_{\operatorname{epi}(\rho^\circ)}(y, -\tau)$.

PROOF. To show (i) holds, consider the expansion of the expressions

$$\begin{aligned}(\delta_{\operatorname{epi}\rho})^{\circ}(y,\tau) &= \inf\{\mu>0 \mid \langle(x,\sigma),(y,\tau)\rangle \le \mu\delta_{\operatorname{epi}\rho}(x,\sigma) \ \ \forall(x,\sigma)\},\\ \delta_{(\operatorname{epi}\rho)^{\circ}}(y,\tau) &= \begin{cases}0 & \text{if } \langle(x,\sigma),(y,\tau)\rangle \le 1 \ \ \forall(x,\sigma)\in\operatorname{epi}\rho\\ +\infty & \text{else.}\end{cases}\end{aligned}$$

Note that the value of $\delta_{(\operatorname{epi}\rho)^{\circ}}(y,\tau)$ is either 0 or $+\infty$. If $\delta_{(\operatorname{epi}\rho)^{\circ}}(y,\tau)=0$, then $\langle(x,\sigma),(y,\tau)\rangle \le 1$ for all $(x,\sigma)\in\operatorname{epi}\rho$. Since ρ is a gauge, $(x,\sigma)\in\operatorname{epi}\rho$ is equivalent to $\rho(\alpha x)\le\alpha\sigma$ for all $\alpha>0$. Then $(\alpha x,\alpha\sigma)\in\operatorname{epi}\rho$ for all $\alpha>0$. Dividing the inequality by α and letting α approach infinity, we have

$$\langle(x,\sigma),(y,\tau)\rangle \le \lim_{\alpha\to+\infty}\frac{1}{\alpha}=0 \ \text{ for all } (x,\alpha)\in\operatorname{epi}\rho.$$

Thus $(\delta_{\operatorname{epi}\rho})^{\circ}(y,\tau)=0$. On the other hand, if $\delta_{(\operatorname{epi}\rho)^{\circ}}(y,\tau)=+\infty$ then there is some $(x,\sigma)\in\operatorname{epi}\rho$ with $\langle(x,\sigma),(y,\tau)\rangle>1$. Then $\delta_{\operatorname{epi}\rho}(x,\sigma)=0$ and there is no $\mu>0$ such that $\langle(x,\sigma),(y,\tau)\rangle \le \mu\delta_{\operatorname{epi}\rho}(x,\sigma)$. Thus $(\delta_{\operatorname{epi}\rho})^{\circ}(y,\tau)=+\infty$.

To show (ii) holds, we have the following set of equivalences

$$\begin{aligned}(y,\tau)\in(\operatorname{epi}\rho)^{\circ} &\iff \langle(x,\sigma),(y,\tau)\rangle\le 1 \ \ \forall(x,\sigma)\in\operatorname{epi}(\rho)\\ &\iff \langle(x,\sigma),(y,\tau)\rangle\le 0 \ \ \forall x \text{ and } \rho(x)\le\sigma\\ &\iff \langle x,y\rangle\le-\tau\rho(x) \ \ \forall x\\ &\iff \rho^{\circ}(y)=\inf\{\mu>0\mid\langle x,y\rangle\le\mu\rho(x) \ \ \forall x\}\le-\tau\\ &\iff (y,-\tau)\in\operatorname{epi}(\rho^{\circ}).\end{aligned}$$

Here, the second equivalence was established in the proof of (i) and the third comes from setting σ as the minimal value $\sigma=\rho(x)$. Thus $\delta_{(\operatorname{epi}\rho)^{\circ}}(y,\tau)=\delta_{\operatorname{epi}(\rho^{\circ})}(y,-\tau)$. □

3.3.2. Gauge Duality Theorem

Given the supporting propositions established in Section 3.3.1, we are now prepared to prove the following theorem.

THEOREM 3.3.1. *Let P-ineq and GD-ineq be the inequality-constrained pair of models from (3.2.21). Also let* $\mathcal{C}=\{x \mid \rho(Ax-b)\le\epsilon\}$ *with* $0<\epsilon<\rho(b)$ *and assume* $\operatorname{ri}(\mathcal{C})\cap\operatorname{range}(A)$ *and*

$A^{-1}\mathcal{C}$ are not empty. Then the model GD-ineq is the gauge dual of the primal P-ineq based on the gauge duality of

(i) the nonlinear gauge dual pair (3.2.2), and

(ii) the linear gauge dual pair (3.2.20).

PROOF. Item (i) is a direct result of Proposition 3.3.2, which gives the antipolar set $\mathcal{C}' = \{A^*y \mid \langle b, y\rangle - \epsilon\rho^\circ(y) \geq 1\}$. As a result, the argument of the GD-nonlin objective κ° is A^*y and GD-ineq is equivalent to GD-nonlin.

To prove item (ii), we must first express P-ineq as a linear gauge model of the form P-lin. Define the function $\phi(x, r, \sigma) = \kappa(x) + \delta_{\text{epi}\,\rho}(r, \sigma)$. Since the epigraph of a gauge is closed under positive scaling, the indicator function $\delta_{\text{epi}\,\rho}$ is a gauge. By Proposition 3.3.4, ϕ is a gauge, and thus P-ineq is equivalent to the linear gauge model

$$\begin{aligned} \min_{x,r,\sigma} \quad & \phi(x, r, \sigma) \\ \text{s.t.} \quad & r = b - Ax \\ & \sigma = \epsilon. \end{aligned} \tag{3.3.9}$$

To combine these linear constraints, define the following extended matrix and vectors

$$\overline{A} = \begin{bmatrix} A & I & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \overline{x} = \begin{bmatrix} x \\ r \\ \sigma \end{bmatrix}, \quad \overline{b} = \begin{bmatrix} b \\ \epsilon \end{bmatrix}, \tag{3.3.10}$$

and define the spaces $\overline{\mathcal{X}} = \mathcal{X} \times \mathcal{Y} \times \{\mathbb{R} \cup +\infty\}$ and $\overline{\mathcal{Y}} = \mathcal{Y} \times \{\mathbb{R} \cup +\infty\}$.

Then (3.3.9) has the linear form equivalent to P-lin

$$\min_{\overline{x} \in \overline{\mathcal{X}}} \phi(\overline{x}) \quad \text{s.t.} \quad \overline{A}\overline{x} = \overline{b}. \tag{3.3.11}$$

This model has the following gauge dual per (3.2.20)

$$\min_{\overline{y} \in \overline{\mathcal{Y}}} \phi^\circ(\overline{A}^*\overline{y}) \quad \text{s.t.} \quad \langle \overline{b}, \overline{y}\rangle = 1, \tag{3.3.12}$$

where $\overline{A}^*\overline{y} = (A^*y, y, \tau)$ and $\langle \overline{b}, \overline{y}\rangle = \langle b, y\rangle + \sigma\tau$.

Taking the polar of ϕ, we have

$$\begin{aligned}\phi^\circ(A^*y, y, \tau) &= \max\{\kappa^\circ(A^*y),\ (\delta_{\operatorname{epi}\rho})^\circ(y,\tau)\} \\ &= \max\{\kappa^\circ(A^*y),\ \delta_{(\operatorname{epi}\rho)^\circ}(y,\tau)\} \\ &= \kappa^\circ(A^*y) + \delta_{(\operatorname{epi}\rho)^\circ}(y,\tau) \\ &= \kappa^\circ(A^*y) + \delta_{\operatorname{epi}(\rho^\circ)}(y,-\tau).\end{aligned} \tag{3.3.13}$$

The first equality is a result of Proposition 3.3.5. The second and last equalities are results of Proposition 3.3.6, (i) and (ii), respectively. And the third equality is a consequence of the indicator function mapping to $\{0, +\infty\}$.

The indicator function $\delta_{\operatorname{epi}(\rho^\circ)}(y, -\tau)$ corresponds to the constraint $\rho^\circ(y) \le -\tau$. This constraint and the equality constraint $\langle b, y\rangle + \sigma\tau = 1$ may be combined as

$$\langle b, y\rangle - \sigma\rho^\circ(y) \ge \langle b, y\rangle + \sigma\tau = 1.$$

Thus we eliminate τ and recover GD-ineq. □

3.3.3. Weak Duality, Strong Duality, and Optimality Conditions

With the gauge duality of the pair of models P-GD-ineq (3.2.21) established in Section 3.3.2, we now proceed to establish weak duality, strong duality, and optimality conditions for this pair. These properties will play a central role in developing the GDD algorithm for optimizing the PLGD model (3.2.1), allowing us to develop termination conditions as well as a recovery method for the primal signal x from a dual iterate y. The weak duality property of P-GD-ineq (3.2.21) is a straightforward consequence of the definition of the polar of a function and inequality constraints in this model.

PROPOSITION 3.3.7. (weak duality) *Assume the primal model P-ineq (3.2.21) is feasible and $0 \le \epsilon < \rho(b)$. Then for all primal and gauge dual feasible pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ we have*

$$\kappa(x)\kappa^\circ(\mathcal{A}^*y) \ge 1. \tag{3.3.14}$$

PROOF. Since x and y are feasible for P-GD-ineq (3.2.21), we have

$$\begin{aligned} 1 &\leq \langle b, y\rangle - \epsilon\rho^\circ(y) \\ &\leq \langle b, y\rangle - \rho(b - Ax)\rho^\circ(y) \\ &\leq \langle b, y\rangle - \langle b - Ax, y\rangle \\ &= \langle x, A^*y\rangle \\ &\leq \kappa(x)\kappa^\circ(A^*y). \end{aligned} \tag{3.3.15}$$

Here, the first and second inequalities are due to GD-ineq and P-ineq feasibility, respectively. The third and fourth inequalities are a consequence of polar functions. □

In order to show strong duality holds for the pair P-GD-ineq (3.2.21), we add the assumption that ρ° is continuous and use the strong duality properties of the Lagrange primal and dual pair (see [**51**, Section 28])

$$\begin{array}{llll} \min\limits_{x\in\mathcal{X}} & \kappa(x) & \max\limits_{y\in\mathcal{Y}} & \langle b, y\rangle - \epsilon\rho^\circ(y) \\ \text{(P-ineq)} \quad \text{s.t.} & \rho(Ax - b) \leq \epsilon \qquad \text{(LD-ineq)} \quad \text{s.t.} & \kappa^\circ(A^*y) \leq 1. \end{array} \tag{3.3.16}$$

This proof strategy identifies a sequence of Lagrange dual feasible vectors which may be rescaled to create to a gauge dual feasible sequence. To guarantee the limit of this sequence exists, we require that ρ° is continuous.

PROPOSITION 3.3.8. (strong duality) *Assume the primal model P-ineq (3.2.21) is feasible, ρ° is continuous, and $0 \leq \epsilon < \rho(b)$. Then P-ineq admits an optimal variable $x_\star$ and*

$$\kappa(x_\star)\nu_{GD} = 1 \tag{3.3.17}$$

where ν_{GD} is the optimal GD-ineq value

$$\nu_{GD} := \inf_{\langle b,y\rangle - \epsilon\rho^\circ(y) \geq 1} \kappa^\circ(A^*y). \tag{3.3.18}$$

Additionally, if P-ineq is strictly feasible then P-ineq and GD-ineq admit an optimal pair $(x_\star, y_\star) \in \mathcal{X} \times \mathcal{Y}$ and

$$\kappa(x_\star)\kappa^\circ(\mathcal{A}^*y_\star) = 1, \tag{3.3.19}$$

where $y_\star$ is a rescaling of the optimal LD-ineq variable $\hat{y}$ such that

$$y_\star = \frac{\hat{y}}{\langle b, \hat{y}\rangle - \epsilon\rho^\circ(\hat{y})}. \tag{3.3.20}$$

PROOF. Since P-ineq is feasible, Lagrange duality implies that LD-ineq is bounded above. Additionally, since LD-ineq is strictly feasible for $y = 0$ (i.e., $\kappa^\circ(0) = 0 < 1$ and 0 is in the relative interior of the Lagrange dual feasible set), [**51**, Theorem 28.2] indicates that P-ineq admits an optimal variable $\hat{x}$ and LD-ineq has finite optimal objective value

$$\nu_{LD} := \sup_{\kappa^\circ(A^*y)\leq 1} \langle b, y\rangle - \epsilon\rho^\circ(y) < +\infty.$$

Furthermore, [**51**, Theorem 28.4] tells us this pair has zero Lagrange duality gap.

Since $\hat{x}$ is optimal for P-ineq, $x_\star = \hat{x}$ is our desired primal solution. By strong Lagrange duality $\nu_{LD} = \kappa(x_\star)$, and by weak gauge duality (Proposition 3.3.7) this value is strictly greater than zero. Let $\{y_i\}$ be a LD-ineq feasible sequence such that $\langle b, y_i\rangle - \epsilon\rho^\circ(y_i) \to \nu_{LD} > 0$. Since ρ° is continuous, there exists a subsequence $\{y_{i_j}\}$ such that $\langle b, y_{i_j}\rangle - \epsilon\rho^\circ(y_{i_j})$ is bounded above zero for all j. Rescaling this sequence such that

$$\mathbf{y}_j = \frac{y_{i_j}}{\langle b, y_{i_j}\rangle - \epsilon\rho^\circ(y_{i_j})},$$

we have $\langle b, \mathbf{y}_j\rangle - \epsilon\rho^\circ(\mathbf{y}_j) = 1$ for all j, and thus $\{\mathbf{y}_j\}$ is a GD-ineq feasible sequence. Then we have

$$\begin{aligned}
\nu_{GD} &\leq \lim_{j\to\infty} \kappa^\circ(A^*\mathbf{y}_j) \\
&= \lim_{j\to\infty} \tfrac{1}{\langle b, y_{i_j}\rangle - \epsilon\rho^\circ(y_{i_j})}\kappa^\circ(A^*y_{i_j}) \\
&= \tfrac{1}{\kappa(x_\star)} \cdot \lim_{j\to\infty} \kappa^\circ(A^*y_{i_j}) \leq \tfrac{1}{\kappa(x_\star)},
\end{aligned}$$

where the last inequality is due to $\{y_{i_j}\}$ being LD-ineq feasible (i.e., $\kappa^\circ(A^*y_{i_j}) \leq 1$ for all j). And by weak gauge duality (Proposition 3.3.7) we have $\nu_{GD} \geq \frac{1}{\kappa(x_\star)}$, giving (3.3.17).

If P-ineq is strictly feasible, then by [**51**, Theorem 28.2] LD-ineq will admit a finite optimal solution $\hat{y}$. This variable may be rescaled to obtain the optimal GD-ineq variable (3.3.20) and the same subsequence argument gives (3.3.19).

□

We close this section by establishing the optimality conditions of the pair P-GD-ineq (3.2.21) which are primarily a consequence of the strong and weak duality results above.

PROPOSITION 3.3.9. (optimality conditions) *If the primal model P-ineq (3.2.21) is feasible, ρ° is continuous, and $0 \leq \epsilon < \rho(b)$, then the pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ is primal-dual optimal if and only if the following conditions hold:*

$$\rho(Ax - b) = \epsilon \qquad \text{primal feasibility} \tag{3.3.21a}$$

$$\langle b, y\rangle - \epsilon\rho^\circ(y) = 1 \qquad \text{dual feasibility} \tag{3.3.21b}$$

$$\langle b - Ax, y\rangle = \rho(b - Ax)\rho^\circ(y) \qquad \text{complementarity} \tag{3.3.21c}$$

$$\langle x, A^*y\rangle = \kappa(x)\kappa^\circ(A^*y) = 1 \qquad \text{strong duality} \tag{3.3.21d}$$

$$\frac{1}{\rho^\circ(y)} y \in \partial\rho(\cdot)(b - Ax) \qquad \text{primal subdifferential} \tag{3.3.21e}$$

$$\frac{1}{\epsilon}(b - Ax) \in \partial\rho^\circ(\cdot)(y) \qquad \text{dual subdifferential} \tag{3.3.21f}$$

PROOF. If these conditions hold, then (x, y) are primal-dual feasible and have zero duality gap (i.e., $\kappa(x)\kappa^\circ(A^*y) = 1$). Thus by strong duality (Proposition 3.3.8) (x, y) are an optimal pair.

Likewise, if (x, y) are an optimal pair, then strong duality implies $\kappa(x)\kappa^\circ(A^*y) = 1$. Thus the set of inequalities in the weak duality proof (3.3.15) are tight and the first four conditions hold.

The last two conditions are a consequence of the first four conditions and polar relations. To show the primal subdifferential condition, note that for all z we have $\langle z, y\rangle \leq \rho(z)\rho^\circ(y)$. The primal feasibility and complementarity conditions give $\langle b - Ax, y\rangle = \epsilon\rho^\circ(y)$. Then for all z we have

$$\langle z - (b - Ax), y\rangle \leq \rho(z)\rho^\circ(y) - \epsilon\rho^\circ(y).$$

Replacing $\epsilon = \rho(b - Ax)$ and dividing by $\rho^\circ(y)$, we have for all z

$$\rho(z) \geq \rho(b - Ax) + \langle z - (b - Ax), y/\rho^\circ(y)\rangle,$$

and thus $\frac{1}{\rho^\circ(y)} y \in \partial\rho(\cdot)(b - Ax)$.

For the dual subdifferential condition, we have $\langle b - Ax, z\rangle \leq \rho(b - Ax)\rho^\circ(z) = \epsilon\rho^\circ(z)$ for all z. Subtracting $\langle b - Ax, y\rangle = \epsilon\rho^\circ(y)$ gives $\langle b - Ax, z - y\rangle \leq \epsilon\rho^\circ(z) - \epsilon\rho^\circ(y)$ for all z, or

$$\rho^\circ(z) \geq \rho^\circ(y) + \langle (b - Ax)/\epsilon, z - y\rangle,$$

and thus $\frac{1}{\epsilon}(b - Ax) \in \partial\rho^\circ(\cdot)(y)$.

□

3.4. Theory Applied to the PLGD Model

This section establishes optimality conditions and a primal signal recovery strategy for the PLGD model (3.2.1), two essential tools for developing the Gauge Dual Descent (GDD) algorithm presented in Chapter 4. We see first that the duality of the PLP-PLGD pair (3.2.1) is a direct result of the duality previously established for the pair P-GD-ineq (3.2.21). Thus Proposition 3.3.9 applies to the PLGD model, allowing us to develop termination conditions and a primal signal recovery method for any given PLGD optimization method. We also present a more general optimality condition which applies to the GDD algorithm.

We begin by establishing the duality of the PLP-PLGD pair (3.2.1) based on the duality established in Theorem 3.3.1. In the PLP model, the linear operator $\mathcal{A}$ is defined as a map from the space of Hermitian matrices $\mathcal{H}^n$ to $\mathbb{R}^m$ (see (2.4.1) for details). We may pass the PLP constraint $X \succeq 0$ into the objective by setting $\kappa(X) := ||X||_1 + \delta_{(\cdot)\succeq 0}(X)$. The polar of κ has the form

$$\kappa^\circ(Z) = \inf\left\{\mu > 0 \mid \langle X, Z\rangle \leq \mu||X||_1 \ \forall X \succeq 0\right\},$$

which can be simplified by considering two cases. If $\lambda_1(Z)$ is positive then $\kappa^\circ(Z)$ operates as the dual norm of the Schatten 1-norm restricted the positive eigenspace of Z. Otherwise, if $Z \preceq 0$ then $\kappa^\circ(Z)$ is zero, giving

$$\kappa^\circ(Z) = \begin{cases} \lambda_1(Z) & \text{if } Z \succeq 0 \\ 0 & \text{else.} \end{cases} \tag{3.4.1}$$

We may also set $\rho(y) := ||y||_2$, which has the polar (or dual norm) $\rho^\circ(y) = ||y||_* = ||y||_2$. Thus PLP has the form of P-ineq (3.2.21)

$$\begin{aligned} \min_{X \in \mathcal{H}^n} \quad & \kappa(X) = ||X||_1 + \delta_{(\cdot)\succeq 0}(X) \\ \text{s.t.} \quad & \rho(\mathcal{A}(X) - b) = ||\mathcal{A}(X) - b||_2 \leq \epsilon. \end{aligned}$$

Then by Theorem 3.3.1, the GD-ineq model has the objective $\kappa^\circ(\mathcal{A}^* y)$ and constraint $\langle b, y\rangle - \epsilon||y||_2 \geq 1$. We may further simplify κ° by noting that $\kappa(X)$ is strictly positive and finite for all feasible $X \in \mathcal{H}^n$. Then by weak duality (Proposition 3.3.7) $\kappa^\circ(\mathcal{A}^* y)$ is likewise strictly positive and finite for all feasible y. Thus $\kappa^\circ(\mathcal{A}^* y) = \lambda_1(\mathcal{A}^* y)$ and we recover the PLGD model from the GD-ineq model.

Next we see that the PLP-PLGD (3.2.1) optimality conditions and primal recovery property are a direct result of the propositions in Section 3.3 applied to primal-gauge dual space $\mathcal{X} \times \mathcal{Y} = \mathcal{H}^n \times \mathbb{R}^m$. These two corollaries rely on the von Neumann trace inequality, which states that for all $n \times n$ complex matrices A and B,

$$\langle A, B\rangle \leq \sum_{i=1}^{n} \sigma_i(A)\sigma_i(B), \tag{3.4.2}$$

and equality holds if and only if A and B are simultaneously diagonalizable [**33**].

Corollary 3.4.1. (PLP-PLGD optimality conditions) *If PLP in (3.2.1) is feasible and* $0 \leq \epsilon < ||b||_2$, *then* $(X, y) \in \mathcal{H}^n \times \mathbb{R}^m$ *is primal-dual optimal if and only if the following conditions hold*

(a) $X \succeq 0$ *and* $||\mathcal{A}(X) - b||_2 = \epsilon$,

(b) $\langle b, y\rangle - \epsilon||y||_2 = 1$,

(c) $\langle b - \mathcal{A}(X), y\rangle = ||\mathcal{A}(X) - b||_2 \cdot ||y||_2$,

(d) $\langle X, \mathcal{A}^* y\rangle = ||X||_1 \cdot \lambda_1(\mathcal{A}^* y) = 1$,

(e) $\lambda_i(X) \cdot (\lambda_1(\mathcal{A}^* y) - \lambda_i(\mathcal{A}^* y)) = 0$ *for all* $i = 1, \ldots, n$;

(f) X *and* $\mathcal{A}^* y$ *are simultaneously diagonalizable.*

Proof. Since $\rho^\circ(\cdot) = ||\cdot||_2$ is continuous, we may invoke Proposition 3.3.9. The first four conditions are identical to those discussed in Proposition 3.3.9 for the more general P-GD-ineq pair (3.2.21). Thus these conditions holding is equivalent to (X, y) being primal-dual optimal.

Then we must simply show the first four conditions imply the last two. Applying the von Neumann trace inequality to the matrices X and $\mathcal{A}^*y$, we have

$$1 = \langle X, \mathcal{A}^*y \rangle \leq \sum_{i=1}^{n} \sigma_i(X)\sigma_i(\mathcal{A}^*y) \leq ||X||_1 \cdot \lambda_1(\mathcal{A}^*y) = 1. \tag{3.4.3}$$

Replacing the two inequalities in (3.4.3) with equalities, we see that the matrices X and $\mathcal{A}^*y$ are simultaneously diagonalizable (by the von Neumann trace inequality), and for all i, if $\lambda_i(X) > 0$ then we must have $\lambda_i(\mathcal{A}^*y) = \lambda_1(\mathcal{A}^*y)$. □

In addition to Corollary 3.4.1, we now establish a more general first-order optimality condition for convex optimization which applies to the PLGD model (3.2.1). Recall that convex optimization is the minimization of any convex function f over a convex set $\mathcal{C}$, and a first-order method is any iteration $y_+ = \Pi_{\mathcal{C}}(y - \alpha g)$, where g is some descent direction in $\partial f(y)$ and $\alpha > 0$ is a steplength chosen with some linesearch policy. In Chapter 4 we will develop a first-order method for optimizing (3.2.1), but for now consider any first-order PLGD method for $f(y) = \lambda_1(\mathcal{A}^*y)$ and $\mathcal{C} = \{y \in \mathbb{R}^m \mid \langle b, y \rangle - \epsilon||y||_2 \geq 1\}$.

The following result can be viewed as a fixed-point optimality condition, where $y_\star$ is optimal if $y_\star = \Pi_{\mathcal{C}}(y_\star - \alpha g)$ for some $g \in \partial f(y_\star)$ and all $\alpha > 0$. To prove this result, we frame (3.2.1) as an unconstrained convex optimization problem by defining $F(y) := f(y) + \delta_{\mathcal{C}}(y)$, where $\delta_{\mathcal{C}}(y)$ is the indicator function (1.0.8) of $\mathcal{C}$. Then optimizing (3.2.1) is equivalent to minimizing the unconstrained function F over $\mathbb{R}^m$.

Proposition 3.4.2. *Let f be a proper convex function over a finite-dimensional Euclidean space $\mathcal{Y}$ and $\mathcal{C} \subseteq \mathcal{Y}$ a convex set. Also assume* $\mathrm{ri}(\mathrm{dom}(f)) \cap \mathrm{ri}(\mathcal{C})$ *is not empty. Then $y_\star$ is optimal for the minimization problem*

$$\min_y F(y) := f(y) + \delta_{\mathcal{C}}(y) \tag{3.4.4}$$

if $y_\star = \Pi_{\mathcal{C}}(y_\star - \alpha g)$ for some nonzero $g \in \partial f(y_\star)$ and all $\alpha > 0$.

PROOF. We begin by demonstrating the first-order optimality condition for unconstrained proper convex functions. A variable $y_\star$ is optimal for (3.4.4) if and only if

$$F(y_\star) = F(y_\star) + 0^T(y - y_\star) \leq F(y) \quad \forall y \in \mathcal{Y}. \tag{3.4.5}$$

Then zero is in the subdifferential of F at $y_\star$, i.e.,

$$y_\star \text{ is optimal if and only if } 0 \in \partial F(y_\star). \tag{3.4.6}$$

Thus it suffices to show $0 \in \partial F(y_\star)$. Note that $-g$ is not a descent direction for f, since $f(y_\star) = f(\Pi_{\mathcal{C}}(y_\star - \alpha g))$ for all $\alpha > 0$. Then $-g$ is in the normal cone (1.0.9) of $\mathcal{C}$ at $y_\star$. Additionally, the normal cone $N_{\mathcal{C}}(y_\star)$ is equivalent to the subdifferential of the indicator function $\partial\delta_{\mathcal{C}}(y_\star)$, since

$$\begin{aligned} \partial\delta_{\mathcal{C}}(y_\star) &= \{g \mid \delta_{\mathcal{C}}(y) \geq \delta_{\mathcal{C}}(y_\star) + \langle g, y - y_\star\rangle \ \ \forall y\} \\ &= \{g \mid 0 \geq \langle g, y - y_\star\rangle \ \ \forall y \in \mathcal{C}\} \\ &= N_{\mathcal{C}}(y_\star). \end{aligned} \tag{3.4.7}$$

Finally, since f and $\mathcal{C}$ are convex and $\text{ri}(\text{dom}(f)) \cap \text{ri}(\mathcal{C}) \neq \emptyset$, we have the set equality $\partial F(y_\star) = \partial f(y_\star) + \partial\delta_{\mathcal{C}}(y_\star)$ [**51**, Theorem 23.8]. Then $g \in \partial f(y_\star)$ and $-g \in \partial\delta_{\mathcal{C}}(y_\star)$ imply that $0 \in \partial F(y_\star)$, and thus $y_\star$ is optimal for (3.4.4).

□

Thus $y_\star = \Pi_{\mathcal{C}}(y_\star - \alpha g)$ is another optimality condition for the PLGD model (3.2.1). Next we apply Corollary 3.4.1 to establish a primal recovery strategy for (3.2.1).

COROLLARY 3.4.3. (PLP-PLGD primal recovery) *Assume the optimality conditions of Corollary 3.4.1 hold. Let $y_\star \in \mathbb{R}^m$ be an arbitrary optimal solution for the PLP-PLGD pair (3.2.1), $U \in \mathbb{C}^{n\times r}$ the eigenvectors corresponding to the algebraically largest eigenvalue λ_1 of $\mathcal{A}^* y_\star$, and r the multiplicity of λ_1. Then $X_\star \in \mathbb{C}^{n\times n}$ is a solution to (3.2.1) if and only if there exists an $r \times r$ matrix $S \succeq 0$ such that*

(a) $X_\star = USU^*$,
(b) $b - \mathcal{A}(X_\star) \in \epsilon\partial||\cdot||_2(y_\star)$.

PROOF. If $X_\star$ is a solution to (3.2.1), then (e) of Corollary 3.4.1 implies that rank of $X_\star$ is no larger than the multiplicity r of $\lambda_1(\mathcal{A}^* y_\star)$, and (f) implies that $X_\star$ has the form $X_\star = USU^*$ for an $r \times r$ matrix $S \succeq 0$. Also, the dual subdifferential condition of Proposition 3.3.9 implies that (b) holds.

Conversely, if (a) and (b) hold then it can be shown that the optimality conditions (a)-(d) of Corollary 3.4.1 also hold. The optimality of $y_\star$ gives $\langle b, y_\star \rangle - \epsilon ||y_\star||_2 = 1$. Conditions (a) and (c) may be derived from definitions of the dual norm $\rho^\circ(\cdot) = ||\cdot||_2$, which we temporarily denote as $||\cdot||_*$ for clarity. The subdifferential of the dual norm (3.2.7) is the set of maximizers which attain the dual norm value, that is $\partial ||\cdot||_*(y) = \{x \mid ||y||_* = \langle x, y \rangle\}$. Then we have

$$\langle b - \mathcal{A}(X_\star), y_\star \rangle = \epsilon ||y_\star||_*. \tag{3.4.8}$$

Like the polar (3.2.5), the dual norm $||\cdot||_*$ may also be defined as the function which most tightly satisfies the Cauchy-Schwarz inequality $|\langle x, y \rangle| \leq ||x|| \cdot ||y||_*$. This definition gives

$$\langle b - \mathcal{A}(X_\star), y_\star \rangle = ||b - \mathcal{A}(X_\star)|| \cdot ||y_\star||_*. \tag{3.4.9}$$

Equations (3.4.8) and (3.4.9) prove the optimality conditions (a) and (c).

Finally, we establish optimality condition (d) with the equality

$$1 = \langle X_\star, \mathcal{A}^* y_\star \rangle = \sum_{i=1}^{n} \sigma_i(X_\star) \sigma_i(\mathcal{A}^* y_\star) = ||X_\star||_1 \cdot \lambda_1(\mathcal{A}^* y_\star).$$

In this expression, the first equality is a result of optimality conditions (a)-(c), which cause the first four lines of (3.3.15) to hold with equality. The second equality is a consequence of the von Neumann trace inequality (3.4.2) holding tightly. The columns of $U \in \mathbb{C}^{n \times r}$ span the eigenspace of the algebraically largest eigenvalue λ_1 of $\mathcal{A}^* y_\star$. Since the diagonalization of $X_\star = USU^*$ only requires a unitary transformation of U, this transformation will not affect the eigenspace of $\mathcal{A}^* y_\star$. Thus $X_\star$ and $\mathcal{A}^* y_\star$ are simultaneously diagonalizable. Finally, the third equality is a result of $X_\star$ having rank at most r and λ_1 having multiplicity r.

Thus optimality conditions (a)-(d) hold, and $X_\star$ is a solution to (3.2.1).

□

Corollary 3.4.3 allows us to recover a signal x by using the dual variable y to denoise the noisy observation $b = \mathbf{b} + \eta$. Assuming $y \neq 0$, the 2-norm used in the PLP-PLGD pair (3.2.1) gives $\partial||\cdot||(y) = \nabla||\cdot||_2(y) = y/||y||_2$. If the lifted true signal $X_\star = \mathbf{x}\mathbf{x}^*$ is in the set of optimal matrices, then $\mathcal{A}(X_\star) = \mathbf{b}$ and Corollary 3.4.3 indicates that the corresponding dual optimal variable $y_\star$ will have the relation

$$\eta = (\mathbf{b} + \eta) - \mathbf{b} = b - \mathcal{A}(X_\star) = \epsilon\nabla||\cdot||_2(y_\star) = \epsilon\frac{y_\star}{||y_\star||_2}. \tag{3.4.10}$$

Thus the noise term η is proportional to one of the optimal dual variables.

Unfortunately, in the case of noisy phase retrieval there is no guarantee that the lifted true signal $X = \mathbf{x}\mathbf{x}^*$ will be an optimal matrix for the PLP-PLGD pair (3.2.1). Instead, the gauge dual variable y can be viewed as a parameter which learns the noise term η with some degree of inaccuracy. As we will see in Section 5.3, the tendency of y to learn the noise term persists even when the rank of an optimal PLP-PLGD matrix $X_\star$ is greater than one and a given first-order method is unable to converge to $y_\star$.

Given the optimality and primal recovery properties established above, we now proceed to develop the Gauge Dual Descent (GDD) algorithm, a first-order algorithm for optimizing the PLGD model (3.2.1) and recovering a primal signal x from the PLGD solution y.

CHAPTER 4

The Gauge Dual Descent Algorithm for the PLGD Model

4.1. Introduction

In this chapter we present the Gauge Dual Descent (GDD) algorithm, a projected gradient descent algorithm for optimizing the PLGD model (3.2.1). Section 4.2 develops the computational steps necessary for the GDD algorithm. We then present the GDD algorithm and discuss specific implementation details. Section 4.3 demonstrates the effectiveness of the GDD algorithm for noiseless phase retrieval problems. We close with a summary of the challenges to the GDD algorithm which are addressed in the remainder of this dissertation.

The GDD algorithm presented in this chapter was first established in [**27**]. As we will see in Section 5.3, the challenges posed by noisy phase retrieval are intrinsic to the PLGD model (3.2.1) itself, and independent of the choice of gradient-based descent method. Thus, due to the effectiveness of the GDD algorithm for noiseless phase retrieval and the existence of a comprehensive, specialized software package by the authors of [**27**], we use the GDD algorithm to examine the behavior of gradient-based descent methods for optimizing (3.2.1). All implementation details in this chapter are identical to the original software package unless otherwise noted.

4.2. The Gauge Dual Descent Algorithm

We begin this section by examining the features of the PLGD model (3.2.1) which are essential for developing a projected gradient descent algorithm. Recall that projected gradient descent algorithms have a basic iterate update $y_+ = \Pi_{\mathcal{C}}(y - \alpha g)$, where $-g$ is a descent direction based on first-order information, α is a steplength determined by some policy, and $\Pi_{\mathcal{C}}$ is projection onto the feasible region $\mathcal{C}$. Since we are primarily concerned with large-scale phase retrieval problems, the descent direction must be constructed using only first-order information, i.e., the gradient or a subgradient of the objective function. Additionally, each objective function evalution $\lambda_1(\mathcal{A}^* y)$

requires the solution of a costly eigenvalue problem, so its computation should be kept to a minimum. Finally, the PLGD model (3.2.1) is a convex problem, having both convex objective function $\lambda_1(\mathcal{A}^*y)$ and convex constraint domain $\mathcal{C} = \{y \in \mathbb{R}^m \mid \langle b, y\rangle - \epsilon||y||_2 \geq 1\}$. Therefore, the projected gradient descent algorithm should take advantage of this convexity and determine the steplength α using a linesearch with minimal backtracking (or evaluations of $\lambda_1(\mathcal{A}^*y)$) and guaranteed convergence. Thus our method of choice for optimizing the PLGD model (3.2.1) is a projected gradient descent method with adaptive steplength based on the local differentiability of $\lambda_1(\mathcal{A}^*y)$. Note that one could also use a quasi-Newton or spectral bundle method to optimize the PLGD model.

Construction of the GDD algorithm requires the following sequence of computational steps. First, a descent direction is chosen from the gradient or subdifferential (1.0.17) of the objective function. Next, an initial steplength is computed and used to initialize a linesearch (backtracking) method for determining the dual update y_+. The linesearch and update both require a method for projecting onto the feasible region $\mathcal{C}$. A primal recovery step is used to recover the primal signal update x_+ from the dual update y_+. Finally, a primal refinement step and a dual refinement step are performed, which were shown by the authors of [**27**] to accelerate convergence of the GDD algorithm.

To determine the descent vector, we consider the subdifferential of the function $\lambda_1(\mathcal{A}^*y)$

$$\partial\lambda_1(\mathcal{A}^*y) = \{\mathcal{A}(V_1TV_1^*) \mid T \succeq 0,\ \mathrm{tr}(T) = 1\}, \tag{4.2.1}$$

where $V_1 \in \mathbb{C}^{n\times r_1}$ are the eigenvectors corresponding to the algebraically largest eigenvalue λ_1 of $\mathcal{A}^*y$ and r_1 is the multiplicity of λ_1 [**48**, Section 6.7]. Note that evaluation of the objective function $\lambda_1(\mathcal{A}^*y)$ likewise returns the (sub)gradient of this function as a product of the eigenvector(s) corresponding to λ_1. If the eigenvalue λ_1 has separation from the second eigenvalue λ_2, then the function $\lambda_1(\mathcal{A}^*y)$ is differentiable at y and we have the descent vector

$$g = \nabla\lambda_1(\mathcal{A}^*y) = \mathcal{A}(v_1v_1^*). \tag{4.2.2}$$

However, if the multiplicity of λ_1 is greater than one, then $\lambda_1(\mathcal{A}^*y)$ is nondifferentiable and we may select any $g \in \partial\lambda_1(\mathcal{A}^*y)$. In practice, we consider the function $\lambda_1(\mathcal{A}^*y)$ differentiable if

$$\frac{\lambda_1 - \lambda_2}{\lambda_1} \geq \text{tol}_{\text{diff}}, \tag{4.2.3}$$

for some tolerance tol_{diff}.

Next, the descent vector g is used to identify an initial steplength and perform a linesearch. If $\lambda_1(\mathcal{A}^*y)$ is differentiable, then we take an initial step with Barzilai-Borwein steplength [**4**]

$$\alpha = \frac{\langle dy, dy\rangle}{\langle dy, dg\rangle}, \tag{4.2.4}$$

where $dy = y_k - y_{k-1}$ and $dg = \nabla\lambda_1(\mathcal{A}^*y_k) - \nabla\lambda_1(\mathcal{A}^*y_{k-1})$. We then perform a linesearch with Wolfe conditions (see [**47**, Section 3.1]) on the problem

$$\min_{\alpha} \ \lambda_1(\mathcal{A}^*y(\alpha)), \qquad y(\alpha) = \Pi_{\mathcal{C}}(y - \alpha g). \tag{4.2.5}$$

This method converges R-linearly for strongly convex functions and is found to outperform standard gradient descent significantly on differentiable functions [**69**]. However, the linesearch has the added cost of additional objective evaluations $\lambda_1(\mathcal{A}^*y)$ if the initial value for α does not satisfy the Wolfe conditions.

If instead (4.2.3) fails and $\lambda_1(\mathcal{A}^*y)$ is nondifferentiable, then the we revert to a decreasing steplength sequence $\{\alpha_k\}$. For convex models like the PLGD model, it is know that any sequence of steplengths satisfying the conditions

$$\lim_{k\to\infty} \alpha_k = 0 \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k = \infty \tag{4.2.6}$$

will generate a sequence y_k converging to an optimal solution (see, e.g., [**7**, Proposition 1.2.3 and Section 3.3.1]). A typical choice for this steplength sequence is $\alpha_k = \mathcal{O}(1/k)$.

The linesearch subproblem (4.2.5) and the resulting update $\hat{y} = y - \alpha g$ require projection onto the feasible region $\mathcal{C} = \langle b, y\rangle - \epsilon||y||_2 \geq 1$. If $\epsilon = 0$, this projection has the closed form expression

$$\Pi_C(\hat{y}) = \begin{cases} \hat{y} + \frac{1-\langle b,\hat{y}\rangle}{||b||_2^2} b & \text{if } \langle b, \hat{y}\rangle < 1 \\ \hat{y} & \text{else.} \end{cases} \tag{4.2.7}$$

If $\epsilon > 0$, then the projection is the solution to the problem

$$\min_{y\in\mathbb{R}^m} \tfrac{1}{2}||y - \hat{y}||_2^2 \quad \text{subject to} \quad \langle b, y\rangle - \epsilon||y||_2 \geq 1. \tag{4.2.8}$$

The KKT conditions of this problem can be simplified into a one-dimensional degree-4 polynomial whose largest real root is used to express $\Pi_{\mathcal{C}}(\hat{y})$ again as a linear combination of $\hat{y}$ and b (details omitted, see [**8**, Chapter 5]).

Given the updated dual variable y, we must recover a primal variable (signal) x to test for primal feasibility ($||\mathcal{A}(xx^*) - b||_2 \leq \epsilon$). Corollary 3.4.3 indicates that the general primal recovery problem is

$$\min_{S\succeq 0} ||\mathcal{A}(V_1 S V_1^*) - b_\epsilon||_2^2, \quad b_\epsilon := b - \epsilon\tfrac{y}{||y||_2}, \tag{4.2.9}$$

where $V_1 \in \mathbb{C}^{r_1\times n}$ are the eigenvectors corresponding to λ_1, r_1 is the multiplicity of λ_1, $S \in \mathbb{C}^{r_1\times r_1}$ is positive semidefinite, and b_ϵ is the noisy observation b with a removal of the current approximation $\epsilon y/||y||_2$ to the noise term η as shown in (3.4.10). If $\lambda_1(\mathcal{A}^*y)$ is differentiable, then λ_1 is unique and (4.2.9) simplifies to the closed-form expression

$$\hat{x} = [\langle \mathcal{A}(v_1 v_1^*), b_\epsilon\rangle]_+ / ||\mathcal{A}(v_1 v_1^*)||_2^2. \tag{4.2.10}$$

The previous steps constitute a basic projected gradient descent method for optimizing the PLGD model. We now discuss a pair of refinement steps which exploit the PLP-PLGD optimality conditions (Corollaries 3.4.1, 3.4.3) to accelerate the convergence of this algorithm. A primal refinement step aims to recover a better approximate signal x than the primal recovery solution (4.2.10). If the phase retrieval model has no noise, then a dual refinement step further accelerates the convergence rate of the descent method.

The primal refinement step makes further use of the fact that $\epsilon y/||y||_2$ approximates the noise term η, as indicated by Corollary 3.4.3. Setting $b_\epsilon = b - \epsilon y/||y||_2$, we then solve the unconstrained nonconvex problem

$$\min_{x \in \mathbb{C}^n} h(x) := \tfrac{1}{4}||\mathcal{A}(xx^*) - b_\epsilon||_2^2, \tag{4.2.11}$$

which is initialized with the primal recovery solution $\hat{x}$ from (4.2.10). The problem (4.2.11) can be interpreted as a partially denoised version of the wflow least-squares problem (2.4.9), where the observation b has been replaced by the partially denoised observation b_ϵ.

For noiseless phase retrieval problems, an additional dual refinement method promotes convergence of the dual iterate y. In essence, this step uses optimality conditions and the refined signal x from (4.2.11) to identify the corresponding dual variable y. If the optimal matrix $X_\star = \mathbf{x}\mathbf{x}^*$ is rank-one, then Corollary 3.4.3 (a) indicates that the algebraically largest eigenpair $(\lambda_1^\star, v_1^\star)$ of $\mathcal{A}^* y_\star$ will be unique. Thus $v_1^\star$ will be a rescaling of the optimal signal $\mathbf{x}$, and Corollary 3.4.1 (d) (strong duality) gives this rescaling

$$||X_\star||_1 = ||\mathbf{x}||_2^2 = 1/\lambda_1^\star \implies \mathbf{x} = v_1^\star/\sqrt{\lambda_1^\star}.$$

As a result, the dual refinement step attempts to find the update $y \in \mathcal{C}$ which satisfies $[\mathcal{A}^* y]x = \lambda_1 x$ by solving the constrained linear-least-squares problem

$$\min_{y \in \mathbb{R}^m} \tfrac{1}{2}||[\mathcal{A}^* y]x - \lambda_1 x||_2^2 \quad \text{subject to} \quad \langle b, y\rangle - \epsilon||y||_2 \geq 1, \tag{4.2.12}$$

where $\lambda_1 = \lambda_1(\mathcal{A}^* y)$ is the most recent PLGD objective evaluation and x is the solution to (4.2.11). If the refined iterate $\hat{y}$ improves the dual objective, i.e., $\lambda_1(\mathcal{A}^*\hat{y}) < \lambda_1(\mathcal{A}^* y)$, then $\hat{y}$ replaces y. This spacer iterate, as described in [**7**, Proposition 1.2.5], is guaranteed not to interfere with the convergence behavior of the underlying method.

The above sequence of steps and methods leads to the Gauge Dual Descent (GDD) algorithm for optimizing the PLGD model (3.2.1):

Algorithm 3 Gauge Dual Descent (GDD) algorithm

Input: Sensing operator $\mathcal{A}$ and adjoint $\mathcal{A}^*$, observation vector b, initial dual variable y_0, estimate of total noise level ϵ, convergence tolerances.

Output: Approximate solution signal x.

Initialization: Set $k = 0$.

while not converged **do**

  Compute algebraically largest eigenvalues and corresponding eigenvectors: (λ_1, v_1) and (λ_2, v_2) from $\mathcal{A}^* y_k$.

  Compute (sub)gradient: $g = \mathcal{A}(v_1 v_1^*)$ based on (4.2.1).

  Determine differentiability of $\lambda_1(\mathcal{A}^* y_k)$ based on (4.2.3).

  if $\lambda_1(\mathcal{A}^* y_k)$ is differentiable **then**

    Linesearch: Perform linesearch (4.2.5) with initial step α (4.2.4) to obtain y_{k+1}.

  else

    Projected subgradient step: Set $y_{k+1} = \Pi_{\mathcal{C}}(y_k - \alpha g)$ with α from (4.2.6).

  end if

  Primal recovery: Compute $\hat{x}$ based on (4.2.10).

  Primal refinement: Find x_{k+1} as the solution to (4.2.11) initialized with $\hat{x}$ and y_{k+1}.

  if $\epsilon = 0$ **then**

    Dual refinement: Find $\widehat{y}$ based on (4.2.12).

    if $\lambda_1(\mathcal{A}^* \widehat{y}) < \lambda_1$, set $y_{k+1} = \widehat{y}$

  end if

  Update: $k = k + 1$.

end while

Return: $x = x_k$.

We close this section by describing the implementation details of the GDD algorithm. This algorithm was originally established in [**27**] and implemented in the accompanying software package

`low-rank-opt`[1] (with the identical clone[2]). All tests, experiments, and new methods in this dissertation are available for reproduction in a branched package[3]. This branch includes a few minor changes to the original algorithm which are noncritical to the behavior of the GDD algorithm (see the package repository for details).

In the case of noisy phase retrieval, the input ϵ is a measure of the total noise

$$\epsilon = ||\eta||_2 = ||b - \mathbf{b}||_2$$

of the model (1.0.1). Convergence of the GDD algorithm (step 1) is based on strong duality (Corollary 3.4.1, d) along with primal and dual feasiblility. Since dual feasibility is maintained throughout the GDD algorithm, only primal feasibility and strong duality must be measured. Thus the GDD algorithm terminates when both of the following conditions are met:

$$||\mathcal{A}(xx^*) - b||_2 \leq \epsilon + \text{tol}_{\text{feas}}(1 + ||b||_2), \tag{4.2.13}$$

$$||xx^*||_1 \cdot \lambda_1 \left(\mathcal{A}^* y\right) \leq 1 + \text{tol}_{\text{gap}}. \tag{4.2.14}$$

The convergence tolerances are set to $\text{tol}_{\text{feas}} = \text{tol}_{\text{gap}} = 2 \times 10^{-4}$ for noisy phase retrieval, and $\text{tol}_{\text{feas}} = \text{tol}_{\text{gap}} = 1 \times 10^{-5}$ for noiseless.

If the dual variable y is not provided, then it is initialized as $y = \Pi_{\mathcal{C}}(b)$. On the 0-th iteration of the GDD algorithm, steps 5-9 are skipped. This strategy has the result of avoiding an additional eigenvalue computation (step 6) during initialization. Additionally, the GDD algorithm has the default setting of skipping dual refinement in the presence of noise (as dual refinement inhibits convergence for noisy phase retrieval models, see Section 5.3).

The (sub)gradient computation (Algorithm 3, line 3) as described in (4.2.1) is always set to $g = \mathcal{A}(v_1 v_1^*)$ for the eigenvector v_1 regardless of the multiplicity of λ_1. Since noisy phase retrieval problems often deal with nondifferentiable objectives (see Section 5.3), the GDD algorithm includes a test for differentiability (4.2.3) with the default tolerance $\text{tol}_{\text{diff}} = 10^{-5}$. If this condition fails during some iteration k, the GDD algorithm performs a projected subgradient step (step 8) with

[1] `https://www.cs.ubc.ca/~mpf/pubs/low-rank-spectral-optimization-via-gauge-duality/`
[2] `https://github.com/Will-Wright/low-rank-opt-orig`
[3] `https://github.com/Will-Wright/low-rank-opt-rapid-eig`

a specific steplength α_k. Testing indicates that projecting the Barzilai-Borwein steplength (4.2.4) onto an interval with monotonically decreasing bounds maintains progress of the GDD algorithm across test cases. Thus the steplength is set to

$$\alpha_k = \min\left\{u_k, \max\left\{l_k, \frac{\langle dy, dy\rangle}{\langle dy, dg\rangle}\right\}\right\}, \tag{4.2.15}$$

where $u_k = 200/k$ and $l_k = 1/k$.

Computationally, most of the steps in the GDD algorithm are very simple. The three key exceptions are the eigenvalue computation (steps 2, 6, and 14), the primal refinement (step 11), and the dual refinement (step 13). In the original implementation, the eigenvalue computation is performed using the MATLAB function `eigs` (see Section 6.3 for details about this method), where only the first two algebraically largest eigenvalues λ_1, λ_2 are requested. The primal refinement (step 11) is computed using `minFunc`, a quasi-Newton solver for unstrained optimization, with the descent direction determined using the limited-memory (l-)BFGS method for Hessian approximation [**56**]. Dual refinement (step 13) is computed with `minConf`, another quasi-Newton solver which is optimized for problems like (4.2.12) with expensive objective functions and simple constraints [**57**], [**58**].

In each of these three subroutines, the primary computational cost comes from $\mathcal{A}$-products (2.4.2), where each $\mathcal{A}(xx^*)$ product requires L DFTs and each $[\mathcal{A}^*y]x$ product requires $2L$ DFTs. Thus we measure computational costs in terms of number of DFTs, following the convention of [**14**] and [**27**].

Theoretically, the steplength strategy in the GDD algorithm is guaranteed to converge for all feasible problems (i.e., find an optimal pair $(X_\star, y_\star)$ which satisfies the (PLGD) optimality conditions of Corollary 3.4.1) [**69**], [**7**, Proposition 1.2.3 and Section 3.3.1]. Yet the rate of convergence differs greatly for noiseless and noisy phase retrieval problems. The GDD algorithm is particularly efficient for noiseless problems, as we discuss briefly in Section 4.3. However, noisy problems pose a unique set of challenges which are intrinsic to the PLGD model (3.2.1) which are addressed in the remainder of this dissertation.

4.3. Noiseless Phase Retrieval

This section examines the behavior of the GDD algorithm (Algorithm 3) for noiseless phase retrieval problems. The efficiency of the GDD algorithm for noiseless problems is well-established in [**27**, Sections 5.1.1, 5.1.3]. The purpose of this section is to establish the role of the primal (4.2.11) and dual refinement (4.2.12) steps for noiseless phase retrieval so we may examine the utility of these steps for noisy phase retrieval in Chapter 5.

We begin by discussing the general behavior of each refinement step and the relationship between the two. For noiseless problems ($\epsilon = 0$), the primal refinement step (4.2.11) is initialized with $b_\epsilon = b$ and is therefore independent of the dual variable y other than the fact that (4.2.11) is initialized with $\hat{x}$ from (4.2.10). In this case, (4.2.11) is equivalent to the wflow least-squares problem (2.4.9) discussed in Section 2.4. As proved in [**65**] and restated at the end of Section 2.4, if this problem has a sufficient oversampling L, then the objective function of (4.2.11) will have no spurious local minima, only one global minima (up to a phase constant), and negative directional curvature at all saddle points. As a result, for noiseless problems the primal refinement step effectively solves the phase retrieval problem on the first iteration of the GDD algorithm (i.e., x typically has a relative error $||\mathbf{x}\mathbf{x}^* - xx^*||_F/||\mathbf{x}||_2^2$ on the order of 10^{-7} or smaller). Thus the primal refinement step serves as an effective standalone method for promoting the quick convergence of the primal signal x.

In contrast to primal refinement, the dual refinement problem (4.2.12) is directly dependent on the accuracy of the primal signal x. If the dual refinement is initialized with an inaccurate signal, then the returned dual variable $\hat{y}$ may be inferior to the initial update y_+. Yet $\hat{y}$ may also satisfy the condition for replacing the previous variable (step 14), preventing the GDD algorithm from converging.

Table 4.1 compares the behavior of these refinement steps for a random noiseless phase retrieval problem.

	Primal & dual ref.	Primal ref. only	Dual ref. only	No ref.
Iterations	1	62	1,000	1,000
DFTs	21,800	349,420	273,817,590	14,686,690
Relative error	7.469×10^{-9}	8.097×10^{-9}	1.166×10^{2}	1.176×10^{2}

TABLE 4.1. The GDD algorithm (Algorithm 3) with and without primal (4.2.11) and dual refinement (4.2.12). This experiment involves a random gaussian signal of size $n = 128$ with $L = 10$ observations and no noise. Both termination conditions (4.2.13) and (4.2.14) are required. Signal relative error is measured as $||\mathbf{x}\mathbf{x}^* - x_0 x_0^*||_F / ||\mathbf{x}||_2^2$.

Table 4.1 demonstrates that primal refinement is necessary for the GDD algorithm to converge, and dual refinement is essential to the efficiency of the GDD algorithm. With only primal refinement, the GDD algorithm exhibits a convergence rate typical of a steepest descent method. In this case, primal feasibility (4.2.13) is achieved in the first few iterates. Yet the duality gap conditon (4.2.14) is not satisfied until the dual variable y is sufficiently close to $y_\star$. With only the dual refinement, the GDD algorithm fails to converge as the dual problem (4.2.12) is initialized with an inaccurate signal x. When the GDD algorithm is run without either refinement step, it again fails to converge.

However, when both the primal and dual refinement subroutines are used, the GDD algorithm rarely takes more than a few iterations. Since steps 5-9 are skipped during the 0-th iterate, the GDD algorithm serves as an improvement to the wflow algorithm. Both algorithms set the initial signal as the eigenvector corresponding to the algebraically largest eigenvalue of $\mathcal{A}^* b$ (where the GDD algorithm first rescales b by projecting it onto $\mathcal{C}$). Yet the GDD algorithm uses more effective search directions generated by the l-BFGS method rather than the Wirtinger derivative. Next, the dual refinement problem (4.2.12) is initialized with a sufficient pair of terms (λ_1, x) to recover a nearly optimal dual iterate y. Thus the primal feasibility (4.2.13) and duality gap (4.2.14) conditions are typically met within a few iterations.

The GDD algorithm with primal and dual refinement is very effective for noiseless phase retrieval, acting as a robust, efficent competitor to the wflow algorithm with the added benefit of greater signal accuracy [**27**, Section 5.1.1, 5.1.3]. Note that the optimality of y is unnecessary

if we simply want to minimize the noiseless problem $||\mathcal{A}(xx^*) - b||_2$. To this end, the GDD algorithm as implemented includes a setting specifically for noiseless problems, where the strong duality condition (4.2.14) is dropped and only primal feasibility (4.2.13) is required. This primal feasibility version of the GDD algorithm typically converges in 0-1 iterations with about the same computational cost as the wflow algorithm.

Unlike noiseless phase retrieval, noisy phase retrieval poses a few key challenges for the GDD algorithm. This dissertation addresses the convergence and computational challenges noisy phase retrieval poses for the GDD algorithm with the following contributions.

Chapter 5 addresses the convergence challenges the GDD algorithm (Algorithm 3) experiences for noisy phase retrieval problems. We begin by demonstrating that the GDD algorithm stagnates prior to convergence for noisy problems and determining the cause of this stagnation. We then establish new termination conditions which indicate stagnation, so we may treat the GDD algorithm as a black-box and focus our attention on handling the *evolving matrix eigenvalue problem* (EMEP) as defined in Chapter 6.

In Chapter 6 we define the EMEP in the GDD algorithm and develop a new strategy for handling this problem. We see that the algebraically largest eigenvalues of the EMEP cluster for later iterates, resulting in more difficult eigenvalue problems. To develop a new strategy for handling the EMEP, Section 6.3 reviews the *implicitly restarted Arnoldi method* (IRAM), a common method for large-scale eigenvalue problems. Next, Section 6.4 develops *the IRAM with adaptive parameter selection*, a new strategy for choosing the IRAM parameters to handle the EMEP. We close Chapter 6 by examining how this new strategy relates to the clustering of eigenvalues in the EMEP.

Finally, Chapter 7 provides performance results for the IRAM with adaptive parameter selection, demonstrating the efficiency of this method compared to the default IRAM parameter choice for the GDD algorithm. Chapter 8 concludes this dissertation with a summary of our work and suggestions for future work.

CHAPTER 5

Algorithm Stagnation for Noisy Phase Retrieval

5.1. Introduction

In Chapter 4 we developed the GDD algorithm (Algorithm 3) to optimize the PLGD model (3.2.1) and saw that the GDD algorithm is efficient and accurate for noiseless phase retrieval. We now examine the tendency of the GDD algorithm to fail to converge for noisy phase retrieval and establish new termination conditions for the GDD algorithm.

Section 5.2 describes our method of constructing experimental noisy PLGD models and establishes potential residuals for measuring the progress of the GDD algorithm. Section 5.3 then examines the tendency of the GDD algorithm not to converge for noisy phase retrieval problems. We see that signals observed with nontrivial noise tend to have optimal PLGD matrices $X_\star$ with rank greater than one, preventing first-order algorithms like the GDD algorithm from converging. To handle this issue, Section 5.4 establishes new termination conditions based on heuristic evidence which indicates that the GDD algorithm has stopped making signal recovery progress. These new termination conditions allow us to treat the GDD algorithm as a black-box solver in Chapter 6, where we focus on the challenging sequence of eigenvalue problems in the GDD algorithm.

5.2. Experimental Models and Residuals

This section describes two methods for creating experimental noisy phase retrieval problems and presents a set of potential residuals to measure the progress of the GDD algorithm (Algorithm 3). The *phase retrieval problem with Gaussian noise* imitates the typical phase retrieval scenario and is used throughout this dissertation as the default method for creating noisy phase retrieval problems. The *phase retrieval problem with synthetic noise* is constructed around the PLGD primal recovery conditions (Corollary 3.4.3) and is used exclusively in Section 5.3 to examine the convergence behavior of the GDD algorithm. The set of potential residuals is a combination of those discussed in Chapter 4 and new residuals based on the variables available in the GDD algorithm.

We begin by describing experimental noisy phase retrieval problems which have *Gaussian noise*. Recall that the noisy phase retrieval problem (1.0.1) involves an observation $b = \mathbf{b} + \eta$, where the true observation $\mathbf{b}$ is contaminated by some nontrivial noise η. To mimic the typical experimental noisy phase retrieval scenario, we begin with an unknown signal $\mathbf{x}$. A sensing operator $\mathcal{A}$ (2.4.2) is then chosen with diagonal mask matrices C_j whose diagonal elements have complex standard Gaussian distribution (1.0.19) (see Chapter 1 for an explanation of masks). The sensing operator is then used to create the true observation $\mathbf{b} = \mathcal{A}(\mathbf{x}\mathbf{x}^*)$, and a noise term η is chosen with real standard Gaussian distribution (1.0.18). Finally, the noisy observation is set as $b = \mathbf{b} + \eta$. Altogether, given a signal $\mathbf{x}$, a sensing operator $\mathcal{A}$ (2.4.2), and noise ratio ϵ_{rel}, we create the phase retrieval problem with Gaussian noise experimentally with the steps

$$\begin{aligned} &1) \quad \mathbf{b} = \mathcal{A}(\mathbf{x}\mathbf{x}^*), \\ &2) \quad \eta \sim \mathcal{N}(0,1), \\ &3) \quad b = \mathbf{b} + \eta, \end{aligned} \tag{5.2.1}$$

where η is rescaled in the third step to satisfy the noise ratio $\epsilon_{\text{rel}} = ||\eta||_2/||b||_2$. Thus we define the *phase retrieval problem with Gaussian noise* as

$$\begin{aligned} &\text{find} \quad x \\ &\text{s.t.} \quad ||\mathcal{A}(xx^*) - b||_2 \leq \epsilon, \end{aligned} \tag{5.2.2}$$

where $b = \mathbf{b} + \eta$ is constructed experimentally using the steps (5.2.1). Likewise, the *PLGD model with Gaussian noise* refers to the the PLGD model (3.2.1)

$$\begin{aligned} &\min_y \quad \lambda_1(\mathcal{A}^* y) \\ \text{(PLGD)} \quad &\text{s.t.} \quad \langle b, y\rangle - \epsilon||y||_2 \geq 1, \end{aligned} \tag{5.2.3}$$

where again $b = \mathbf{b} + \eta$ is constructed experimentally using the steps (5.2.1).

To demonstrate how the differentiability of the dual objective $\lambda_1(\mathcal{A}^* y)$ impacts the behavior of the GDD algorithm, Section 5.3 also considers experimental noisy phase retrieval problems which we say have *synthetic noise*. The purpose of this synthetic noise is to create a PLGD model where the dual objective $\lambda_1(\mathcal{A}^* y)$ is differentiable at $y_\star$. To achieve this goal, the phase retrieval problem with synthetic noise is constructed to satisfy the PLGD primal recovery conditions of Corollary 3.4.3.

First we choose a random variable with standard Gaussian distribution (1.0.18) to be the optimal dual variable $y_\star$ and use this vector to construct the rest of the noisy phase retrieval problem. The true signal $\mathbf{x}$ is set as the eigenvector corresponding to the algebraically largest eigenvalue of $\mathcal{A}^* y_\star$ per Corollary 3.4.3 (a). This signal is then used to create a true observation $\mathbf{b}$ which is noised with a rescaling of $y_\star$ per Corollary 3.4.3 (b) and (3.4.10). Altogether, given a noise ratio ϵ_{rel} we have the following steps for constructing the phase retrieval problem with synthetic noise

$$\begin{aligned} &1) \quad y_\star \sim \mathcal{N}(0,1), \\ &2) \quad \mathbf{x} \text{ is the eigenvector corresponding to} \\ &\qquad \text{the algebraically largest eigenvalue of } \mathcal{A}^* y_\star, \\ &3) \quad \mathbf{b} = \mathcal{A}(\mathbf{x}\mathbf{x}^*), \\ &4) \quad b = \mathbf{b} + \epsilon \tfrac{y_\star}{||y_\star||_2}, \end{aligned} \tag{5.2.4}$$

where $\epsilon = \epsilon_{\text{rel}}||b||_2$. Steps one and two in (5.2.4) guarantee the PLP-PLGD optimality conditions (Corollary 3.4.1) will hold for a known pair ($X_\star = \mathbf{x}\mathbf{x}^*$, $y_\star$) with rank-one optimal matrix $X_\star$. Additionally, step four in (5.2.4) satisfies the primal recovery equation (3.4.10), guaranteeing the primal refinement strategy (4.2.11) used in the GDD algorithm will recovery the true signal $\mathbf{x}$ when initialized with $y_\star$. We define the *phase retrieval problem with synthetic noise* as

$$\begin{aligned} \text{find} \quad & x \\ \text{s.t.} \quad & ||\mathcal{A}(xx^*) - b||_2 \leq \epsilon, \end{aligned} \tag{5.2.5}$$

where $b = \mathbf{b} + \epsilon \frac{y_\star}{||y_\star||_2}$ is constructed experimentally using the steps (5.2.4). Likewise, the *PLGD model with synthetic noise* refers to the the PLGD model (3.2.1)

$$\begin{aligned} \min_{y} \quad & \lambda_1(\mathcal{A}^* y) \\ \text{(PLGD)} \quad \text{s.t.} \quad & \langle b, y \rangle - \epsilon ||y||_2 \geq 1, \end{aligned} \tag{5.2.6}$$

where again $b = \mathbf{b} + \epsilon \frac{y_\star}{||y_\star||_2}$ is constructed experimentally using the steps (5.2.4).

Next we establish an exhaustive list of residuals which are available for measuring the progress of the GDD algorithm. Here the PLGD dual matrix is defined $A := \mathcal{A}^*(y)$ and previous iterates are denoted with a hat (e.g., $\hat{y} = y_{k-1}$). Note that this set of residuals does not contain a residual

for dual feasibility because the GDD algorithm maintains dual feasibility (each gradient descent step projects the dual iterate y onto the feasible set).

(5.2.7a)	signal relative error	$\|\|xx^* - \mathbf{x}\mathbf{x}^*\|\|_F / \|\|\mathbf{x}\mathbf{x}^*\|\|_F$
(5.2.7b)	dual relative error	$\|\|y - y_\star\|\|_2 / \|\|y_\star\|\|_2$
(5.2.7c)	primal true relative error	$\|\|\mathcal{A}(xx^*) - \mathbf{b}\|\|_2 / \|\|\mathbf{b}\|\|_2$
(5.2.7d)	primal relative error	$\rho := \|\|\mathcal{A}(xx^*) - b\|\|_2 / \|\|b\|\|_2$
(5.2.7e)	primal difference	$\|\rho - \hat{\rho}\| / \|\rho\|$
(5.2.7f)	duality gap	$\gamma := \|\|xx^*\|\|_1 \cdot \lambda_1 - 1$
(5.2.7g)	duality gap difference	$\|\gamma - \hat{\gamma}\| / \|\gamma\|$
(5.2.7h)	dual objective difference	$\|\lambda_1 - \hat{\lambda}_1\| / \|\lambda_1\|$
(5.2.7i)	dual variable difference	$\|\|y - \hat{y}\|\|_2 / \|\|y\|\|_2$
(5.2.7j)	dual matrix difference	$\|\|A - \hat{A}\|\|_F / \|\|A\|\|_F$

The residuals in (5.2.7) can be separated into three groups. The first group contains ideal error measurements, including the signal relative error (5.2.7a), dual relative error (5.2.7b), and primal true relative error (5.2.7c). The next group of residuals are those used as termination conditions for the GDD algorithm in Chapter 4: the primal relative error (5.2.7d) and the duality gap (5.2.7f), used in conditions (4.2.13) and (4.2.14), respectively. The final group of residuals are the five difference residuals (5.2.7e, g-j). We present these difference residuals as a new set of measurements for determining when the GDD algorithm is no longer making signal recovery progress and termination should be declared. As we will see in Section 5.4, the primal difference (5.2.7e) and dual variable difference (5.2.7i) are the two difference residuals best suited for determining stagnation of the GDD algorithm, and thus used to establish new termination conditions in that section.

5.3. Algorithm Stagnation

In this section we demonstrate the tendency of the GDD algorithm (Algorithm 3) not to converge when solving PLGD models with Gaussian noise (5.2.3). We see that PLGD models with Gaussian noise (5.2.3) typically have an optimal matrix $X_\star$ with rank greater than one. In this circumstance, Corollary 3.4.1 (e) indicates that the algebraically largest eigenvalue of the optimal dual matrix $\mathcal{A}^* y_\star$ will also be greater than one. As a result, the dual objective $\lambda_1(\mathcal{A}^* y)$ will be nondifferentiable in a neighborhood of $y_\star$ and first-order algorithms like the GDD algorithm may not converge. Additionally, since $X_\star \neq \mathbf{x}\mathbf{x}^*$, the primal refinement strategy (4.2.11) used in the GDD algorithm is not guaranteed to recover the true signal $\mathbf{x}$.

In order to describe the behavior of the GDD algorithm for noisy phase retrieval problems, we use the following terminology to make the distinction between the GDD algorithm converging to an optimal solution and converging to a nonoptimal variable. Recall that the GDD algorithm has *converged* at a given iterate if the conditions (4.2.13) and (4.2.14) are satisfied. We say that the GDD algorithm *failed to converge* if conditions (4.2.13) or (4.2.14) are not satisfied for any iterate within a maximum number of iterations (in this chapter we choose 1,000 iterations). In contrast with converging, we say the GDD algorithm has *stagnated* at a given iterate if the progress from the previous iterate is trivial and the GDD algorithm has not yet converged. Specifically, the GDD algorithm has stagnated at a given iterate if a chosen subset of difference residuals (5.2.7e, g-j) are below a required set of tolerances.

The authors of [**27**] use PLGD models with synthetic noise (5.2.6) to demonstrate that the GDD algorithm is able to converge and accurately recover the true signal $\mathbf{x}$ when the optimal matrix in the PLGD model is rank-one (i.e., $X_\star = \mathbf{x}\mathbf{x}^*$). Table 5.1 depicts the results of the GDD algorithm applied to the PLGD model with synthetic noise (5.2.6), corroborating the results in [**27**, Section 5.1.2].

	$L = 5$		$L = 10$		$L = 15$	
	Iters	xErr	Iters	xErr	Iters	xErr
$\epsilon_{\text{rel}} = 0.05$	154	8.581×10^{-3}	112	8.442×10^{-3}	355	1.618×10^{-3}
$\epsilon_{\text{rel}} = 0.15$	207	2.772×10^{-3}	255	3.500×10^{-4}	82	1.196×10^{-2}
$\epsilon_{\text{rel}} = 0.30$	186	1.636×10^{-3}	104	2.047×10^{-3}	111	4.606×10^{-3}

TABLE 5.1. Number of iterations and signal relative error (5.2.7a) (xErr) for the GDD algorithm (Algorithm 3) applied to the PLGD model with synthetic noise (5.2.6). Signal size is $n = 128$, with various noise ratios ϵ_{rel} and oversampling values L. The convergence conditions (4.2.13) and (4.2.14) are set to tolerances $\text{tol}_{\text{feas}} = \text{tol}_{\text{gap}} = 2 \times 10^{-4}$. Note that each signal relative error in Table 5.1 was 1-3 orders of magnitude smaller than the relative error of the signal returned by the wflow algorithm (Algorithm 2) for the same problem.

Table 5.1 demonstrates that the GDD algorithm tends to converge within a few hundred iterations when solving PLGD models with synthetic noise (5.2.6). However, the convergence behavior depicted in Table 5.1 does not occur when the GDD algorithm solves PLGD models with Gaussian noise (5.2.3). In Table 5.2, the experiments from Table 5.1 are replaced with phase retrieval problems with Gaussian noise (5.2.2). Note that the iteration count is replaced by the final duality gap value, since the GDD algorithm failed to converge after 1,000 iterations for all cases.

	$L = 5$		$L = 10$		$L = 15$	
	duGap	xErr	duGap	xErr	duGap	xErr
$\epsilon_{\text{rel}} = 0.05$	323.03	9.072×10^{-2}	101.56	4.053×10^{-2}	93.08	3.112×10^{-2}
$\epsilon_{\text{rel}} = 0.15$	448.68	4.200×10^{-1}	364.07	1.249×10^{-1}	374.10	9.443×10^{-2}
$\epsilon_{\text{rel}} = 0.30$	492.56	$1.096e \times 10^{0}$	665.73	2.973×10^{-1}	747.68	1.958×10^{-1}

TABLE 5.2. Final duality gap (5.2.7f) (duGap) and signal relative error (5.2.7a) (xErr) for the GDD algorithm (Algorithm 3) applied to PLGD models with Gaussian noise (5.2.3). The GDD algorithm failed to converge after 1,000 iterations for all problems. Signal size is $n = 128$, with various noise ratios ϵ_{rel} and oversampling values L. The convergence conditions (4.2.13) and (4.2.14) are set to tolerances $\text{tol}_{\text{feas}} = \text{tol}_{\text{gap}} = 2 \times 10^{-4}$.

As we see in Table 5.2, the duality gap value (5.2.7f) remains several orders of magnitude above the duality gap convergence tolerance (4.2.14). As a result, the GDD algorithm fails to converge for PLGD models with Gaussian noise (5.2.3).

The next experiment demonstrates that the GDD algorithm applied to PLGD models with Gaussian noise (5.2.3) tends to achieve primal feasibility (4.2.13) during the early iterations and then stagnates within a few hundred iterations. Since the model (5.2.3) is constructed without knowledge of an optimal PLP-PLGD pair $(X_\star, y_\star)$, we use the interior-point solver `SDPT3` [**66**] to obtain the pair $(X_\star, y_\star)$ within square-root machine-precision. In order to use this interior-point solver, the models in Figure 5.1 are very small.

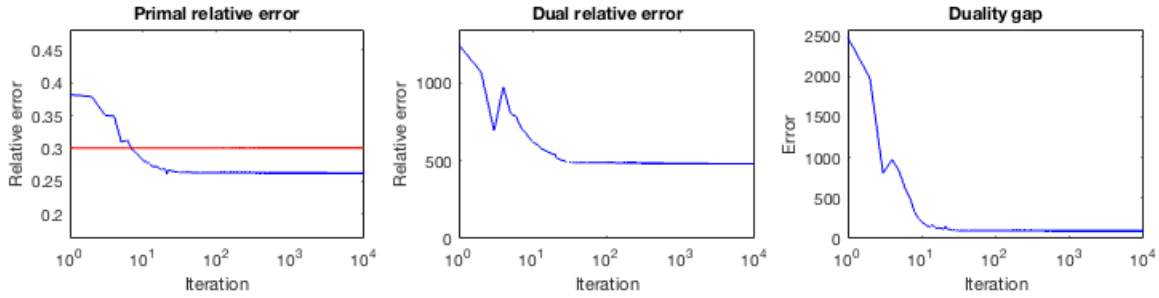

FIGURE 5.1. Primal relative error (5.2.7d), dual relative error (5.2.7b), and duality gap (5.2.7f) for 10,000 iterates of the GDD algorithm (Algorithm 3) applied to a natural noise model with $n = 16$, $L = 6$ observations, and noise ratio 0.30. The horizontal axis is log-plotted to highlight early progress along with later stagnation. The pair $(X_\star, y_\star)$ are computed with `SDPT3`.

Figure 5.1 demonstrates that the GDD algorithm stagnates when attempting to solve a PLGD model with Gaussian noise (5.2.3). In this problem, primal feasibility (4.2.13) is acheived at iterate 6, and further progress is made over the following early iterations, as indicated by the primal relative error (5.2.7d) plot. However, the strong duality condition (4.2.14) is never satisfied, and the final duality gap at iterate 10,000 is 97.63. Likewise, the dual variable y_k does not approach the optimal dual variable $y_\star$, as indicated by the dual relative error (5.2.7b) plot. In this particular PLGD model, $X_\star$ is rank three, causing the dual objective $\lambda_1(\mathcal{A}^* y)$ to be nondifferentiable near $y_\star$ per Corollary 3.4.1 (e). Thus the GDD algorithm identifies the dual objective $\lambda_1(\mathcal{A}^* y)$ as nondifferentiable at iterate 84 reverts to the monotone stepsize sequence (4.2.15).

The next experiment establishes that PLGD models with Gaussian noise (5.2.3) almost always have an optimal matrix $X_\star$ with rank greater than one, and this rank problem causes the stagnation of the GDD algorithm. Table 5.3 compares PLGD models with synthetic (5.2.6) and Gaussian noise (5.2.3), depicting the rank of the optimal matrix $X_\star$ and the behavior of the GDD algorithm.

		Synthetic noise			Gaussian noise		
		$L = 4$	$L = 6$	$L = 8$	$L = 4$	$L = 6$	$L = 8$
	$\text{rank}(X_\star)$	1	1	1	3.41	3.28	3.27
$\epsilon_{\text{rel}} = 0.05$	GDD itns.	125.09	144.20	182.26	1,000	1,000	1,000
	duGap	1.17_{-4}	1.18_{-4}	1.18_{-4}	9.60	3.88	3.86
	$\text{rank}(X_\star)$	1	1	1	2.99	3.00	3.04
$\epsilon_{\text{rel}} = 0.15$	GDD itns.	62.48	85.32	95.58	1,000	1,000	1,000
	duGap	1.20_{-4}	1.35_{-4}	1.33_{-4}	15.9	14.0	15.8
	$\text{rank}(X_\star)$	1	1	1	2.64	2.70	2.89
$\epsilon_{\text{rel}} = 0.30$	GDD itns.	40.34	50.88	64.28	1,000	1,000	1,000
	duGap	1.17_{-4}	1.23_{-4}	1.27_{-4}	21.2	26.7	31.5

TABLE 5.3. The GDD algorithm (Algorithm 3) results and optimal matrix rank for PLGD models with synthetic (5.2.6) and Gaussian noise (5.2.3). This table depicts the mean rank of $X_\star$, number of the GDD algorithm iterations, and final duality gap (5.2.7f) (duGap) for 100 random phase retrieval models with signal size $n = 16$, noise ratio ϵ_{rel}, and oversampling L. In all synthetic noise models (5.2.6), the solution $X_\star$ was rank-one and the GDD algorithm converged. In all Gaussian noise models (5.2.3), the algorithm reached the maximum of 1,000 iterations without attaining the termination condition (4.2.14). Note that pairs $(X_\star, y_\star)$ are computed with `SDPT3` and numbers n_{-k} are shorthand for $n \times 10^{-k}$.

Table 5.3 demonstrates that PLGD models with Gaussian noise (5.2.3) typically have optimal matrices with rank greater than one, causing the GDD algorithm to stagnate. The GDD algorithm cannot attain an optimal primal matrix $X_\star$ with rank greater than one because the primal recovery (4.2.10) and refinement (4.2.11) steps used in this algorithm only return a rank-one matrix $X = xx^*$. Additionally, Corollary 3.4.1, (e) indicates that $\text{rank}(X_\star)$ is a lower bound on the multiplicity of the algebraically largest eigenvalue of $\mathcal{A}^* y_\star$. Thus the objective function $\lambda_1(\mathcal{A}^* y)$ will

be nondifferentiable in some neighborhood around $y_\star$ and the GDD algorithm will stagnate prior to approaching $y_\star$. As a result, the GDD algorithm cannot attain a pair (X, y) that are sufficiently close to the optimal pair $(X_\star, y_\star)$ and fails to satisfy the duality gap condition (4.2.14).

In contrast to the Gaussian models (5.2.3), we see that PLGD models with synthetic noise (5.2.6) consistenly have rank-one optimal matrices $X_\star$, allowing the GDD algorithm to converge. Each synthetic noise model (5.2.6) in Table 5.3 had an optimal dual matrix $\mathcal{A}^* y_\star$ with a unique algebraically largest eigenvalue, making the dual objective function $\lambda_1(\mathcal{A}^* y)$ differentiable at $y_\star$ and allowing the GDD algorithm to approach the optimal dual variable $y_\star$. Once the pair (X, y) are sufficiently close to the optimal pair, the duality gap condition (4.2.14) is met and convergence is declared. At this point, the primal recovery equation (3.4.10) successfully denoises the noisy observation $b = \mathbf{b} + \epsilon y_\star / ||y_\star||_2$ because the synthetic noise steps (5.2.4) guarantee an exact relation $\eta = \epsilon y_\star / ||y_\star||_2$ between the noise term η and optimal dual variable $y_\star$. Thus the GDD algorithm is able to return a matrix X which accurately approximates the optimal matrix $X_\star = \mathbf{x}\mathbf{x}^*$.

Table 5.3 also helps to explain why dual refinement (Algorithm 3, step 13) is not beneficial for PLGD models with Gaussian noise (5.2.3). As we saw in Table 4.1, an inaccurate signal x can cause the dual refinement problem (4.2.12) to return an unreliable update $\hat{y}$. Figure 5.2 depicts the progress made by the GDD algorithm with and without dual refinement for three PLGD models with Gaussian noise (5.2.3).

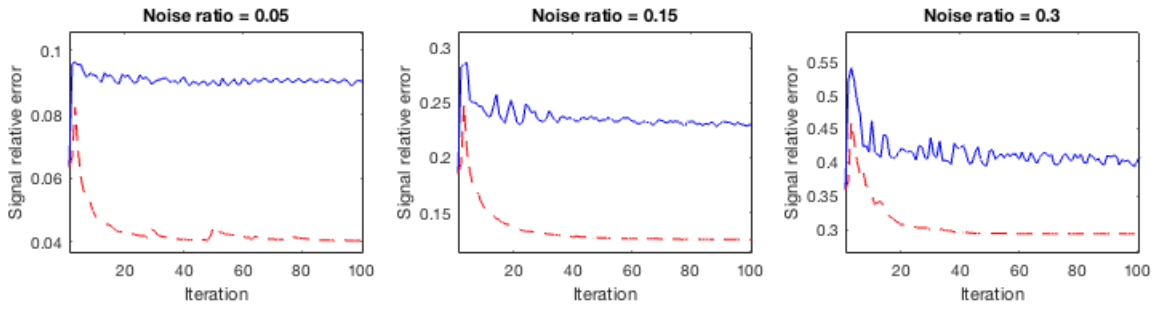

FIGURE 5.2. A comparison of the GDD algorithm (Algorithm 3) with and without dual refinement (4.2.12) for PLGD models with Gaussian noise (5.2.3) with various noise levels, with random Gaussian signal of size $n = 128$ and $L = 10$ observations. The solid blue line indicates dual refinement, and the dashed red line indicates no dual refinement.

Figure 5.2 demonstrates that the dual refinement step (4.2.12) inhibits the progress otherwise made by the GDD algorithm. This step is initialized with a signal x which corresponds to the rank-one matrix iterate $X = xx^*$. Since the optimal matrix $X_\star$ tends to have rank greater than one, the dual refinement problem (4.2.12) will not be properly initialized and may return a poor update $\hat{y}$. Thus the dual refinement step in the GDD algorithm is not beneficial for signal recovery in phase retrieval problems with Gaussian noise (5.2.2).

5.4. New Termination Conditions

In Section 5.3, we saw that the GDD algorithm (Algorithm 3) tends to stagnate when solving phase retrieval problems with Gaussian noise (5.2.2), failing to satisfy the duality gap termination condition (4.2.14) original established in [**27**]. This section establishes new termination conditions for the GDD algorithm for PLGD models with Gaussian noise (5.2.3) and demonstrates the effectiveness of these conditions for identifying stagnation of the GDD algorithm. (For a comparison of the unused residuals from (5.2.7), see Appendix B.)

We begin by presenting the new termination conditions for the GDD algorithm for PLGD models with Gaussian noise (5.2.3). To identify stagnation of the GDD algorithm and declare termination, the primal difference (5.2.7e) is set to

$$\frac{|\rho - \hat{\rho}|}{\rho} \leq \text{tol}_{\text{primal}} = 10^{-5}, \quad \rho = ||\mathcal{A}(xx^*) - b||_2 \tag{5.4.1}$$

and the dual variable difference (5.2.7i) is set to

$$\frac{||y - \hat{y}||_2}{||y||_2} \leq \text{tol}_{\text{dual}} = 10^{-4}, \tag{5.4.2}$$

where a hat indicates the previous iterate (e.g., $\hat{y} = y_{k-1}$). Termination is declared when (5.4.1) and (5.4.2) are satisfied or after a maximum of 300 iterations are performed. After termination, the signal returned corresponds to the signal among the previous 20 with the smallest duality gap (5.2.7f) value.

To demonstrate the effectiveness of these termination conditions, we consider a set of six PLGD models with Gaussian noise (5.2.3) solved with the GDD algorithm. The signal relative error (5.2.7a) serves as a control measurement to identify when the GDD algorithm has stagnated and should terminate. Figure 5.3 depicts the signal relative error (5.2.7a) for each of these models.

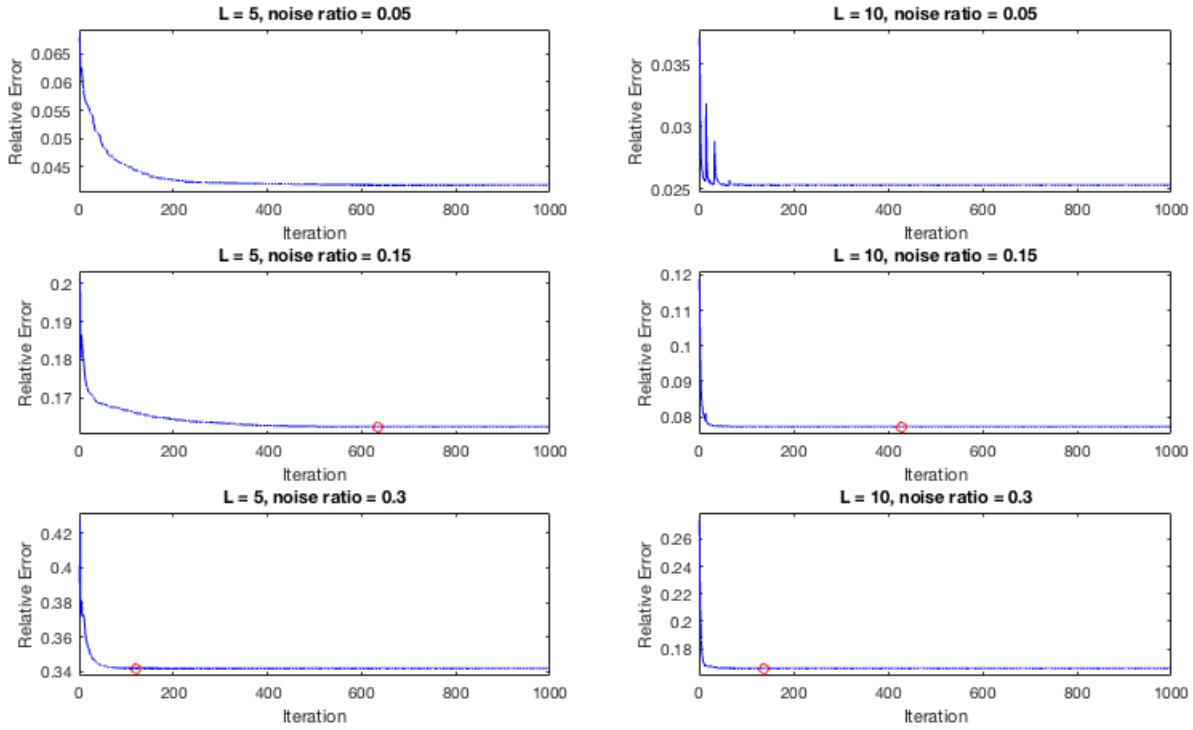

FIGURE 5.3. Signal relative errors (5.2.7a) for six PLGD models with Gaussian noise (5.2.3) solved with the GDD algorithm (Algorithm 3). The image in Figure 1.2 was resized to 64×64 pixels, made grayscale, and six models were generated with oversampling rates of $L = 5, 10$ and noise ratios $\epsilon_{\text{rel}} = 0.05, 0.15, 0.30$. These models were then solved using the GDD algorithm set to terminate after 1,000 iterations. The red circle (where present) indicates when the dual objective was determined nondifferentiable.

Figure 5.3 depicts the early progress and quick stagnation of the GDD algorithm for these PLGD models with Gaussian noise (5.2.3). For each model, virtually no measurable progress was made after iteration 500. Yet the point at which each model stagnates appears to differ, and in one case ($L = 10$ and $\epsilon_{\text{rel}} = 0.05$) the signal quality appears to vary greatly for neighboring iterates. Thus we examine the behavior in Figure 5.3 to identify the desired interval of iterates in which the GDD algorithm should terminate for each model.

The results in Figure 5.3 suggest that models with a larger oversampling rate and those with a larger noise ratio make progress faster and stagnate earlier. One indicator that the point of stagnation differs for these models is the point at which the GDD algorithm identifies the objective

function $\lambda_1(\mathcal{A}^* y)$ as nondifferentiable (i.e., the condition (4.2.3) fails). The models in Figure 5.3 with the least noise never identified a nondifferentiable objective, whereas the noisiest models identified nondifferentiable objectives very early (at iterate 122 for $L = 5$ and 137 for $L = 10$). Another indicator that the models of Figure 3 stagnate at different rates is the point at which the primal objective is found to be feasible, as described in Table 5.4.

	$L = 5$	$L = 10$
$\epsilon_{\text{rel}} = 0.05$	33	3
$\epsilon_{\text{rel}} = 0.15$	N/A	3
$\epsilon_{\text{rel}} = 0.30$	22	3

TABLE 5.4. Iterate at which the GDD algorithm (Algorithm 3) became primal feasible for models from Figure 5.3.

Table 5.4 shows that all three models with oversampling $L = 10$ found feasible signals after just 3 iterations, yet the models with $L = 5$ were a bit slower, and in particular the model with $L = 5$ and $\epsilon_{\text{rel}} = 0.15$ never identified a feasible signal.

Based on these observations, Table 5.5 proposes iterate intervals in which the GDD algorithm appears to have stagnated for each model in Figure 5.3, providing a guideline for assessing the new termination conditions (5.4.1) and (5.4.2).

	$L = 5$	$L = 10$
$\epsilon_{\text{rel}} = 0.05$	200-400	50-200
$\epsilon_{\text{rel}} = 0.15$	200-400	50-100
$\epsilon_{\text{rel}} = 0.30$	100-200	50-100

TABLE 5.5. Intervals of iterates at which the GDD algorithm (Algorithm 3) appears to stagnate for models from Figure 5.3.

In order to demonstrate that the primal difference (5.2.7e) and dual variable difference (5.2.7i) are accurate indicators of the stagnation of the GDD algorithm, Figure 5.4 depicts the behavior of these residuals for the models in Figure 5.3. Note that the vertical axis indicates specific tolerances

and the horizontal axis indicates the first iterate at which the GDD algorithm would satisfy this tolerance.

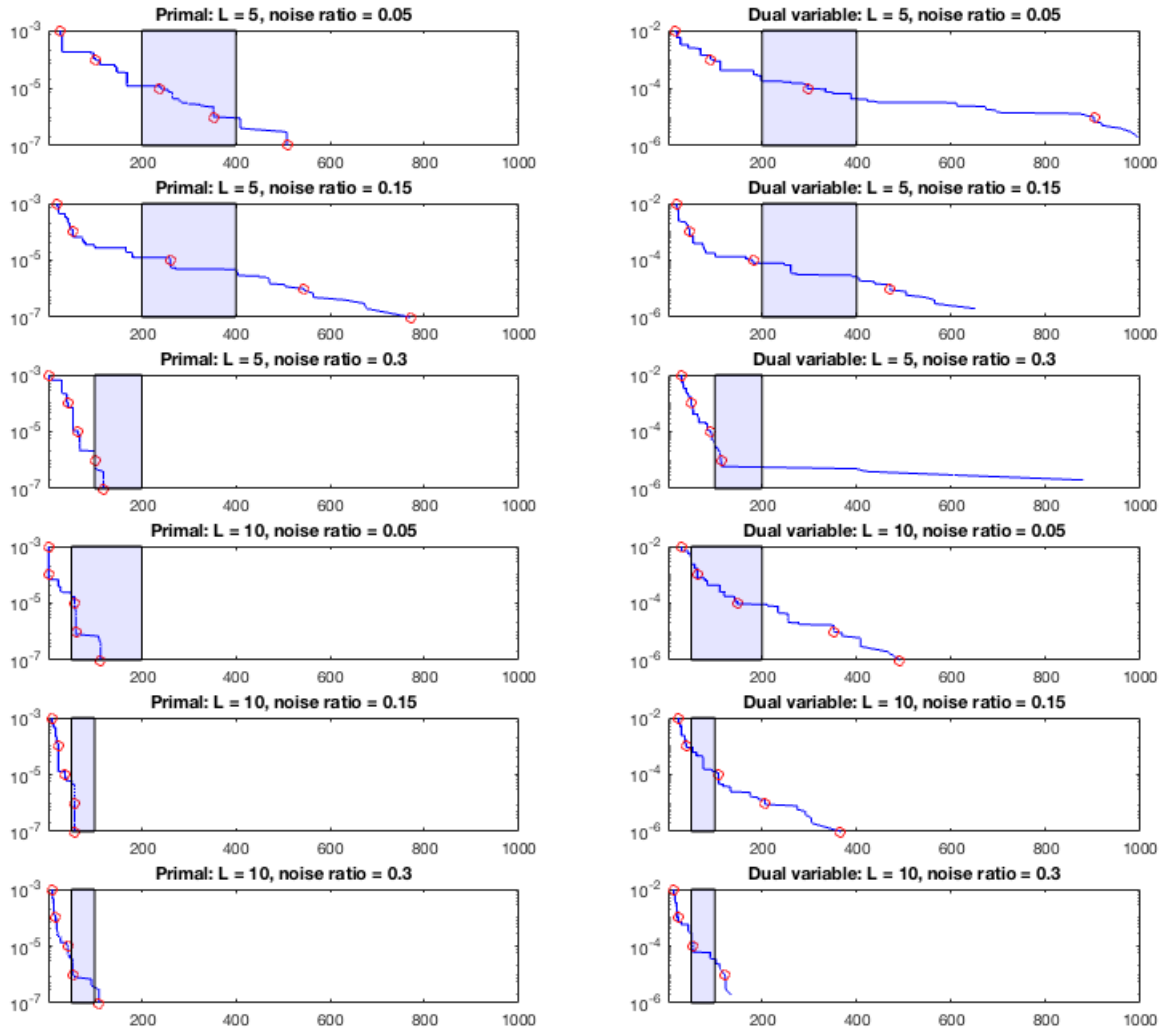

FIGURE 5.4. Plots of tolerance values against the iterate at which the GDD algorithm (Algorithm 3) first satisfies this tolerance for the models discussed in Figure 5.3. Tolerances depicted are difference values for the primal objective (5.2.7e) and dual variable (5.2.7i). Red circles are placed at tolerances 10^{-n}. The blue rectangles indicate the proposed intervals of termination from Table 5.5.

For each of the models in Figure 5.4, the termination conditions (5.4.1) and (5.4.2) are satisfied within or close to each respective interval of stagnation from Table 5.5. For the models with oversampling $L = 5$, the GDD algorithm satisfies the tolerances $\text{tol}_{\text{primal}} = 10^{-5}$ from (5.4.1) and

$\text{tol}_{\text{dual}} = 10^{-4}$ from (5.4.2) within or slightly prior to each respective desired interval. Likewise, the GDD algorithm achieve these desired tolerances for the models with $L = 10$ within or slightly after each desired interval. As an additional precaution, the maximum iteration count of 300 iterates will prevent running the GDD algorithm after it has stagnated.

Note that there is also theoretical justification for selecting the dual variable difference (5.2.7i) as a termination condition. Proposition 3.4.2 showed that the variable y is optimal for the PLGD model (3.2.1) if $y = \Pi_{\mathcal{C}}(y - \alpha g)$ for some $g \in \partial f(y)$ and all $\alpha > 0$. This property corresponds to the termination condition

$$||\Pi_{\mathcal{C}}(y - \alpha g) - y||_2 \leq \text{tol}.$$

If we make this condition relative by setting $\text{tol} = 10^{-4}||\Pi_{\mathcal{C}}(y - \alpha g)||_2$, then we recover the new dual variable difference condition (5.4.2).

One additional concern for termination conditions is the nonmonotonic nature of the GDD algorithm. In Figure 5.3, the model with $L = 10$ and $\epsilon_{\text{rel}} = 0.05$ demonstrates that the GDD algorithm may produce neighboring signal iterates with varying accuracy. Since this algorithm is nonmonotonic and relies on a subroutine to recover the current approximate signal, the accuracy of the sequence of recovered signals can vary dramatically. Thus we need a reliable indicator to select a sufficiently accurate signal among the previous iterates. Figure 5.5 depicts the signal relative error (5.2.7a) and duality gap (5.2.7f) for the $L = 10$, $\epsilon_{\text{rel}} = 0.05$ model. Note that the new termination conditions $\text{tol}_{\text{primal}} = 10^{-5}$ and $\text{tol}_{\text{dual}} = 10^{-4}$ would select the 148th iterate as the candidate solution.

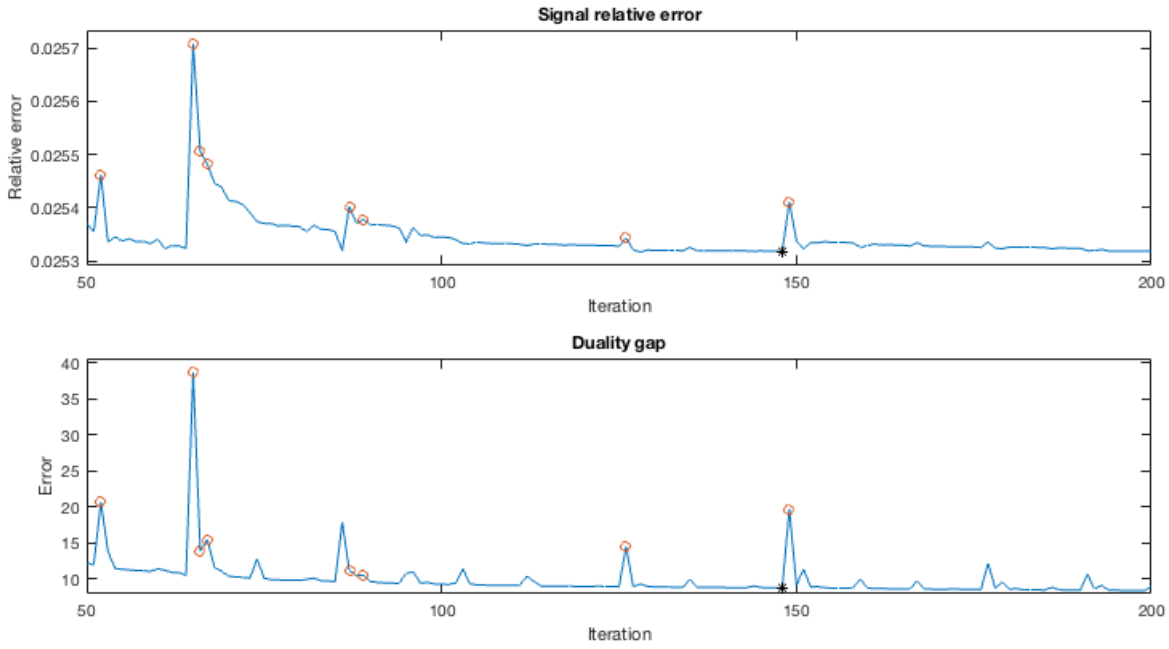

FIGURE 5.5. Signal relative error (5.2.7a) and duality gap (5.2.7f) for the model from Figure 5.3 with oversampling $L = 10$ and noise ratio $\epsilon_{\text{rel}} = 0.05$. Red circles indicate signals with relative error at least 0.05% above the mean of their ten neighbors, and the black asterisk indicates both the final iterate based on new termination conditions and the optimal iterate based on minimum duality gap.

Figure 5.5 demonstrates that the duality gap (5.2.7f) violation closely matches the signal relative error (5.2.7a). Among the previous iterates, those with the least accurate signal (indicated by the red circle) are accurately identified as having a duality gap value greater than the minimum (black asterisk). Thus once the GDD algorithm terminates, we select the signal among the last 20 with the lowest duality gap value.

Given the new termination conditions (5.4.1), (5.4.2), the maximum iteration count of 300, and the signal selection method discussed above, the Figure 5.6 depicts the iterate at which the GDD algorithm would terminate for each PLGD model with Gaussian noise from Figure 5.3.

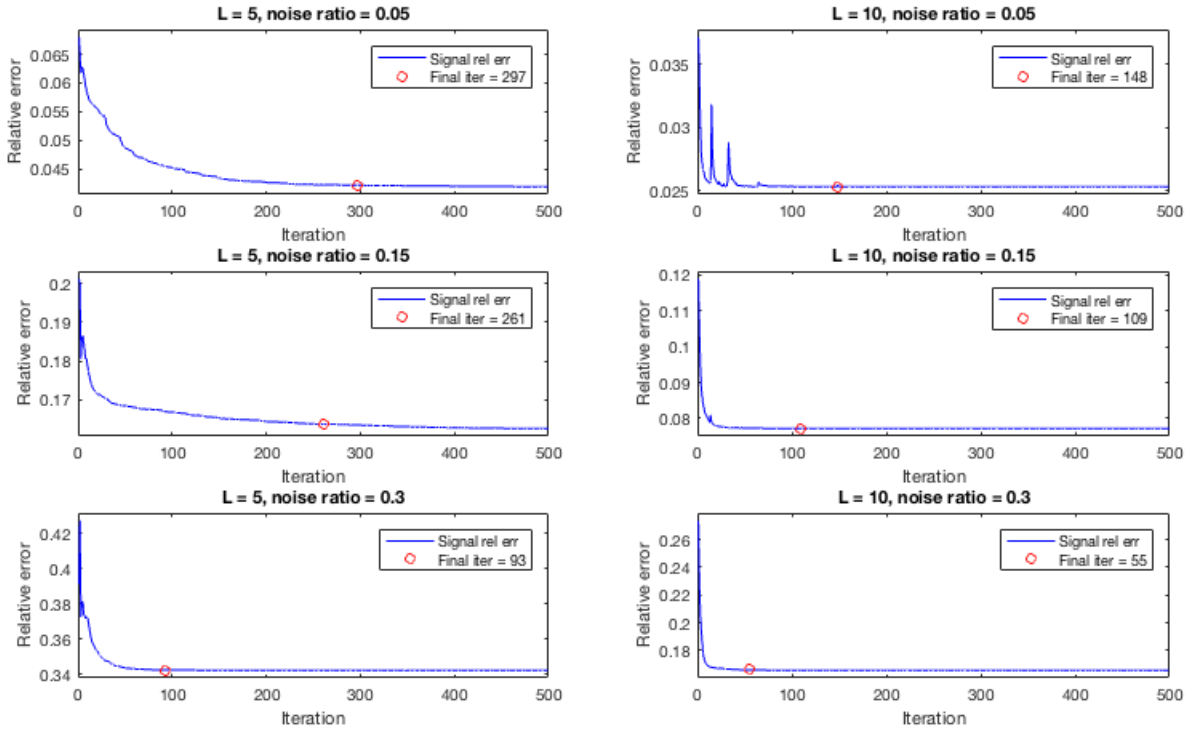

FIGURE 5.6. Iterate at which the GDD algorithm (Algorithm 3) terminates for the models from Figure 5.3 based on new termination conditions.

For each model in Figure 5.6, the GDD algorithm successfully terminates within or nearby the interval of stagnation established in Table 5.5. Given the new termination conditions (5.4.1) and (5.4.2), we may now treat the GDD algorithm as a black-box algorithm for noisy phase retrieval and examine the sequence of eigenvalue problems generated by the GDD algorithm.

CHAPTER 6

Evolving Matrix Eigenvalue Computation

6.1. Introduction

In this chapter we examine the *evolving matrix eigenvalue problem* (EMEP) in the GDD algorithm (Algorithm 3) and develop a new strategy for solving this problem.

Section 6.2 defines the EMEP and examines its computational costs and evolving spectrum across matrix iterates A_k. In particular, we observe that the algebraically largest eigenvalues tend to cluster as the algorithm proceeds, leading to more difficult eigenvalue problems for later matrix iterates. To handle these eigenvalue problems, Section 6.3 reviews the *implicitly restarted Arnoldi method* (IRAM), a modern Krylov subspace method established in [**62**] and examined thoroughly in [**3**, Chapter 5]. The convergence behavior of the IRAM is based on the spectrum of the given eigenvalue problem as well as the IRAM parameters chosen by the user. To understand and exploit the convergence behavior of the IRAM, Section 6.3 reviews the subroutines of the IRAM and summarizes relevant convergence results.

Next, Section 6.4 presents *the IRAM with adaptive parameter selection*, a new strategy for solving the EMEP by adaptively changing the number of requested eigenvalues r in the IRAM (Algorithm 7). Section 6.5 closely examines the EMEP for two PLGD models with Gaussian noise (5.2.3) to demonstrate an observed correlation between the clustering of the algebraically largest eigenvalues in the EMEP and an increase in the *empirically optimal* value $\bar{r}$ in the IRAM, which we define as the parameter $r_{min} \leq \bar{r} \leq r_{max}$ corresponding to the fewest matrix-vector products required for the IRAM to converge for a given EMEP iterate. We see that the IRAM with adaptive parameter selection properly tracks the empirically optimal value $\bar{r}$ for these models. Performance results for the IRAM with adaptive parameter selection are presented in Chapter 7. Also note that all EMEP (6.2.2) experiments in this chapter are performed with the GDD algorithm using

the new termination conditions (5.4.1) and (5.4.2) established in Chapter 5 and are available for reproduction.[1]

6.2. The Evolving Matrix Eigenvalue Problem

In this section we examine the sequence of eigenvalue problems in the GDD algorithm (Algorithm 3), which we define as the *evolving matrix eigenvalue problem* (EMEP). We will see that the EMEP is the most computationally expensive subroutine in the GDD algorithm. We also observe that the EMEP matrix iterates have a spectrum which evolves in a predictable way from early to later iterates.

6.2.1. EMEP Definition

We begin by formally defining the EMEP. Generally speaking, we are concerned with a sequence of eigenvalue problems in which each matrix is dependent on the results of the previous problem. For each iterate k in this sequence of problems, we have some Hermitian matrix iterate $A_k \in \mathcal{H}^n$ and seek its j algebraically largest eigenvalues $\Lambda_j^{(k)}$ and the corresponding eigenvectors $V_j^{(k)}$. The topic of eigenvalue problems with evolving matrices was recently studied in [**54**]. Some examples of this problem include subspace tracking in signal processing (see, e.g., [**19**], [**64**], [**68**], [**20**]), matrix completion (e.g., [**46**]), and the Kohn-Sham equation in density functional theory (e.g., [**55**]).

To define the EMEP, first note that each eigenvalue problem in the GDD algorithm (steps 2, 6, and 14) requires the two algebraically largest eigenvalues of the matrix iterate $A_k = \mathcal{A}^* y_k$, where $\mathcal{A}^*$ is defined as (2.4.2). In the GDD algorithm, the matrix A_0 is initialized as $A_0 = \mathcal{A}^* b$, where b is the observation vector in the PLGD model (3.2.1). For each iterate $k > 0$, the update matrix $A_k = \mathcal{A}^* y_k$ is computed with $y_k = \Pi_{\mathcal{C}}(y_{k-1} - \alpha_{k-1} g_{k-1})$, where the gradient $g_{k-1} = \mathcal{A}(v_1 v_1^*)$ is a function of the eigenvector v_1 corresponding to the algebraically largest eigenvalue of A_k, and the steplength α_{k-1} is determined using a linesearch on the minimization problem

$$\min_{\alpha} \quad \lambda_1 \left(\mathcal{A}^* (\Pi_{\mathcal{C}}(y_{k-1} - \alpha g_{k-1})) \right). \tag{6.2.1}$$

[1] `https://github.com/Will-Wright/low-rank-opt-rapid-eig`

Thus we define the sequence of eigenvalue problems generated by the GDD algorithm as the *evolving matrix eigenvalue problem* (EMEP)

$$\begin{aligned} &\text{for} \quad k = 0, 1, \ldots, K \\ &\text{find} \quad \left(\lambda_1^{(k)},\ v_1^{(k)}\right) \text{ and } \left(\lambda_2^{(k)},\ v_2^{(k)}\right) \text{ of } A_k, \end{aligned} \tag{6.2.2}$$

where $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$ are the two algebraically largest eigenvalues of the *matrix iterate* $A_k = \mathcal{A}^* y_k$, and y_k is the previous dual variable generated by the GDD algorithm (from step 2, 6, or 14). Note that y_k is dependent on $v_1^{(k-1)}$ and y_{k-1}, and thus each eigenvalue problem in the GDD algorithm is dependent on the previous matrix iterate A_{k-1}. Also note that the steplength α_{k-1} returned by the linesearch problem (6.2.1) influences the rate at which the sequence of matrices $A_0, A_1, \ldots$ evolves, since

$$\begin{aligned} ||A_k - A_{k-1}|| &= ||\mathcal{A}^*(\Pi_{\mathcal{C}}(y_{k-1} - \alpha_{k-1} g_{k-1})) - \mathcal{A}^* y_{k-1}|| \\ &= ||\mathcal{A}^*(\Pi_{\mathcal{C}}(y_{k-1} - \alpha_{k-1} g_{k-1}) - y_{k-1})|| \\ &\leq ||\mathcal{A}^*|| \cdot ||\Pi_{\mathcal{C}}(y_{k-1} - \alpha_{k-1} g_{k-1}) - y_{k-1}||_2 \\ &\leq ||\mathcal{A}^*|| \cdot ||y_{k-1} - \alpha_{k-1} g_{k-1} - y_{k-1}||_2 = ||\mathcal{A}^*|| \cdot |\alpha_{k-1}| \cdot ||g_{k-1}||_2, \end{aligned} \tag{6.2.3}$$

where we have the induced norm of $\mathcal{A}^*$

$$||\mathcal{A}^*|| = \sup_{||w||_2 = 1} ||\mathcal{A}^* w||.$$

6.2.2. Computational Costs

Next, we examine the overall computational costs of the EMEP (6.2.2). As discussed in Section 4.2, the main computational costs in the GDD algorithm (Algorithm 3) are the EMEP (step 2, 6, and 14), the primal refinement (step 11), and the dual refinement (step 13). Since we are focused on PLGD models with Gaussian noise (5.2.3), the dual refinement step of the GDD algorithm is skipped (see the end of Section 5.3 for an explanation). In both the EMEP (6.2.2) and the primal refinement step, the primary computational cost comes from $\mathcal{A}$-products (2.4.2), where each $\mathcal{A}(xx^*)$ product requires L DFTs and each $[\mathcal{A}^* y]x$ product requires $2L$ DFTs. Thus we measure computational costs in terms of the number of DFTs, following the convention of [**14**] and [**27**]. Also note that the computation of the eigenpair (λ_1, v_1) in the EMEP (6.2.2) must be very accurate in

order to determine an accurate descent step $g = \mathcal{A}(v_1 v_1^*)$ in the GDD algorithm. Table 6.1 depicts the number of DFTs in the GDD algorithm for a variety of noisy problems.

			EMEP			Primal refinement		All other steps	
n	L	ϵ_{rel}	`eigs` calls	Minutes	DFTs	Minutes	DFTs	Minutes	DFTs
4,096	5	0.05	228	13.13 (0.94)	51,935 (0.97)	0.73 (0.05)	1,516 (0.03)	0.04	17
4,096	5	0.15	120	6.63 (0.94)	31,085 (0.97)	0.45 (0.06)	1,076 (0.03)	0.01	10
4,096	5	0.30	52	3.56 (0.89)	16,410 (0.95)	0.45 (0.11)	854 (0.05)	0.01	4
4,096	10	0.05	190	12.06 (0.96)	72,587 (0.98)	0.45 (0.04)	1,819 (0.02)	0.03	29
4,096	10	0.15	106	8.60 (0.96)	51,450 (0.98)	0.30 (0.03)	1,194 (0.02)	0.02	17
4,096	10	0.30	111	17.95 (0.98)	107,936 (0.99)	0.36 (0.02)	1,420 (0.01)	0.01	18
16,384	5	0.05	199	46.09 (0.95)	69,745 (0.98)	2.13 (0.04)	1,468 (0.02)	0.06	16
16,384	5	0.15	91	27.71 (0.95)	41,880 (0.98)	1.34 (0.05)	853 (0.02)	0.03	8
16,384	5	0.30	61	30.95 (0.94)	45,834 (0.98)	2.04 (0.06)	1,026 (0.02)	0.02	5
16,384	10	0.05	160	56.73 (0.97)	92,391 (0.98)	1.64 (0.03)	1,560 (0.02)	0.07	25
16,384	10	0.15	103	36.30 (0.97)	60,189 (0.98)	1.21 (0.03)	1,167 (0.02)	0.05	17
16,384	10	0.30	47	18.48 (0.96)	30,498 (0.98)	0.65 (0.03)	617 (0.02)	0.02	8

TABLE 6.1. The GDD algorithm (Algorithm 3) runtime and number of DFTs (with percentage of the total in parentheses) for the EMEP (6.2.2), primal refinement (solving (4.2.11) in step 11) and all other operations. Here n is signal size (i.e., number of pixels squared in the image from Figure 1.2), L is number of observations, and ϵ_{rel} is the noise ratio.

The results in Table 6.1 demonstrate the essential computational challenges of the GDD algorithm. First, the EMEP is the dominant computational cost in the algorithm, and its proportion to other costs (in both runtime and number of DFTs) increases as the size of the model increases. Additionally, the primal refinement step requires a small but nontrivial amount of computation. All other operations accounted for 0.00% of the overall runtime.

We will now profile the cost of the EMEP (6.2.2). Figure 6.1 depicts the number of matrix-vector products $[\mathcal{A}^* y_k]x$ in the EMEP (6.2.2) for each of the six smaller models from Table 6.1.

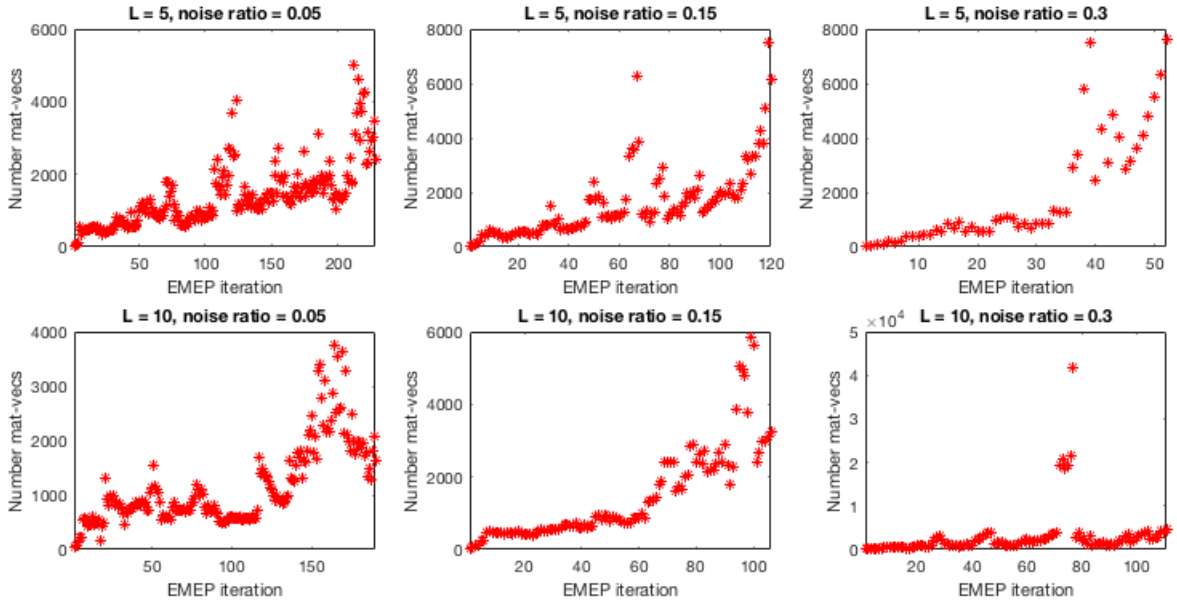

FIGURE 6.1. Number of matrix-vector products for each iteration in the EMEP (6.2.2) for the six smaller models from Table 6.1 .

Figure 6.1 demonstrates that the number of matrix-vector products required for each EMEP (6.2.2) iterate varies greatly from earlier to later iterates. In each model, the later iterates account for the majority of the computational cost of the EMEP (6.2.2). Additionally, some iterates require far more matrix-vector products than others (e.g., the iterates around $k = 75$ in the bottom-right plot). To help explain this change in computational costs, we now proceed to examine how the spectrum of the EMEP evolves.

6.2.3. Evolving Spectrum Distribution

We close this section by examining the evolving spectrum distribution of the EMEP (6.2.2). We find that the algebraically largest eigenvalues begin to cluster for later EMEP iterates, likely causing those eigenvalue problems to be more difficult. First, we examine the full spectrum of a few EMEP matrix iterates in Figure 6.2.

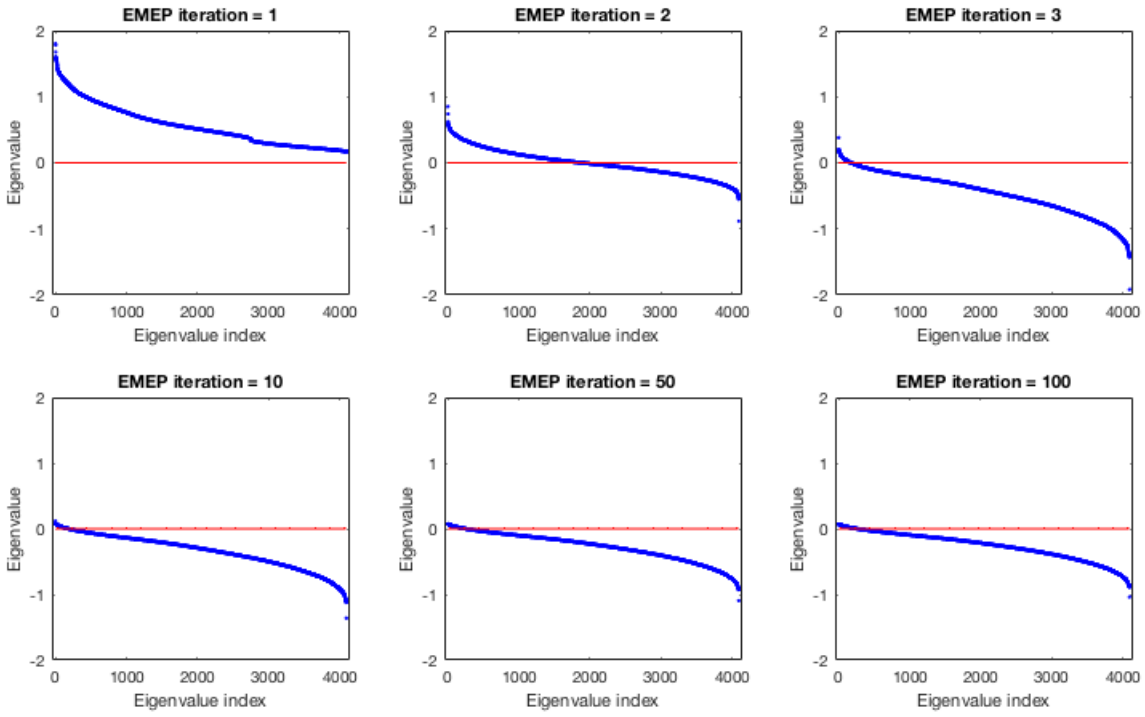

FIGURE 6.2. Spectrum of specific EMEP (6.2.2) matrix iterates A_k for the model from Table 6.1 with signal size $n = 4,096$, oversampling $L = 5$, and noise ratio $\epsilon_{\text{rel}} = 0.15$.

As we see in Figure 6.2, the spectrum of the matrix iterates A_k in the EMEP (6.2.2) shifts from completely positive for A_1 to mostly negative for later iterates. This shift in spectrum is a consequence of optimizing the PLGD model with Gaussian noise (5.2.3). The first matrix iterate $A_1 = \mathcal{A}^* b$ will always be positive-semidefinite because the components of the observation $b = [b_1; b_2; \ldots; b_L]$ are all nonnegative and thus for all x we have

$$x^*[\mathcal{A}^* b]x = \sum_{j=1}^{L} [FC_j x]^* \text{Diag}(b_j) FC_j x \geq 0.$$

Since the GDD algorithm minimizes the objective function $\lambda_1(\mathcal{A}^* y_k)$, the algebraically largest eigenvalue $\lambda_1^{(k)}$ of $\mathcal{A}^* y_k$ can be expected to decrease for later iterates k.

As we will see in Section 6.3, the convergence rate of eigenvalue methods often depends on the distance between the desired eigenvalue λ_j and the next algebraically largest eigenvalue λ_{j+1}.

Figure 6.3 depicts the 20 algebraically largest eigenvalues of the EMEP (6.2.2) iterates from Figure 6.2.

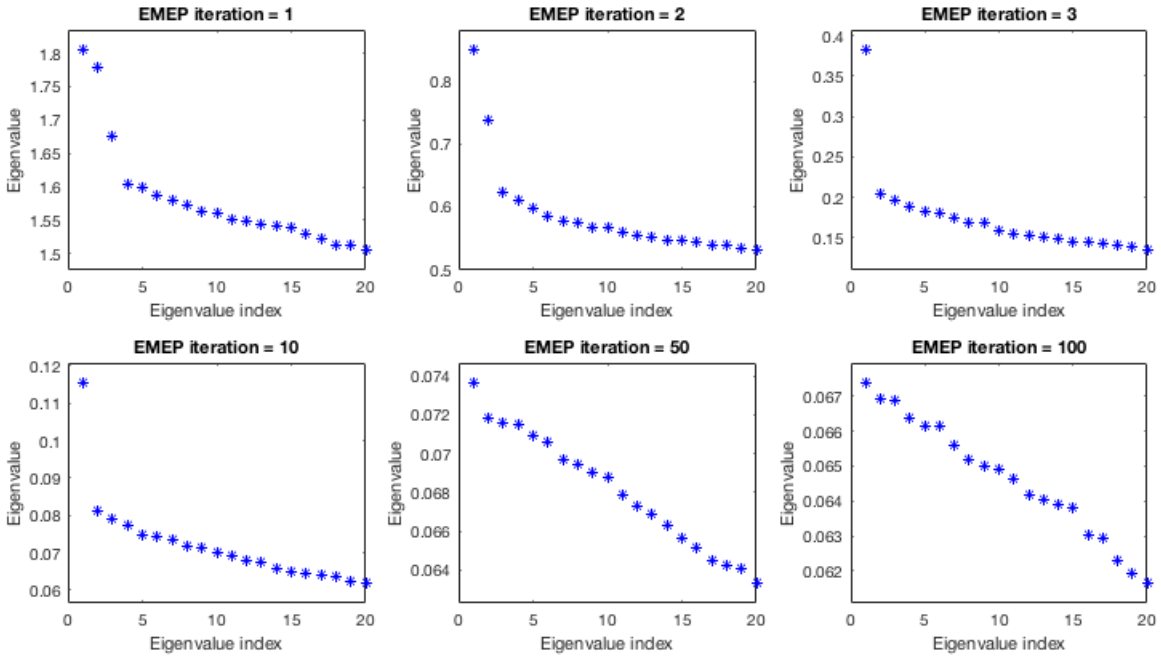

FIGURE 6.3. Twenty algebraically largest eigenvalues of specific EMEP (6.2.2) matrix iterates A_k for the model from Table 6.1 with signal size $n = 4,096$, oversampling $L = 5$, and noise ratio $\epsilon_{\text{rel}} = 0.15$.

Figure 6.3 demonstrates that the algebraically largest eigenvalues of the matrix iterates A_k cluster together as the EMEP (6.2.2) progresses. In general, this clustering in the EMEP (6.2.2) spectrum is expected. Section 5.3 demonstrated that PLGD models with Gaussian noise (5.2.3) typically have optimal primal matrices $X_\star$ with rank greater than one (see Table 5.3). And Corollary 3.4.1 (e) indicates that $\text{rank}(X_\star)$ is a lower bound on the multiplicity r of the algebraically largest eigenvalue of the optimal dual matrix $\mathcal{A}^* y_\star$. Thus for later EMEP (6.2.2) iterates k, we can expect some r algebraically largest eigenvalues $\lambda_1^{(k)}, \lambda_2^{(k)}, \ldots, \lambda_r^{(k)}$ to have a decreasing relative difference

$$\frac{\lambda_i^{(k)} - \lambda_{i+1}^{(k)}}{\lambda_i^{(k)}}$$

for $i = 1, 2, \ldots, r-1$.

The clustering of the algebraically largest eigenvalues as depicted in Figure 6.3 is known to slow the convergence rate of modern Krylov subspace methods. Yet the choice of the parameters in these Krylov subspace methods can also affect their convergence rate significantly. A thorough understanding of one such method (the implicitly restarted Arnoldi method) and its subroutines will help us to develop a new, adaptive strategy for choosing parameters for the EMEP (6.2.2) in Section 6.4.

6.3. The Implicitly Restarted Arnoldi Method

In this section we review the *implicitly restarted Arnoldi method* (IRAM), a common large-scale eigenvalue method. First proposed by Sorensen [**62**], [**63**], the IRAM is a combination of two essential algorithms. The *m-step Arnoldi iteration* is used to build a matrix Q_m of m basis vectors which approximates the desired eigenspace. The *p-step shifted QR iteration* restarts the matrix Q_m with a specific strategy to damp unwanted eigenvalues, resulting in a smaller matrix Q_j of $j < m$ basis vectors. Since the m-step Arnoldi iteration is an extension of the *power method*, we first discuss the Power method before developing the IRAM. Altogether, the algorithms in this section are presented in the order depicted in Figure 6.4.

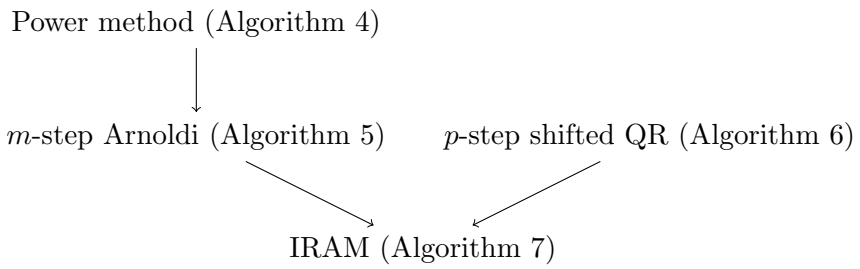

FIGURE 6.4. Dependency chart for the IRAM.

This section follows the treatment found in [**31**, Chapters 8, 10], with occasional minor changes in notation.

6.3.1. The Power Method

The first method we consider is the *power method*, a method for determining the largest magnitude eigenvalue λ_1 and corresponding eigenvector v_1 of a Hermitian matrix A. The power method

is based on the property that if λ_1 is strictly larger in magnitude than the next largest magnitude eigenvalue and the initial vector $q^{(0)}$ has a nonzero component in the direction of v_1 (i.e., $v_1^* q^{(0)} \neq 0$), then the sequence

$$q^{(0)}, \frac{Aq^{(0)}}{||Aq^{(0)}||_2}, \frac{A^2q^{(0)}}{||A^2q^{(0)}||_2}, \frac{A^3q^{(0)}}{||A^3q^{(0)}||_2}, \dots$$

will have v_1 as its limit.

Formally, Algorithm 4 presents the power method as seen in [**31**, Section 8.2.1].

Algorithm 4 Power method

Input: Hermitian matrix A, initial approximate eigenvector $q^{(0)}$, relative tolerance $\text{tol}_{\text{rel}} > 0$.

Output: Approximate largest magnitude eigenvalue λ and the corresponding eigenvector v.

Initialize: $q^{(0)} = q^{(0)}/||q^{(0)}||_2$, $\rho^{(0)} = [q^{(0)}]^* A q^{(0)}$, $r^{(0)} = Au^{(0)} - \rho^{(0)} q^{(0)}$, $i = 1$.
while *not converged:* $||r^{(i)}||_2/(||Aq^{(i)}||_2 + |\rho^{(i)}|) > \text{tol}_{\text{rel}}$ **do**
    $z^{(i)} = Aq^{(i-1)}$
    $q^{(i)} = z^{(i)}/||z^{(i)}||_2$
    $\rho^{(i)} = [q^{(i)}]^* z^{(i)}$
    $r^{(i)} = Aq^{(i)} - \rho^{(i)} q^{(i)}$, $i = i + 1$
end while
Return: $(\lambda, v) = (\rho^{(i-1)}, q^{(i-1)})$.

The simplicity of power method allows for insightful convergence results like the Theorem 6.3.1, in which we assume the matrix A is real for clarity.

THEOREM 6.3.1. *Suppose $A \in \mathbb{R}^{n \times n}$ is symmetric with an eigenvalue decomposition*

$$V^* A V = Diag(\lambda_1, \lambda_2, \dots, \lambda_n),$$

where $V = [\ v_1 \mid v_2 \mid \cdots \mid v_n\ ]$ is orthogonal and $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Let the vectors $q^{(i)}$ be generated by Algorithm 4 and define $\theta_i \in [0, \pi/2]$ as

$$\cos(\theta_i) = \left| v_1^T q^{(i)} \right|.$$

If $\cos(\theta_0) \neq 0$*, then for* $i = 0, 1, \ldots$ *we have*

$$|\sin(\theta_i)| \leq \tan(\theta_0) \left|\frac{\lambda_2}{\lambda_1}\right|^i, \tag{6.3.1}$$

$$\left|\lambda^{(i)} - \lambda_1\right| \leq \max_{2 \leq j \leq n} |\lambda_1 - \lambda_i| \tan(\theta_0)^2 \left|\frac{\lambda_2}{\lambda_1}\right|^{2i}. \tag{6.3.2}$$

PROOF. See [**31**, Theorem 8.2.1]. □

Theorem 6.3.1 establishes that the convergence rate of the power method (Algorithm 4) is dependent on the distance between $|\lambda_1|$ and $|\lambda_2|$. If this distance $\epsilon = |\lambda_1| - |\lambda_2|$ is very small relative to $|\lambda_1|$, then we have

$$\left|\frac{\lambda_2}{\lambda_1}\right| = \frac{|\lambda_1| - \epsilon}{|\lambda_1|} = 1 - \frac{\epsilon}{|\lambda_1|} \approx 1,$$

and $|\sin(\theta_i)|$ in (6.3.1) may decreases very slowly.

6.3.2. The m-Step Arnoldi Iteration

The next method we consider is the *m-step Arnoldi iteration* which extends the power method (Algorithm 4) to achieve a superior convergence rate. An iteration of the power method generates a new approximate eigenvector ($q^{(i)}$ from steps 3 and 4) by normalizing the matrix-vector product of the previous vector. In essence, the power method searches for the largest magnitude eigenvalue λ_1 and corresponding eigenvector v_1 of a matrix A in the one-dimensional subspace $V = \text{span}\{Aq_1\}$. The m-step Arnoldi iteration extends the power method by searching for the Ritz pair (1.0.12) (θ_1, u_1) for A with respect to the m-dimensional *Krylov subspace*

$$\mathcal{K}_m(A, q_1) = \text{span}\{q_1, Aq_1, A^2q_1, \ldots, A^{m-1}q_1\}. \tag{6.3.3}$$

Algorithm 5 (as described in [**31**, Algorithm 10.5.1]) builds a unitary basis Q_m of $\mathcal{K}_m(A, q_1)$ which may be used to find the Ritz pair (θ_1, u_1).

Algorithm 5 m-step Arnoldi iteration

Input: Matrix $A \in \mathbb{C}^{n\times n}$, number of Arnoldi steps m, initial approximate eigenvector q_1.

Output: Hessenberg matrix H_m, basis Q_m, residual r_m.

1: *Initialize:* $q_1 = q_1/||q_1||_2, \quad z = Aq_1, \quad \alpha_1 = q_1^* z, \quad r_1 = z - \alpha_1 q_1, \quad Q_1 = [q_1], \quad H_1 = [\alpha_1]$.

2: **for** $i = 1, \ldots, m-1$ **do**

3: $\quad \beta_i = ||r_i||_2, \quad q_{i+1} = r_i/\beta_i$.

4: $\quad Q_{i+1} = [Q_i \mid q_{i+1}], \quad \hat{H}_i = \begin{bmatrix} H_i \\ \beta_i e_i^T \end{bmatrix}$.

5: $\quad z = Aq_{i+1}$.

6: $\quad h = Q_{i+1}^* z, \quad r_{i+1} = z - Q_{i+1} h$.

7: $\quad H_{i+1} = [\hat{H}_i \mid h]$.

8: **end for**

9: *Return:* H_m, Q_m, r_m.

In order to obtain a Ritz pair (θ, u) for A with respect to $\mathcal{K}_m(A, q_1)$, the m-step Arnoldi iteration generates an *m-step Arnoldi decomposition*

$$AQ_m = Q_m H_m + r_m e_m^*, \tag{6.3.4}$$

where H_m is an upper Hessenberg matrix. If (θ, w) is an eigenpair for H_m and $u = Q_m w$ then (6.3.4) implies

$$(AQ_m - Q_m H_m)w = (A - \theta I)u = (e_m^* w) r_m. \tag{6.3.5}$$

Additionally, steps 5 and 6 of Algorithm 5 indicate that r_m is orthogonal to $\mathcal{K}_m(A, q_1)$, and thus (θ, u) is a Ritz pair for A with respect to $\mathcal{K}_m(A, q_1)$.

The use of $\mathcal{K}_m(A, q_1)$ in Algorithm 5 allows for superior convergence to Algorithm 4. Note that the largest magnitude Ritz pair (θ_1, u_1) for A with respect to $\mathcal{K}_m(A, q_1)$ generated by Algorithm 5 is guaranteed to be at least comparable to the m-th iterate of Algorithm 4 since $A^{m-1} q_1 \in \mathcal{K}_m(A, q_1)$. To compare the convergence rates of Algorithms 4 and 5 more precisely, assume the matrix A in real and symmetric. Then the matrix H_m returned by Algorithm 5 is tridiagonal and this algorithm is equivalent to the *m-step Lanczos iteration* [**31**, Algorithm 10.1.1]. In this case, we

have the Theorem 6.3.2. Note that this theorem involves *Chebyshev polynomials* [**53**, Section 4.4], a sequence of polynomials defined recursively as

$$c_k(x) = 2xc_{k-1}(x) - c_{k-2}(x) \tag{6.3.6}$$

for $k \geq 2$, with $c_0 = 1$ and $c_2 = x$.

THEOREM 6.3.2. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric with an eigenvalue decomposition*

$$V^*AV = Diag(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

where $V = [\ v_1 \mid v_2 \mid \cdots \mid v_n\ ]$ is orthogonal and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Suppose the m-step Arnoldi iteration (Algorithm 5) is performed and H_k is the tridiagonal matrix returned by this algorithm. If θ_1 is the algebraically largest eigenvalue of H_m, then

$$\lambda_1 \geq \theta_1 \geq \lambda_1 - (\lambda_1 - \lambda_n)\left(\frac{\tan(\phi_1)}{c_{m-1}(1+2\rho_1)}\right)^2, \tag{6.3.7}$$

where $\cos(\phi_1) = |q_1^T v_1|$,

$$\rho_1 = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}, \tag{6.3.8}$$

and $c_{m-1}(x)$ is the Chebyshev polynomial of degree $m-1$.

PROOF. See [**31**, Theorem 10.1.2]. □

The convergence rate established in Theorem 6.3.2 may also be applied to Algorithm 4, giving the Corollary 6.3.3.

COROLLARY 6.3.3. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric and positive semidefinite with an eigenvalue decomposition*

$$V^*AV = Diag(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

where $V = [\ v_1 \mid v_2 \mid \cdots \mid v_n\ ]$ is orthogonal and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Suppose m steps of the power method (Algorithm 4) are performed and $\gamma_1 = \rho^{(m)}$ is the returned Ritz value. Then

$$\lambda_1 \geq \gamma_1 \geq \lambda_1 - (\lambda_1 - \lambda_n)\tan^2(\phi_1)\left(\frac{\lambda_2}{\lambda_1}\right)^{2(m-1)}, \tag{6.3.9}$$

where $\cos(\phi_1) = |q_1^T v_1|$.

PROOF. See [**31**, Theorem 10.1.2] and replace the Chebyshev polynomial in this proof with $p(x) = x^{k-1}$. □

The lower bounds in Theorem 6.3.2 and Corollary 6.3.3 may be used to compare the expected convergence rates for Algorithms 4 and 5. The following comparison is based on [**31**, Section 10.1.6]. Assume $A \in \mathbb{R}^{n\times n}$ is symmetric and also positive semidefinite for clarity. Assume Algorithms 4 and 5 have been run for m steps with the same initial vector q_1. Let $\gamma_1 = \rho^{(m)}$ be the Ritz value for A generated by step 5 of Algorithm 4. And let θ_1 be the Ritz value for A with respect to $\mathcal{K}_m(A, q_1)$ generated by the algebraically largest eigenvalue of H_m from Algorithm 5. Then we may compare the lower bounds (6.3.9) for γ_1 and (6.3.7) for θ_1 by comparing the values

$$P_{m-1} = \left(\frac{\lambda_2}{\lambda_1}\right)^{2(m-1)}, \tag{6.3.10}$$

$$L_{m-1} = \frac{1}{\left[c_{m-1}\left(2\frac{\lambda_1}{\lambda_2} - 1\right)\right]^2} \geq \frac{1}{\left[c_{m-1}\left(1 + 2\rho_1\right)\right]^2}. \tag{6.3.11}$$

Table 6.2 compares P_{m-1} and L_{m-1} for a few values of m and λ_1/λ_2.

	$m = 10$		$m = 20$	
λ_1/λ_2	P_{m-1}	L_{m-1}	P_{m-1}	L_{m-1}
1.10	1.8×10^{-1}	5.5×10^{-5}	2.7×10^{-2}	2.1×10^{-10}
1.01	8.4×10^{-1}	1.0×10^{-1}	6.9×10^{-1}	2.0×10^{-3}

TABLE 6.2. Lower bound terms (6.3.10) and (6.3.11) for Ritz values generated by Algorithms 4 and 5

Table 6.2 demonstrates that the use of the Krylov subspace $\mathcal{K}_m(A, q_1)$ in Algorithm 5 allows for superior convergence to Algorithm 4. Yet this superior convergence rate is slowed somewhat when the desired eigenvalue λ_1 is close to λ_2. For eigenvalue problems where the value λ_1/λ_2 is very small (like later iterates of the EMEP (6.2.2), as demonstrated in Figure 6.3), we may seek to increase the number of steps m in Algorithm 5. Yet increasing m can be computationally

prohibitive if the eigenvalue problem is very large, requiring significant memory to store Q_m and significant computation to compute the eigenvalue decomposition of H_m.

6.3.3. The p-Step Shifted QR Iteration

To take advantage of the convergence rate of Algorithm 5 for larger eigenvalue problems, the Arnoldi decomposition (6.3.4) may be restarted with the *p-step shifted QR iteration* developed by [**62**] and discussed in [**31**, Sections 10.5.2-3]. To develop this algorithm, assume we are seeking the j algebraically largest eigenvalues of a Hermitian matrix $A \in \mathbb{C}^{n\times n}$ and we require that the m-step Arnoldi decomposition (6.3.4) $AQ_m = Q_m H_m + r_m e_m^*$ has a fixed size $m > j$. First we run Algorithm 5 with the initial vector q_1 to obtain $AQ_m = Q_m H_m + r_m e_m^*$. Next, recall that the matrix H_m may be used to identify the desired Ritz pairs $\{(\theta_i, u_i)\}_{i=1}^{j}$ for A with respect to $\mathcal{K}_m(A, q_1)$, as described in (6.3.5). Yet H_m also contains Ritz values $\theta_{j+1}, \ldots, \theta_m$ which correspond to unwanted eigenvalues of A. To damp these unwanted Ritz values, we may select an appropriate degree $p = m - j$ filter polynomial $p(\lambda)$. The p-step shifted QR iteration uses the filter polynomial

$$p(\lambda) = c \cdot (\lambda - \mu_1)(\lambda - \mu_2) \cdots (\lambda - \mu_p), \tag{6.3.12}$$

where c is a constant and the shift values $\mu_1 = \theta_{j+1}, \ldots, \mu_p = \theta_m$ are the p unwanted Ritz values of A with respect to $\mathcal{K}_m(A, q_1)$. Algorithm 6 (as described in [**31**, Section 10.5.3]) uses the filter polynomial (6.3.12) implicitly by applying p shifted QR steps to H_m.

Algorithm 6 p-step shifted QR iteration (implicit polynomial filtering)

Input: Hessenberg matrix $H_m \in \mathbb{C}^{m\times m}$ and shift values $\mu_1, \ldots, \mu_p$.

Output: Processed Hessenberg matrix $H_m^+ \in \mathbb{C}^{m\times m}$ and change of basis $V \in \mathbb{C}^{m\times m}$, with $H_m^+ = V^* H_m V$.

1: Set $H^{(0)} = H_m$.
2: **for** $i = 1, \ldots, p$ **do**
3: *QR factorization:* $H^{(i-1)} - \mu_i I = V_i R_i$.
4: *Update:* $H^{(i)} = R_i V_i + \mu_i I$.
5: **end for**
6: Set $H_m^+ = H^{(p)}$, $V = V_1 V_2 \cdots V_p$.
7: *Return:* H_m^+, V.

Proposition 6.3.1 establishes that Algorithm 6 implicitly applies the filter polynomial $p(\lambda)$ from (6.3.12) to the initial vector q_1 used to create the m-step Arnoldi decomposition (6.3.4).

PROPOSITION 6.3.1. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian and $AQ_m = Q_m H_m + r_m e_m^*$ be the m-step Arnoldi decomposition (6.3.4) returned by Algorithm 5 with initial vector q_1. And let $\mu_1, \ldots, \mu_p$ be the p smallest algebraic eigenvalues of H_m. Run Algorithm 6 with H_m and $\mu_1, \ldots, \mu_p$ as inputs, and return H_m^+, $V = V_1 \cdots V_p$, and $R = R_p \cdots R_1$.*

Then the restarted matrix $Q_+ = Q_m V$ will have the first column

$$q_+ = Q_m V(:, 1) = p(A) q_1,$$

where $p(\lambda)$ is the filter polynomial (6.3.12) with constant $c = 1/R(1,1)$.

PROOF. See [**31**, Section 10.5.3]. □

After performing Algorithm 6, we have the transformed m-step Arnoldi decomposition (6.3.4)

$$AQ_+ = Q_+ H_+ + r_m e_m^* V, \tag{6.3.13}$$

where $V = V_1 \cdots V_p$ from Algorithm 6 and $Q_+ = Q_m V$. As a consequence of the QR steps used in Algorithm 6, we can also show that the first $j = m - p$ columns of (6.3.13) form a new j-step

Arnoldi decomposition. Note that $V_1, \ldots, V_p$ are all upper Hessenberg due to the QR factorization in step 3 of Algorithm 6. Then V has a lower band p and $V(m, 1 : m-p-1) = V(m, 1 : j-1) = 0$, giving

$$r_m e_m^* V(:, 1 : j) = V(m, j) r_m e_j^*. \tag{6.3.14}$$

Also, H_+ is upper Hessenberg and thus $H_+(j+1 : m, 1 : j) = H_+(j+1, j) e_1 e_j^*$, giving

$$\begin{aligned} Q_+ H_+(:, 1 : j) &= Q_+(:, 1 : j) H_+(1 : j, 1 : j) + Q_+(:, j+1 : m) H_+(j+1 : m, 1 : j) \\ &= Q_+(:, 1 : j) H_+(1 : j, 1 : j) + H_+(j+1, j) Q_+(:, j+1) e_j^*. \end{aligned} \tag{6.3.15}$$

Therefore, if we set $Q_j = Q_+(:, 1 : j) = Q_m V(:, 1 : j)$, $H_j = H_+(1 : j, 1 : j)$, and $r_j = V(m, j) r_m + H_+(j+1, j) Q_+(:, j+1)$, then equations (6.3.13-6.3.15) give the new j-step Arnoldi decomposition

$$\begin{aligned} AQ_j &= AQ_+(:, 1 : j) \\ &= Q_+(:, 1 : j) H_+(1 : j, 1 : j) + \left[V(m, j) r_m + H_+(j+1, j) Q_+(:, j+1) \right] e_j^* \\ &= Q_j H_j + r_j e_j^*, \end{aligned} \tag{6.3.16}$$

and we may resume Algorithm 5 at step $j+1$.

6.3.4. The Implicitly Restarted Arnoldi Method

Combining Algorithms 5 and 6 as described above, we have Algorithm 7 as presented in [**31**, Section 10.5.3].

Algorithm 7 Implicitly restarted Arnoldi method (IRAM)

Input: Matrix $A \in \mathbb{C}^{n\times n}$, initial approximate eigenvector q_1, number of requested algebraically largest eigenvalues j, maximum Arnoldi decomposition (6.3.4) size m.

Output: Approximate algebraically largest eigenpairs (Λ_j, V_j).

1: *Initialize with Algorithm 5:* Perform the m-step Arnoldi iteration with initial vector q_1 to obtain $AQ_m = Q_m H_m + r_m e_m^*$.

2: **while** not converged **do**

3: Compute the eigenvalues $\theta_1, \ldots, \theta_m$ of H_m and identify the $p = m - j$ (unwanted) shift values $\mu_1 = \theta_{j+1}, \ldots, \mu_p = \theta_m$.

4: *Algorithm 6:* Perform the p-step shifted QR iteration to obtain the Hessenberg matrix H_+ and change of basis V.

5: *Restart the Arnoldi factorization:* Set $Q_j = Q_m V(:, 1:j)$, $H_j = H_+(1:j, 1:j)$, and $r_j = V(m, j) r_m + H_+(j+1, j) Q_+(:, j+1)$ per (6.3.16).

6: *Algorithm 5:* Beginning with $AQ_j = Q_j H_j + r_j e_j^*$, perform steps $j+1, \ldots, m$ of the Arnoldi iteration to obtain $AQ_m = Q_m H_m + r_m e_m^*$.

7: **end while**

8: Compute the j algebraically largest eigenvalues $\Lambda_j = \{\lambda_1, \ldots, \lambda_j\}$ and corresponding eigenvectors $u_1, \ldots, u_j$ of H_m. Set $V_j = [\ Q_m u_1 \mid \cdots \mid Q_m v_j\ ]$.

9: *Return:* (Λ_j, V_j).

The IRAM is the eigenvalue method we use in Section 6.4 to handle the EMEP (6.2.2). The choice of parameters m (the Arnoldi decomposition size) and j (number of requested eigenvalues) can greatly impact the efficiency of IRAM (see Section 6.4). For many large-scale eigenvalue problems, the IRAM is a very effective and convenient method. Due to the implicit polynomial filtering in step 4 of IRAM, this method is particularly effective when the j algebraically largest eigenvalues have modest separation from λ_{j+1}. And since the IRAM only has two parameter choices, there is little optimization required by the user.

However, when $\lambda_j \approx \lambda_{j+1}$ and the Arnoldi decomposition size m is not sufficiently large, the IRAM may require many iterations to achieve the desired tolerance. As we will see in Section 6.4,

the appropriate choice of m and j in this circumstance may make the IRAM far more competitive. Yet choosing m and j without prior knowledge of the eigenvalue distribution is inherently heuristic. Additionally, if the inputted matrix A is very large, then it may be prohibitive to store the $Q_m \in \mathbb{C}^{n\times m}$ in active memory. In particular, if the image or signal x being recovered in the PLGD problem has n pixels, then Q_m will require $m-$times as much storage space. Thus we proceed in the next section by considering an alternative Krylov subspace method which does not require parameter tuning, nor a large subspace to be held in memory.

Note that the IRAM is implemented in the numerical software package `ARPACK` (the ARnoldi PACKage) in FORTRAN 77 [**39**]. Many numerical computing environments include large-scale eigenvalue methods which having bindings to ARPACK, including `eigs` in MATLAB, `eigs` and `eigsh` in the Python package SciPy, `eigs` in R, and `eigs` in the Julia package Arpack.jl.

6.4. A New Strategy for the EMEP

In this section we develop a new strategy which uses the IRAM (Algorithm 7) to handle the EMEP (6.2.2) while adaptively changing one of the IRAM parameters based on the results from the previous EMEP iterates. As discussed in Section 6.3, the IRAM has only two key parameters: the number of requested eigenvalues r and the Arnoldi decomposition (6.3.4) size m. By fixing m at an appropriate size and employing an adaptive strategy for choosing r, we may greatly reduce the number of matrix-vector products required for the EMEP. To compare various choices of r, we define the *empirically optimal* parameter $\bar{r}$ as follows.

Definition 6.4.1. The *empirically optimal* parameter $\bar{r}$ is the integer $r_{min} \leq \bar{r} \leq r_{max}$ corresponding to the fewest matrix-vector products required for the IRAM to converge for a given EMEP iterate and fixed parameter m.

We begin by examining the change in computational costs (as measured by matrix-vector products) with respect to various EMEP (6.2.2) iterates k and IRAM parameters r in Figure 6.5. In the original implementation of the GDD algorithm (Algorithm 3), all EMEP iterates were handled using the IRAM with $r = 2$ requested eigenvalues and Arnoldi decomposition size $m = \min\{\max\{2r, 20\}, n\}$, where n is the size of the desired signal x. This choice of m is equivalent to the default parameter setting in the IRAM solver `eigs` for MATLAB and evaluates to $m = 20$

for $n \geq 20$. Yet the plots in Figure 6.5 demonstrate that choosing a fixed parameter $r = 2$ can result in far more matrix-vector products than if we chose the empirically optimal values.

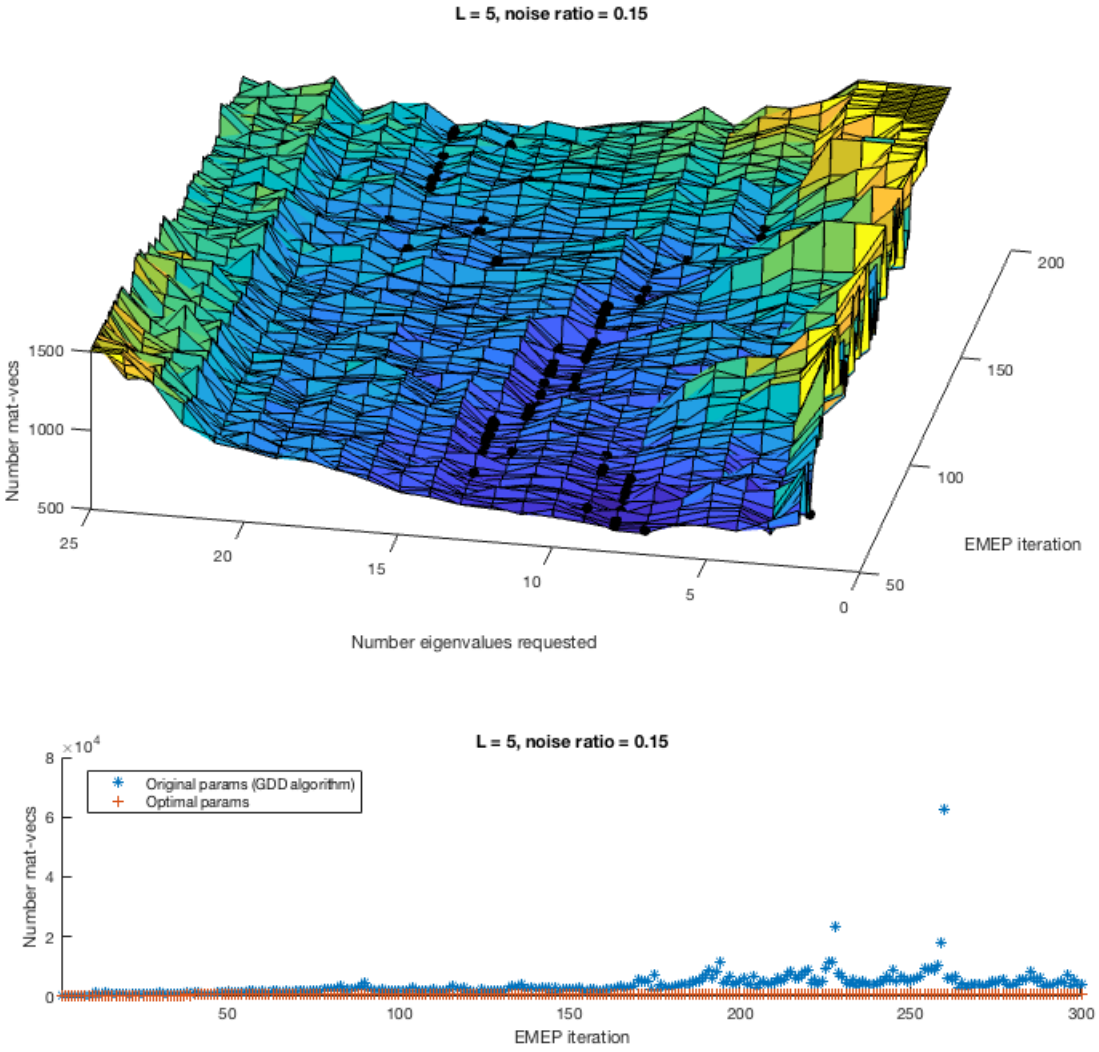

FIGURE 6.5. Performance results for an EMEP from a PLGD model with Gaussian noise (5.2.3) with noise ratio $\epsilon_{\text{rel}} = 0.15$, oversampling rate $L = 5$, and original signal from Figure 1.2 resized to 64×64 pixels. Top: Number of matrix-vector products (capped at 1,500 for better viewing) for various EMEP iterates and number of requested eigenvalues r. Arnoldi decomposition size is set to $m = 40$ and black dots indicate the empirically optimal values $\bar{r}$. Bottom: Plot of IRAM results for the EMEP with empirically optimal values $\bar{r}$ from top plot and fixed parameters $r = 2$ and $m = 20$.

We now examine Figure 6.5 to develop an adaptive strategy for choosing the number of requested eigenvalues r for the sequence of EMEP iterates. The top plot in Figure 6.5 shows that the empirically optimal value $\bar{r}$ typically changes only slightly between EMEP iterates. However, the empirically optimal $\bar{r}$ can increase quickly for later EMEP iterates (as we see around iterate 150 in the top plot in Figure 6.5). Based on these observations, we may develop a strategy for choosing a sequence of parameters $r_0, r_1, \ldots, r_{maxit}$ as follows. In order to measure change in the number of matrix-vector products for each iterate, we require that each pair of parameters r_{k-1} and r_k differ by at least one. To select a new parameter r, we compare the two most recent choices for r and the resulting number of matrix-vector products. If these two choices for r decreased the number of matrix-vector products, we continue to shift the value of r in this direction by one unit; and otherwise we shift r in the opposite direction. We also allow r to increase or decrease r more rapidly by comparing the four most recent choices for r and the resulting number of matrix-vector products using linear interpolation. If these recent choices suggest the same shift as the first comparison, then we shift r in this direction by two units rather than one.

Formally, this adaptive strategy for choosing the number of requested eigenvalues r_k has the following algorithmic structure. To initialize the algorithm, we select a fixed Arnoldi decomposition size m and minimum and maximum values for r (with default values $m = 40$, $r_{min} = 2$, and $r_{max} = \min\{30, m-5\}$). For each EMEP iterate k, we update $r_k = r_{k-1}+\delta$, where $\delta \in \{-2, -1, 1, 2\}$ is a shift based on the number of requested eigenvalues $r_{k-1}, r_{k-2}, \ldots$, and number of matrix-vector products $t_{k-1}, t_{k-2}, \ldots$, for the previous EMEP iterates.

For the first iterate $k = 0$, we set $r_0 = r_{min}$. For future iterates, if $r_{k-1} = r_{min}$ or $r_{k-1} = r_{max}$, we increase or decrease r_k by 1, respectively. Thus, for $k = 1$ we have $r_1 = r_{min} + 1$.

If $k < 4$ and r_{k-1} is not r_{min} or r_{max}, then we determine a *2-step shift value* $\delta_2 \in \{-1, 1\}$ based on r_{k-1}, r_{k-2}, t_{k-1}, and t_{k-2}. If $r_{k-1} > r_{k-2}$ and $t_{k-1} < t_{k-2}$ then the number of matrix-vector products in the EMEP (6.2.2) decreased as the number of requested eigenvalues was increased, suggesting we should increase r_{k-1} by $\delta_2 = 1$. By the same reasoning for the other three inequality cases, we define the *2-step shift value* as

$$\delta_2 = \text{sign}(r_{k-1} - r_{k-2}) \cdot \text{sign}(t_{k-2} - t_{k-1}), \tag{6.4.1}$$

where sign(0) is defined as 1. We then set $r_k = r_{k-1} + \delta_2$.

If $k \geq 4$ and r_{k-1} is not r_{min} or r_{max}, then we compute the 2-step shift value δ_2 from (6.4.1) as well as a *4-step shift value* based on the past four requested eigenvalue numbers and matrix-vector products. To compute the 4-step shift value, we solve the linear interpolation problem

$$\min_{\alpha,\beta} ||\mathbf{t} - \alpha\mathbf{e} - \beta\mathbf{r}||_2, \tag{6.4.2}$$

where $\mathbf{t} = [t_{k-4}, t_{k-3}, t_{k-2}, t_{k-1}]^T$ is the vector of matrix-vector product values, $\mathbf{e} = [1, 1, 1, 1]^T$, and $\mathbf{r} = [r_{k-4}, r_{k-3}, r_{k-2}, r_{k-1}]^T$ is the vector of the number of requested eigenvalues. If the solution to (6.4.2) has $\beta > 0$ then the past four eigenvalue problems suggest that t increases with r, and thus we should decrease r_{k-1}. This reasoning gives the *4-step shift value*

$$\delta_4 = -\text{sign}(\beta), \tag{6.4.3}$$

where β is determined by (6.4.2). If $\delta_2 = \delta_4$, then the 2-step (6.4.1) and 4-step (6.4.3) equations both suggest we should shift in the direction of δ_2, and we select the shift $\delta = 2\delta_2$. If $\delta_2 \neq \delta_4$ then we rely on the 2-step equation (6.4.1) and select the shift value $\delta = \delta_2$. We then set $r_k = \min\{\max\{r_{k-1} + \delta, r_{min}\}, r_{max}\}$ to satisfy the condition $r_{min} \leq r_k \leq r_{max}$.

Given the value for r_k determined above, we compute the two algebraically largest eigenpairs (λ_1, v_1) and (λ_2, v_2) of the matrix A_k using the IRAM (Algorithm 7) with the number of requested eigenvalues r_k. We also collect the number of matrix-vector products t_k used by the IRAM, and return the data: (λ_1, v_1), (λ_2, v_2), t_k, r_k. Altogether, these steps lead to *the IRAM with adaptive parameter selection* for the EMEP (6.2.2).

Algorithm 8 The IRAM with adaptive parameter selection for the EMEP (6.2.2)

Input: Matrix iterate A_k from the EMEP (6.2.2), previous number of matrix-vector products $T = \{t_0, t_1, \ldots, t_{k-1}\}$ required by the IRAM, previous number of requested eigenvalues $R = \{r_0, r_1, \ldots r_{k-1}\}$, r_{min} (default $r_{min} = 2$), r_{max} (default $r_{max} = \min\{30, m-5\}$), and Arnoldi decomposition (6.3.4) size m (default $m = 40$).

Output: Eigenpairs (λ_1, v_1) and (λ_2, v_2), number of matrix-vector products t_k, number of requested eigenvalues r_k.

if $k = 0$ **then**
    $r_k = r_{min}$
else if $r_{k-1} = r_{min}$ **then**
    $r_k = r_{k-1} + 1$
else if $r_{k-1} = r_{max}$ **then**
    $r_k = r_{k-1} - 1$
else if $k < 4$ **then**
    Compute 2-step shift value δ_2 from (6.4.1)
    $r_k = r_{k-1} + \delta_2$
else
    Compute 2-step shift value δ_2 from (6.4.1) and 4-step shift value δ_4 from (6.4.3)
    if $\delta_2 = \delta_4$ **then**
        Set $\delta = 2\delta_2$
    else
        Set $\delta = \delta_2$
    end if
    $r_k = \min\{\max\{r_{k-1} + \delta, r_{min}\}, r_{max}\}$
end if
Algorithm 7: Perform IRAM with matrix A_k, number of requested algebraically largest eigenvalues r_k, and maximum Arnoldi decomposition size m. Return eigenpairs (λ_1, v_1), (λ_2, v_2) and number of matrix-vector products t_k.
Return: (λ_1, v_1), (λ_2, v_2), t_k, r_k.

Note that the only parameters in Algorithm 8 are r_{min}, r_{max}, and the Arnoldi decomposition (6.3.4) size m, which determines the size of the basis $Q_m \in \mathbb{C}^{n\times m}$ in the Arnoldi decomposition $AQ_m = Q_m H_m + r_m e_m^*$. The choice of m is a trade-off between computational efficiency and data storage constraints. We seek the smallest value m possible, since Q_m must be stored in random-access memory and each column of Q_m is the size of the desired signal $\mathbf{x}$ in the phase retrieval problem (1.0.1). However, as we will see in Section 6.5, m must be sufficiently large for the shifted QR iteration (Algorithm 6) in the IRAM to handle the EMEP (6.2.2) efficiently. For now we will let $m = 40$ to demonstrate the behavior of Algorithm 8.

We close this section by showing that Algorithm 8 selects a sequence $r_0, r_1, \ldots r_{maxit}$ which varies significantly and generally tracks the empirically optimal value $\bar{r}$ for each EMEP (6.2.2) iterate.

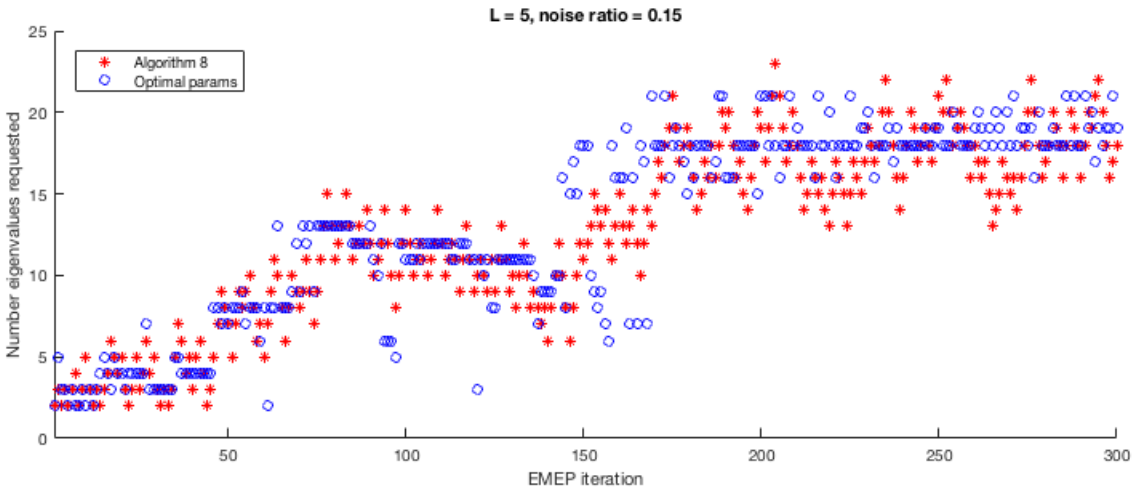

FIGURE 6.6. Plot comparing Algorithm 8 with the empirically optimal value $\bar{r}$. The EMEP (6.2.2) is from Figure 6.5 and Arnoldi decomposition (6.3.4) size is $m = 40$.

Figure 6.6 shows that the parameter values r_k chosen by Algorithm 8 effectively track the empirically optimal value $\bar{r}$. For the majority of iterates, the value r_k is within two units from the empirically optimal $\bar{r}$. As we will see in Section 6.5, these changes in r_k are related to the evolving spectrum of the EMEP (6.2.2).

6.5. Spectrum Distribution and IRAM Behavior

In this section we examine an observed correlation between the clustering of the algebraically largest eigenvalues in the EMEP (6.2.2) and the convergence behavior of the IRAM (Algorithm 7) for various parameters. As we saw in Section 5.3, PLGD models with Gaussian noise (5.2.3) typically have optimal EMEP matrices $\mathcal{A}^* y_\star$ for which the algebraically largest eigenvalue has multiplicity greater than one. Thus we may expect the algebraically largest eigenvalues in the EMEP to cluster for later iterates.

In Section 6.5.1 we examine two PLGD models and show that as the eigenvalues in the EMEP begin to cluster, the empirically optimal value $\bar{r}$ increases such that $\lambda_{\bar{r}+1}$ is not clustered with $\lambda_1, \lambda_2, \ldots, \lambda_{\bar{r}}$. We also see that the value r_k chosen by Algorithm 8 properly tracks the empirically optimal value $\bar{r}_k$ for these two PLGD models. Next, Section 6.5.2 uses these two PLGD models to demonstrate that the Arnoldi decomposition (6.3.4) size $m = 40$ is sufficiently large for the IRAM to perform efficiently, and thus $m = 40$ is an appropriate default parameter for Algorithm 8.

To illustrate these observations, we focus on the EMEP of two particular PLGD models with Gaussian noise (5.2.3) for which the sequence of parameters $r_0, r_1, \ldots r_{maxit}$ chosen by Algorithm 8 varies greatly.

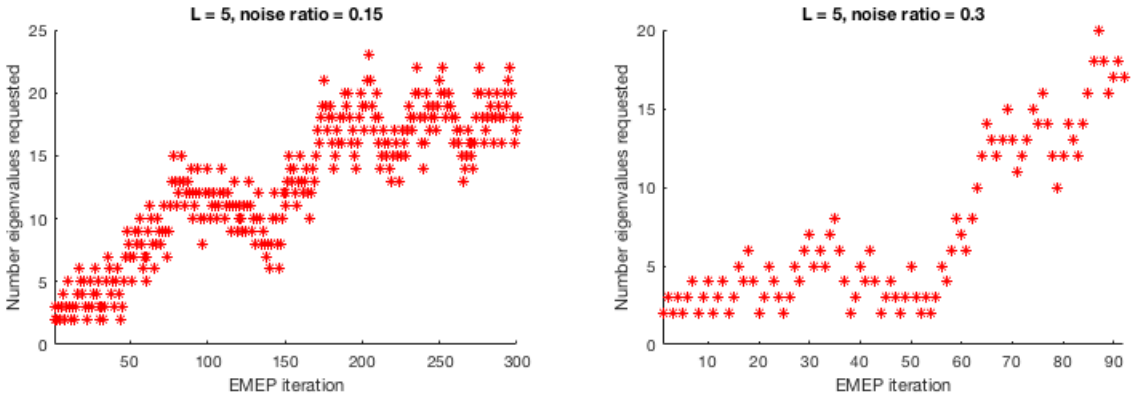

FIGURE 6.7. Number of requested eigenvalues r chosen by Algorithm 8 for the EMEP (6.2.2) from two PLGD models with Gaussian noise (5.2.3), with noise ratio $\epsilon_{\text{rel}} = 0.15, 0.30$, oversampling rate $L = 5$, and original signal from Figure 1.2 resized to 64×64 pixels.

For both PLGD models in Figure 6.7, changes in the EMEP caused Algorithm 8 to increase the number of requested eigenvalues r_k significantly from early to later iterates k. Yet the value

of r_k increased at a quicker rate for the experiment in Figure 6.7 with $L = 5$ and $\epsilon_{\text{rel}} = 0.30$. As we will see in Section 6.5.1, this quick increase from $r_{55} = 3$ to $r_{70} = 13$ coincides with a sudden clustering of the algebraically largest eigenvalues in the EMEP.

6.5.1. Eigenvalue Clustering and IRAM Parameter Choice

We now show that the PLGD models in Figure 6.7 exhibit a correlation between the spectrum distribution of the EMEP (6.2.2) and the convergence behavior of the IRAM (Algorithm 7). For these two models, as the algebraically largest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{m-1}, \lambda_m$ of the EMEP begin to cluster, the IRAM converges much faster if the number of requested eigenvalues $r < m$ is large enough such that λ_{r+1} is not clustered with $\lambda_1, \lambda_2, \ldots, \lambda_r$. Thus, as Algorithm 8 explicitly tracks the empirically optimal sequence of parameters $\bar{r}_k$, this algorithm implicitly tracks the clustering of the algebraically largest eigenvalues in the EMEP. To demonstrate these observations, we consider various EMEP iterates k and number of requested eigenvalues r, and plot the numbers of matrix-vector products for the IRAM as well as the eigenvalue difference $\lambda_{r+1} - \lambda_r$. Figures 6.8 and 6.9 depict about half of the EMEP iterates from the two PLGD models in Figure 6.7, so we may focus on the regions where r_k in Algorithm 8 varies greatest.

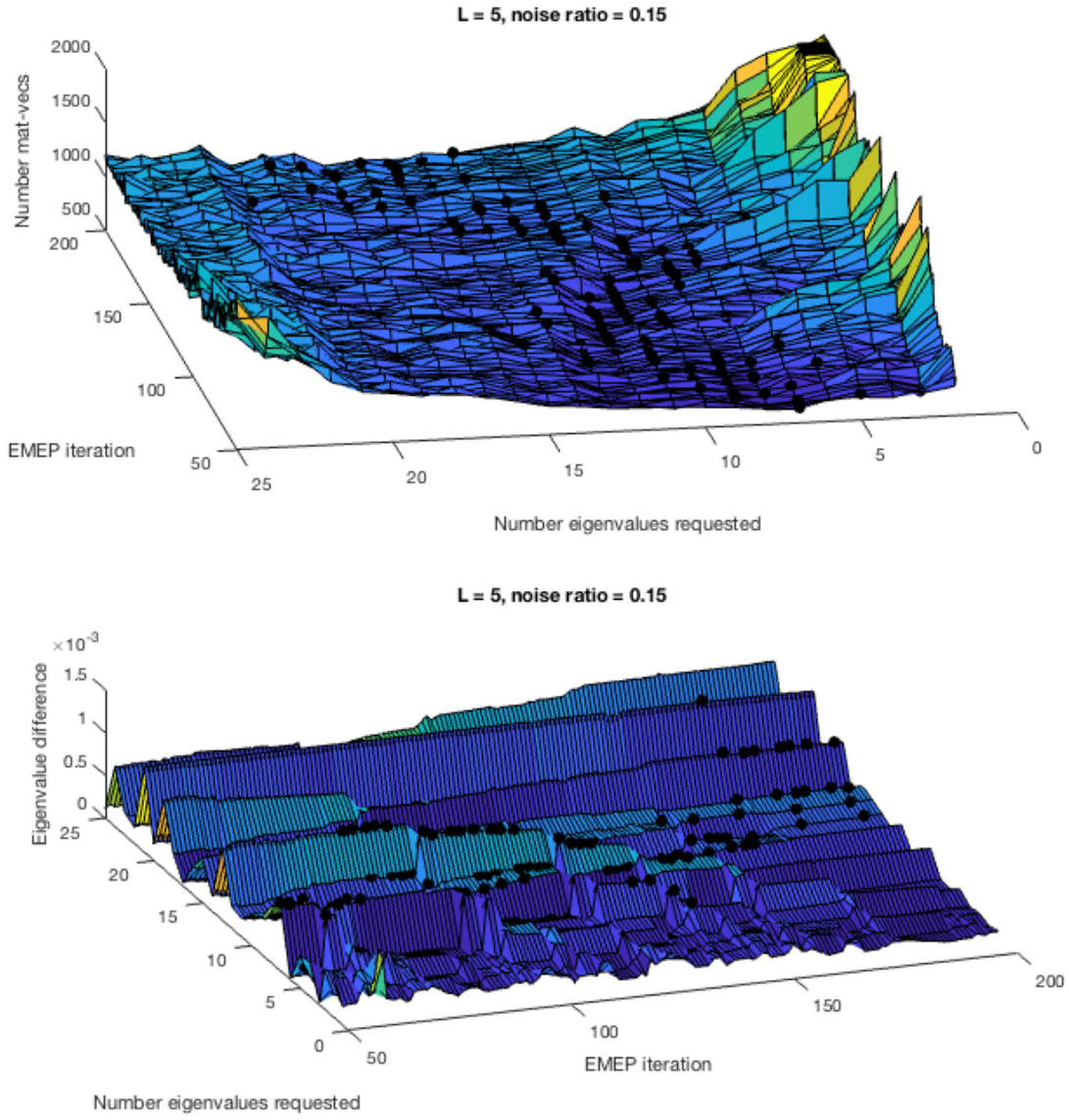

FIGURE 6.8. Behavior of Algorithm 8 for the experiment from Figure 6.7 with $L = 5$, $\epsilon_{\text{rel}} = 0.15$. Top: Number of matrix-vector products (capped at 2,000 for better viewing) for each EMEP (6.2.2) iterate k and number of requested eigenvalues r. Bottom: eigenvalue differences $\lambda_r - \lambda_{r+1}$ for each EMEP (6.2.2) iterate k and number of requested eigenvalues r. Black dots in both plots indicate the value r_k chosen by Algorithm 8.

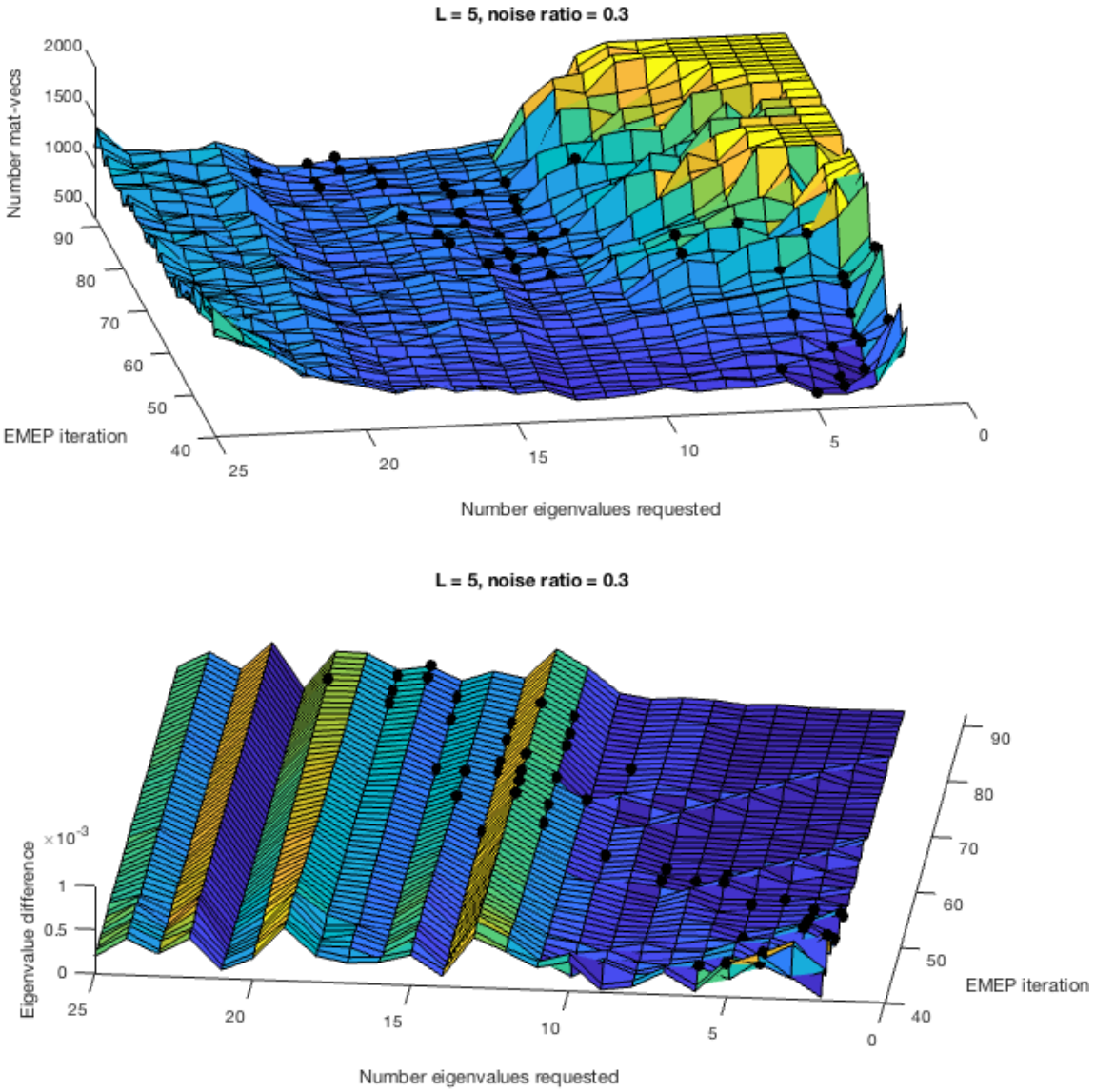

FIGURE 6.9. Behavior of Algorithm 8 for the experiment from Figure 6.7 with $L = 5$, $\epsilon_{\text{rel}} = 0.30$. Top: Number of matrix-vector products (capped at 2,000 for better viewing) for each EMEP (6.2.2) iterate k and number of requested eigenvalues r. Bottom: eigenvalue differences $\lambda_r - \lambda_{r+1}$ for each EMEP (6.2.2) iterate k and number of requested eigenvalues r. Black dots in both plots indicate the value r_k chosen by Algorithm 8.

In Figures 6.8 and 6.9 we see that the clustering of the algebraically largest eigenvalues corresponds to an increase in the empirically optimal value $\bar{r}_k$ for later EMEP (6.2.2) iterates k. In Figure 6.8, the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{10}$ cluster around iterates $125 \leq k \leq 175$; and likewise the empirically optimal value $\bar{r}_{125} \approx 10$ increases to $\bar{r}_{175} \approx 18$. Figure 6.9 depicts a similar shift, where the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{10}$ cluster around iterates $50 \leq k \leq 70$, and the empirically optimal value increases to $\bar{r}_{70} \approx 15$. In the case of Figure 6.9, the most tightly clustered eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_8$ also correspond to parameter values $r = 2, 3, \ldots, 7$ for which the IRAM required the greatest number of matrix-vector products to converge.

This observed correlation between eigenvalue clustering and a change in the convergence rate of the IRAM may be related to the subroutines used in the IRAM. As discussed in Section 6.3, the IRAM is based on the following two algorithms. Given a Hermitian matrix $A \in \mathbb{C}^{n \times n}$, the m-step Arnoldi iteration (Algorithm 5) generates a set of Ritz pairs $(\theta_1, u_1), (\theta_2, u_2), \ldots, (\theta_m, u_m)$ for A with respect to $\mathcal{K}_m(A, q_1)$. Next, the p-step shifted QR iteration (Algorithm 6) restarts the Arnoldi decomposition (6.3.4) by attempting to damp the unwanted part of the spectrum using the Ritz values $\{\theta_{r+1}, \theta_{r+2}, \ldots, \theta_m\}$ (where $m = r + p$). Assume we have an EMEP matrix iterate A_k with some number s of clustered algebraically largest eigenvalues, $\lambda_1 \approx \lambda_2 \approx \cdots \approx \lambda_s$; and assume we select the number of requested eigenvalues $r < s$. When the IRAM builds an Arnoldi decomposition (6.3.4), the s largest Ritz values of A_k with respect to $\mathcal{K}_m(A_k, q_1)$ may include values $\theta_{r+1}, \theta_{r+2}, \ldots, \theta_s$ which are close approximations to the desired eigenvalues. Thus when the shift values $\mu_1 = \theta_{r+1}, \mu_2 = \theta_{r+2}, \ldots, \mu_{s-r} = \theta_s, \ldots, \mu_p = \theta_m$ are passed to the p-step shifted QR iteration, the implicit polynomial filter (6.3.12) will include values $\mu_1, \mu_2, \ldots, \mu_{s-r}$ which damp the desired part of the spectrum. Thus, if $r < s$ we may expect the IRAM to converge more slowly. However, if we select the number of requested eigenvalues $r > s$, then the shift values used in the p-step shifted QR iteration will always include parts of the spectrum which we want to damp and we may expect the IRAM to converge more quickly.

Figures 6.8 and 6.9 also demonstrate that Algorithm 8 properly tracks the empirically optimal value $\bar{r}_k$ as this value changes. In particular, Figure 6.9 indicates that $\bar{r}_k$ increased quickly from iterates $k = 50$ to $k = 70$. Likewise, Algorithm 8 increased r_k by two units for several iterates $55 \leq k \leq 70$ (see Figure 6.7). These two-unit increases were the result of Algorithm 8 properly

identifying a quick increase in the number of matrix-vector products t_k for smaller values r_k. Recall that Algorithm 8 determines the update r_k using two-point and four-point linear interpolations (equations (6.4.1) and (6.4.2), respectively) of the previous parameters $r_{k-1}, r_{k_2}, r_{k-3}, r_{k-4}$ and $t_{k-1}, t_{k_2}, t_{k-3}, t_{k-4}$. For the PLGD model in Figure 6.9, both of these interpolations agreed (i.e., $\delta_2 = 1$ in (6.4.1) was equal to $\delta_4 = 1$ in (6.4.3)) for several iterates $55 \leq k \leq 70$. Thus Algorithm 8 quickly increased r_k, avoiding excessive matrix-vector multiplications.

6.5.2. Selecting the Arnoldi Decomposition Size

Next we show that the default parameter setting $m = 40$ in Algorithm 8 is sufficiently large to minimize the number of matrix-vector products in the EMEP in the models from Figure 6.7. To demonstrate that $m = 40$ is appropriate, we will examine a more difficult eigenvalue problem (i.e., later iterate) from each of these models. Figure 6.10 depicts the number of matrix-vector products for various IRAM parameters r and m for an iterate from each model.

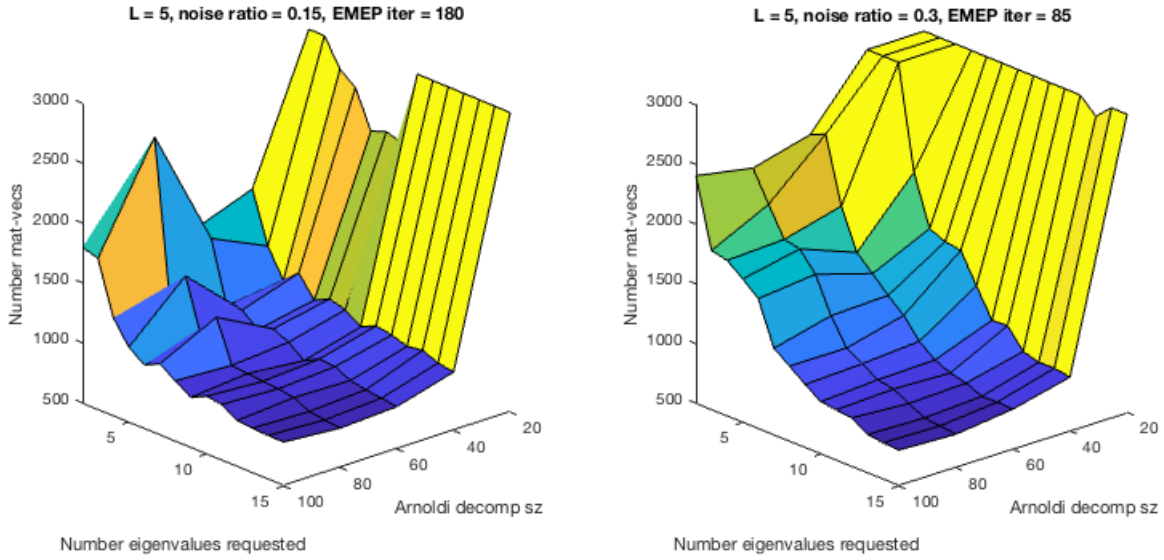

FIGURE 6.10. Number of matrix-vector products (capped at 3,000 for better viewing) for individual EMEP (6.2.2) iterates with varying number of requested eigenvalues r and Arnoldi decomposition (6.3.4) size m. Left: EMEP iterate 180 from Figure 6.8. Right: EMEP iterate 85 from Figure 6.9.

Figure 6.10 suggests that we should not select IRAM parameters below $r = 9$ and $m = 40$ for the EMEP iterate in the left plot, nor should we select parameters below $r = 12$ and $m = 40$ for the EMEP iterate in the right plot. Yet there is no significant benefit to selecting larger parameter

values. Since the EMEP iterates in Figure 6.10 represent later, more difficult eigenvalue problems, this figure suggests that $m = 40$ is sufficiently large for all iterates. Thus we select a fixed Arnoldi decomposition (6.3.4) size $m = 40$ for Algorithm 8.

CHAPTER 7

The Improved Gauge Dual Descent Algorithm and Performance Results

7.1. Introduction

In this chapter we present the Improved Gauge Dual Descent (IGDD) algorithm and examine the performance of the IGDD algorithm. Section 7.2 states the IGDD algorithm, which applies the new termination conditions of Chapter 5 and *the IRAM with adaptive parameter selection* of Chapter 6 to the GDD algorithm (Algorithm 3). Next, Section 7.3 examines the performance of the IGDD algorithm. We demonstrate that the IGDD algorithm is more efficient than the GDD algorithm for a variety of PLGD models with Gaussian noise (5.2.3). We observe that the IGDD algorithm often reduces the number of matrix-vector products in the EMEP (6.2.2) by at least 50% as compared with the GDD algorithm. Further experiments show that this reduction in the number of matrix-vector products is comparable to the reduction observed by solving the EMEP (6.2.2) with the empirically optimal (6.4.1) number of requested eigenvalues for each EMEP iterate. Finally, we demonstrate that the Arnoldi decomposition (6.3.4) size $m = 40$ for the IGDD algorithm strikes a proper balance between increasing computational efficiency and minimizing data storage. Note that all experiments in this chapter involve noisy phase retrieval problems and thus we also use the new termination conditions for the GDD algorithm (otherwise the GDD algorithm would not terminate, as discussed in Section 5.3). All experiments in this chapter are available for reproduction.[1]

7.2. The Improved Gauge Dual Descent Algorithm

In this section we briefly summarize the contributions of Chapters 5 and 6 which lead to the Improved Gauge Dual Descent (IGDD) algorithm.

[1] `https://github.com/Will-Wright/low-rank-opt-rapid-eig`

Section 5.3 demonstrated that the GDD algorithm fails to converge for PLGD models with Gaussian noise (5.2.3). Section 5.4 then established new termination conditions: the primal difference condition (5.4.1) and the dual variable difference condition (5.4.2). Section 5.4 also demonstrated that these new termination conditions accurately identify stagnation of signal recovery progress in the GDD algorithm. For convenience, we restate these two new termination conditions: the primal difference condition

$$\frac{|\rho - \hat{\rho}|}{\rho} \leq \text{tol}_{\text{primal}} = 10^{-5} \tag{7.2.1}$$

where $\rho = ||\mathcal{A}(xx^*) - b||_2$, and the dual variable difference condition

$$\frac{||y - \hat{y}||_2}{||y||_2} \leq \text{tol}_{\text{dual}} = 10^{-4}, \tag{7.2.2}$$

where a hat indicates the previous iterate (e.g., $\hat{y} = y_{k-1}$).

Section 6.4 showed that the computational costs of the GDD algorithm are related to the fixed choice of parameters used in the IRAM (Algorithm 7), the eigenvalue method used by the GDD algorithm. The GDD algorithm uses the IRAM with a fixed the number of requested eigenvalues $r = 2$ and Arnoldi decomposition (6.3.4) size $m = 20$. Section 6.4 examined the empirical performance of the IRAM for a range of parameters r to develop *the IRAM with adaptive parameter selection* (Algorithm 8). Algorithm 8 increases m to the default value 40 and adaptively changes r in order to decrease the number of matrix-vector products required by the IRAM.

Applying these the new termination conditions and Algorithm 8 to the GDD algorithm, we see the following steps change in the GDD algorithm (Algorithm 3). The input now includes additional parameters (r_{min}, r_{max}, m, $\text{tol}_{\text{primal}}$, and tol_{dual}), and the initialization includes the sets $T = \{\emptyset\}$ and $R = \{\emptyset\}$ to track the parameters t_k and r_k for Algorithm 8. Step 1 of Algorithm 3 is modified to include the new termination conditions (7.2.1) and (7.2.2). Step 2 of Algorithm 3 is modified to perform the eigenvalue computation by passing the necessary parameters to Algorithm 8. Finally, step 16 updates the termination parameter $\rho = ||\mathcal{A}(xx^*) - b||_2$ and the sets T and R. Altogether, these changes result in the Improved Gauge Dual Descent (IGDD) algorithm for optimizing the PLGD model (5.2.3).

Algorithm 9 Improved Gauge Dual Descent (IGDD) algorithm

Input: Sensing operator $\mathcal{A}$ and adjoint $\mathcal{A}^*$, observation vector b, initial dual variable y_0, estimate of total noise level ϵ, minimum and maximum number of requested eigenvalues, r_{min} and r_{max}, Arnoldi decomposition size m, and convergence tolerances (7.2.1) and (7.2.2).

Output: Approximate solution signal x.

Initialization: Set $T = \{\emptyset\}$, $R = \{\emptyset\}$, $k = 0$.

while conditions (7.2.1) and (7.2.2) are not satisfied **do**

  Compute algebraically largest eigenvalues and corresponding eigenvectors: Perform Algorithm 8 with inputs $A_k = \mathcal{A}^* y_k$, T, R, r_{min}, r_{max}, and m to obtain (λ_1, v_1), (λ_2, v_2), t_k, r_k.

  Compute (sub)gradient: $g = \mathcal{A}(v_1 v_1^*)$ based on (4.2.1).

  Determine differentiability of $\lambda_1(\mathcal{A}^* y_k)$ based on (4.2.3).

  if $\lambda_1(\mathcal{A}^* y_k)$ is differentiable **then**

    Linesearch: Perform linesearch (4.2.5) with initial step α (4.2.4) to obtain y_{k+1}.

  else

    Projected subgradient step: Set $y_{k+1} = \Pi_{\mathcal{C}}(y_k - \alpha g)$ with α from (4.2.6).

  end if

  Primal recovery: Compute $\hat{x}$ based on (4.2.10).

  Primal refinement: Find x_{k+1} as the solution to (4.2.11) initialized with $\hat{x}$ and y_{k+1}.

  if $\epsilon = 0$ **then**

    Dual refinement: Find $\widehat{y}$ based on (4.2.12).

    if $\lambda_1(\mathcal{A}^* \widehat{y}) < \lambda_1$, set $y_{k+1} = \widehat{y}$.

  end if

  Update: Set $\rho_{k+1} = ||\mathcal{A}(xx^*) - b||_2$. Append t_k to T and r_k to R. Set $k = k + 1$.

end while

Return: $x = x_k$.

For our implementation of the IGDD algorithm, we use the default values $r_{min} = 2$, $r_{max} = \min\{30, m - 5\}$, and $m = 40$ (as discussed in Section 6.4).

7.3. Performance Results

We now examine the performance behavior of the IGDD algorithm (Algorithm 9) for a variety of noisy phase retrieval problems. Section 7.3.1 demonstrates that the IGDD algorithm requires $50 - 80\%$ fewer matrix-vector multiplications than the GDD algorithm for a variety of problems. Furthermore, Section 7.3.2 shows that the performance of the IGDD algorithm is comparable to solving the EMEP (6.2.2) with the empirically optimal (6.4.1) value $\bar{r}_k$ for each iterate k. We see that the number of requested eigenvalues r_k chosen by the IGDD algorithm effectively tracks the empirically optimal number of requested eigenvalues $\bar{r}_k$, causing the IGDD algorithm to minimize the number of matrix-vector products for a fixed Arnoldi decomposition size m. Finally, Section 7.3.3 demonstrates that the default Arnoldi decomposition size $m = 40$ for the IGDD algorithm strikes a reasonable balance between improving computational performance and minimizing memory usage.

7.3.1. Random Gaussian Signals and Large-Scale Images

We begin by demonstrating that the IGDD algorithm requires $50 - 80\%$ fewer matrix-vector multiplications than the GDD algorithm for random signals and large-scale image phase retrieval problems. Figure 7.1 depicts a set of experiments with randomly generated Gaussian signals.

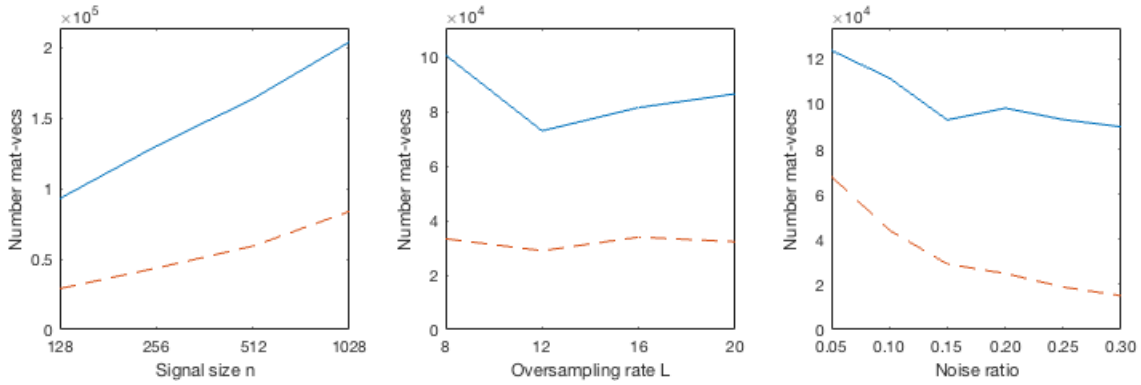

FIGURE 7.1. Performance results for PLGD models with Gaussian noise (5.2.3), where signals are complex with standard Gaussian distribution (1.0.19). Results indicate the GDD algorithm (solid line) and IGDD algorithm (dashed line). Each result is the mean of 10 experiments. Left: Varying signal size n, with fixed noise ratio $\epsilon_{\text{rel}} = 0.15$ and oversampling scaled logarithmically with n (i.e., $L = 10, 12, 12, 14$) as indicated in Theorem 2.4.2. Middle: Varying oversampling rate L, with fixed signal size $n = 128$ and noise ratio $\epsilon_{\text{rel}} = 0.15$. Right: Varying noise ratio ϵ_{rel}, with fixed signal size $n = 128$ and oversampling rate $L = 10$.

In Figure 7.1 we see that the IGDD algorithm requires fewer matrix-vector products than the GDD algorithm for a wide range of phase retrieval problems varying in size, oversampling rate, and noise ratio. The left and middle plots in Figure 7.1 suggest that the IGDD algorithm requires about 60% fewer matrix-vector products regardless of signal size n or oversampling rate L. Additionally, the right plot in Figure 7.1 suggests the IGDD algorithm may reduce matrix-vector products by 80% or more for problems with greater noise. This difference in performance may be related to the evolving spectrum of the EMEP (6.2.2) for noisier phase retrieval problems. As we saw in Section 6.5.1, problems with a higher noise ratio (e.g., Figure 6.9) may see greater clustering of the algebraically largest eigenvalues during early iterations. Recall that the IRAM (Algorithm 7) tends to perform poorly when the number of requested eigenvalues r is such that λ_r is clustered with its algebraically neighboring eigenvalues (i.e., $\cdots \approx \lambda_{r+2} \approx \lambda_{r+1} \approx \lambda_r \approx \lambda_{r-1} \approx \lambda_{r-2} \approx \cdots$). Since the GDD algorithm calls the IRAM (Algorithm 7) with a fixed $r = 2$ requested eigenvalues, the GDD algorithm may require far more matrix-vector products than the IGDD for noisier phase retrieval problems.

We also find that the IGDD algorithm is more efficient than the GDD algorithm for image-based phase retrieval problems. Table 7.1 shows the performance results for two larger images from Figure 7.2.

Image	n	L	GDD	IGDD $m = 40$		IGDD $m = 80$	
Figure 7.2, left	$57,600$	10	562,255	297,767	47%	252,316	55%
		15	647,753	301,752	53%	253,209	61%
Figure 7.2, right	$120,000$	10	852,633	364,164	57%	287,309	66%
		15	563,085	291,630	48%	256,084	55%

TABLE 7.1. Performance results for PLGD models with Gaussian noise (5.2.3), where signals are the images in Figure 7.2. Results indicate total number of matrix-vector products and percent decrease from the GDD algorithm. Parameter m is the Arnoldi decomposition size (6.3.4) for the IRAM (Algorithm 7).

FIGURE 7.2. Images used for experiments in Table 7.1. Top left: original image of my daughter and me, image size $240 \times 240 = 57,600$ pixels. Bottom left: result image after solving EMEP. Top right: original image of UC Davis roundabout, image size $200 \times 600 = 120,000$ pixels. Bottom right: result image after solving EMEP. All experiments have noise ratio $\epsilon_{\text{rel}} = 0.15$ and oversampling $L = 15$.

Table 7.1 demonstrates that the benefits of the IGDD algorithm also apply to large-scale phase retrieval problems. Additionally, a larger Arnoldi decomposition size m further reduces the number of matrix-vector products. Thus, when solving large-scale problems it may be advisable to consider increasing this parameter beyond the default setting of $m = 40$ in the IGDD algorithm if memory constraints permit this increase.

7.3.2. IGDD vs Empirically Optimal Parameter Selection

Next, we demonstrate that the performance improvements of the IGDD algorithm are comparable to the improvements of solving the EMEP (6.2.2) with the empirically optimal (6.4.1) value $\bar{r}_k$ for each EMEP iterate k. Table 7.2 indicates the number of matrix-vector products for six PLGD models with Gaussian noise (5.2.3) using the GDD algorithm ($r = 2$), the IGDD algorithm (r chosen with Algorithm 8), and the empirically optimal sequence of values $\bar{r}_k$.

L	ϵ_{rel}	EMEP its	GDD	IGDD $m = 40$		Empirically optimal $\bar{r}$ $m = 40$	
5	0.05	300	406,308	198,070	51%	179,807	56%
5	0.15	300	1,099,045	258,385	76%	242,003	78%
5	0.30	92	444,697	69,510	84%	58,780	87%
10	0.05	153	80,453	68,709	15%	61,948	23%
10	0.15	108	88,317	57,231	35%	51,311	42%
10	0.30	54	72,486	25,809	64%	23,217	68%

TABLE 7.2. Performance results for PLGD models with Gaussian noise (5.2.3) with original signal from Figure 1.2 resized to 64×64 pixels. Results indicate total number of matrix-vector products and percent decrease from the GDD algorithm. Parameter m is the Arnoldi decomposition size (6.3.4).

For each of the experiments in Table 7.2, we see that the IGDD algorithm offers nearly the same performance improvement as that of solving the EMEP with the empirically optimal values $\bar{r}_k$. Notably, experiments in Table 7.2 with a low oversampling rate ($L = 5$) shows that the IGDD algorithm is particularly effective at decreasing the number of matrix-vector products when there is a large relative difference between the number of matrix-vector products for the GDD algorithm and the empirically optimal choice of values $\bar{r}_k$. To further explore this performance behavior, Figure 7.3 depicts the two PLGD models from Table 7.2 with the largest and smallest relative difference in matrix-vector products (those with $L = 5$, $\epsilon_{\text{rel}} = 0.15$, and $L = 10$, $\epsilon_{\text{rel}} = 0.05$, respectively).

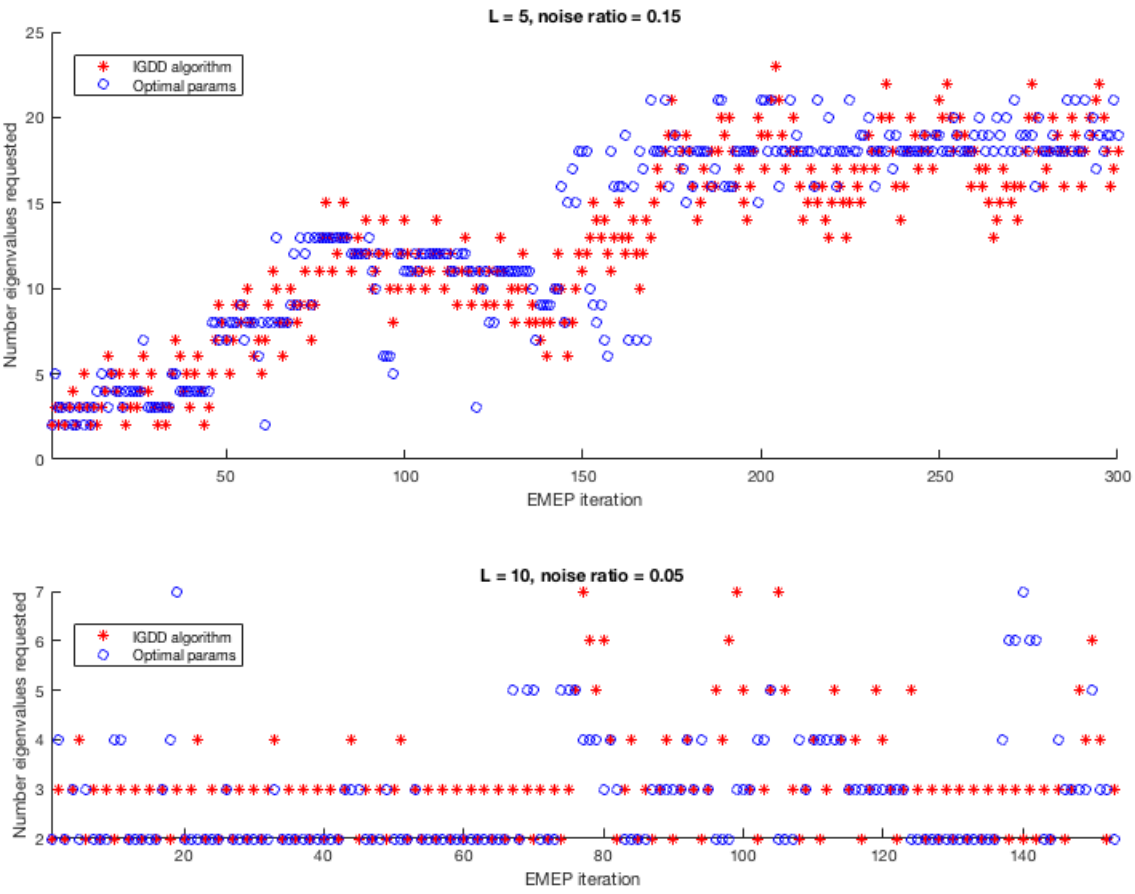

FIGURE 7.3. Number of requested eigenvalues r in the IRAM (Algorithm 7) for two PLGD models from Table 7.2.

In both PLGD models depicted in Figure 7.3, the number of requested eigenvalues r selected by the IGDD algorithm is usually within 1-3 units from the empirically optimal value $\bar{r}$. The bottom plot in Figure 7.3 suggests that the IGDD algorithm required relatively more matrix-vector products than the empirically optimal value $\bar{r}$ because the subroutine Algorithm 8 used in the IGDD algorithm always changes the value of r by one or two units, thus shifting away from the optimal value $\bar{r} = 2$ for many EMEP iterates. As we saw in Section 6.4 (e.g., Figure 6.9) the empirically optimal value $\bar{r}$ may shift rapidly for some PLGD models, and thus Algorithm 8 always changes r by one or two units to continue gathering performance information about the EMEP.

7.3.3. Selection of Arnoldi Decomposition Parameter

Finally, we demonstrate that the default Arnoldi decomposition (6.3.4) size $m = 40$ for the IGDD algorithm significantly improves computational performance while limiting memory usage. Table 7.3 depicts the total number of matrix-vector products required to solve the EMEP (6.2.2) for each PLGD model from Table 7.2 with the IGDD algorithm with various parameter values m.

n = 4,096			GDD	IGDD				
L	ϵ_{rel}	EMEP its		$m = 20$	$m = 40$	$m = 60$	$m = 80$	$m = 100$
5	0.05	300	406,308	358,195	198,070	189,401	192,042	201,270
5	0.15	300	1,099,045	806,412	258,385	224,048	214,118	215,392
5	0.30	92	444,697	175,669	69,510	56,193	55,146	54,987
10	0.05	153	80,453	77,768	68,709	64,300	68,602	73,754
10	0.15	108	88,317	65,833	57,231	53,261	54,388	55,308
10	0.30	54	72,486	28,799	25,809	24,699	25,113	25,491

TABLE 7.3. Total number of matrix-vector products for various PLGD models with Gaussian noise (5.2.3) with original signal from Figure 1.2 resized to 64×64 pixels. Parameter r is the number of requested eigenvalues in the IRAM (Algorithm 7) and m is the Arnoldi decomposition (6.3.4) size.

Table 7.3 demonstrates that the IGDD algorithm reduces the number of matrix-vector products from those of the GDD algorithm for all experiments considered. Yet this cost reduction varies significantly depending on the choice of Arnoldi decomposition (6.3.4) size m. We seek a default setting for the parameter m which is sufficiently large to yield the benefits of the IGDD algorithm, yet sufficiently small as not to impact memory constraints. To select an appropriate default value for m, we examine the two experiments from Table 7.3 with $\epsilon_{\text{rel}} = 0.15, 0.30$ and $L = 5$ which have the greatest original number of matrix-vector products, along with the greatest total decrease in cost when using the IGDD algorithm with a sufficiently large parameter m. Figure 7.4 singles out these two experiments, depicting the number of matrix-vector products for each EMEP (6.2.2) iteration.

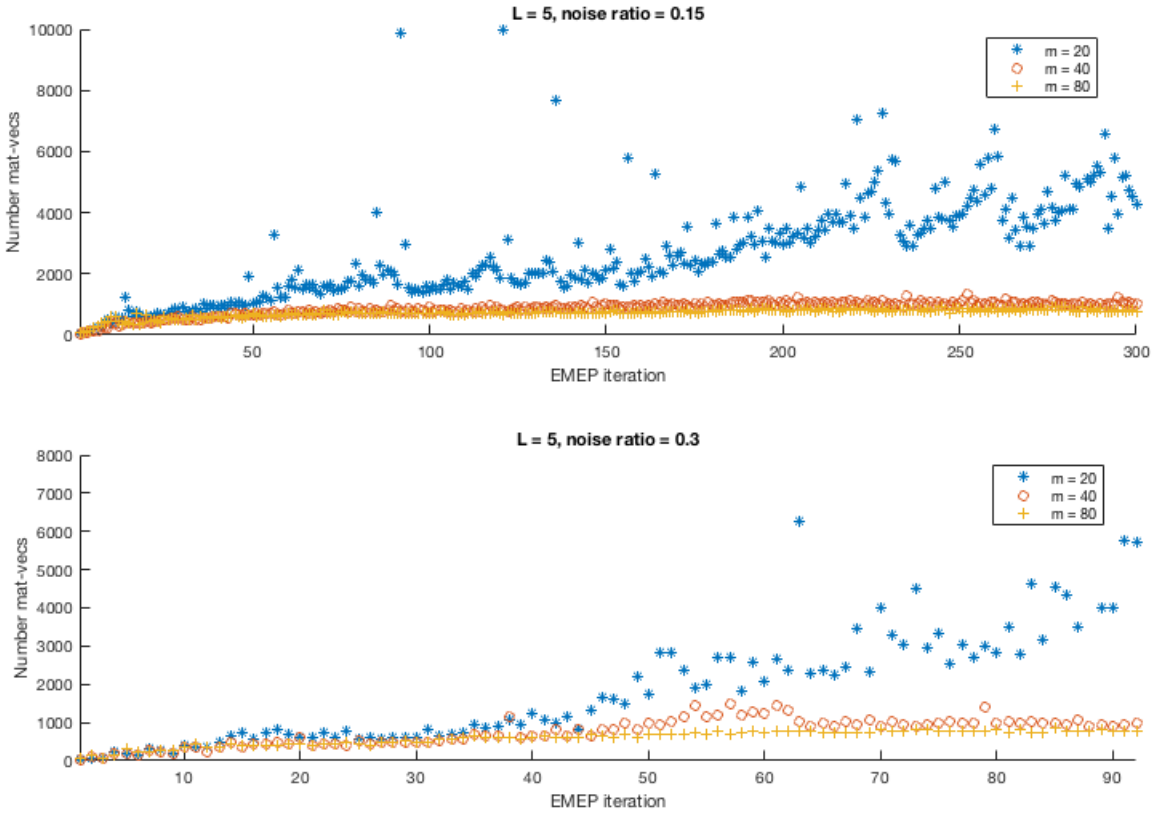

FIGURE 7.4. Number of matrix-vector products for each EMEP (6.2.2) iteration from two experiments in Figure 7.4 with various Arnoldi decomposition size $m = 20, 40, 80$.

Figure 7.4 demonstrates that the Arnoldi decomposition size of $m = 20$ is not sufficiently large to allow the IGDD algorithm to decrease the number of matrix-vector products. The dramatic matrix-vector product spikes for $m = 20$ in Figure 7.4 resemble those first seen in Figure 6.1 when using the GDD algorithm. Yet when the Arnoldi decomposition size is increased to $m = 40$, these cost spikes effectively disappear. The change in number of matrix-vector products between $m = 40$ and $m = 80$ is minimal for each EMEP iterate. Thus the default parameter of $m = 40$ for the IGDD algorithm strikes the proper balance between efficiency and memory constraints.

CHAPTER 8

Conclusion

In this dissertation we examined noisy phase retrieval, focusing on the PLGD model (3.2.1) which we chose to optimize with the Gauge Dual Descent (GDD) algorithm (Algorithm 3 from Chapter 4). Our work provides two main contributions which lead to the development of the Improved Gauge Dual Descent (IGDD) algorithm (Algorithm 9).

The first main contribution was to establish new termination conditions (5.4.1) and (5.4.2) for the GDD algorithm. Chapter 5 showed that PLGD models with Gaussian noise (5.2.3) cause the GDD algorithm to stagnate prior to satisfying the original termination conditions (4.2.13) and (4.2.14). We saw that this stagnation was likely the result of the optimal PLGD matrix $\mathcal{A}^* y_\star$ having an algebraically largest eigenvalue with multiplicity greater than one. Thus the objective function $\lambda_1(\mathcal{A}^* y)$ will be nondifferentiable in the neighborhood of $y_\star$ and we may expect first-order methods like the GDD algorithm to stagnate in this neighborhood. As a result, we established new termination conditions (5.4.1) and (5.4.2) based on empirical evidence of algorithmic stagnation.

The second main contribution was to develop a new strategy for handling the EMEP (6.2.2) in the GDD algorithm. In Chapter 6 we defined the EMEP in the GDD algorithm and examined the evolving spectrum of the EMEP. We observed that the algebraically largest eigenvalues of the EMEP tend to cluster for later iterates, making these eigenvalue problems more difficult for methods like the IRAM (Algorithm 7). We also showed that the EMEP is the computational bottleneck of the GDD algorithm, requiring about 95% of the matrix-vector products in the GDD algorithm. Section 6.3 then reviewed the IRAM and its component algorithms, providing insight for how we may better select the IRAM parameters. In Section 6.4 we developed Algorithm 8, a new strategy for solving the EMEP by selecting the number of requested eigenvalues in the IRAM based on empirical observations of IRAM behavior. Section 6.5 showed that changes in the number of requested eigenvalues, as chosen by Algorithm 8, correspond to clustering of the algebraically largest eigenvalues in the EMEP.

In Chapter 7 we presented the IGDD algorithm which incorporates the new termination conditions of Chapter 5 and the new strategy for the EMEP from Chapter 6. Performance results demonstrated that the IGDD algorithm is more efficient than the GDD algorithm for a variety of noisy phase retrieval problems. In particular, the IGDD algorithm typically reduced the number of matrix-vector products by 50-80% for problems with a low oversampling rate.

In addition to these main contributions, Chapter 3 presented a self-contained, comprehensive treatment of gauge duality theory for establishing and analyzing the PLP-PLGD pair. Also, Appendix A examined some of the benefits of choosing the PLGD model (3.2.1) for noisy phase retrieval. We saw that the PLGD model is well-suited for developing a first-order method, and that the GDD algorithm typically solves noisy phase retrieval problems with greater accuracy than the wflow algorithm (Algorithm 2).

Several theoretical and computational questions still remain for future work. In terms of theory, it would be beneficial to determine the expected rank of the optimal PLGD matrix $\mathcal{A}^* y_\star$ for a given noisy phase retrieval problem. As shown in Chapter 5, PLGD models with Gaussian noise typically have optimal matrices $\mathcal{A}^* y_\star$ with rank ρ greater than one (see Table 5.3). Thus, as the GDD algorithm progresses we may expect the ρ algebraically largest eigenvalues of $\mathcal{A}^* y$ to cluster. Algorithm 8 changes the number of requested eigenvalues r in the IRAM (Algorithm 7) as the spectrum of $\mathcal{A}^* y$ evolves. As we saw in Section 6.5, changes in the choice of r appear to correlate with clustering of the algebraically largest eigenvalues of $\mathcal{A}^* y$. In this sense, the value r selected by Algorithm 8 may be a proxy for determining the rank ρ of $\mathcal{A}^* y_\star$. Knowledge of the expected value for ρ may provide both theoretical justification for Algorithm 8 and a means to improve Algorithm 8.

Many computational questions also remain for future work. In terms of solving the EMEP (6.2.2), we did not examine parallel methods for solving this sequence of problems.[1] We also did not discuss optimizing the method for handling the primal recovery problem (4.2.11) in the GDD algorithm. Recall that the primal recovery problem requires 2-5% of the total DFTs in a

[1] Note that we did consider the *inverse free preconditioned Krylov subspace method* (EIGIFP) [**32**], which was comparable to or slightly slower than IRAM. EIGIFP is implemented in MATLAB and this code had some numerical issues for larger PLGD models.

given PLGD model (see Table 6.1). The current method used is *minFunc* [**56**], which is a quasi-Newton method based only on gradient information. However, a recent paper [**65**] indicates that the objective function in (4.2.11) has no spurious local minima and negative directional curvature at all saddle points. Thus the Hessian of this function may also be used to solve (4.2.11). Finally, we did not discuss alternative methods to the GDD algorithm for optimizing the PLGD model itself.[2]

[2] Note that we did consider *minConf* [**57**], a descent method for problems like the PLGD model with an expensive objective function and cheap projection operator. This method appeared comparable to the GDD algorithm and in some cases faster.

APPENDIX A

A Comparison of PhaseLift-type Models and Phase Retrieval Algorithms

In this appendix we compare several PhaseLift-type models and phase retrieval algorithms to demonstrate the advantages of using the Gauge Dual Descent (GDD) algorithm (Algorithm 3) to solve large-scale, unstructured noisy phase retrieval problems (1.0.1). First, we examine several *PhaseLift-type* models, which are either duals or modifications of the PhaseLift model (2.4.4) discussed in Section 2.4. Among these PhaseLift-type models, we explain why the PLGD model is the best suited for developing a first-order optimization method. Next, we demonstrate experimentally that the GDD algorithm is generally more accurate and robust than the wflow algorithm (Algorithm 2) for noisy phase retrieval with minimal oversampling.

In Section 4.2 we reviewed the steps necessary to develop a generic first-order method. In summary, any first-order method for minimizing an objective function $f(x)$ over some convex set $\mathcal{C}$ will require several evaluations of $f(x)$ and its gradient $\nabla f(x)$ or subdifferential (1.0.17) $\partial f(x)$, and several projections onto $\mathcal{C}$. Also note that the method used in this section for creating noisy phase retrieval problems is presented in Section 5.2.

We begin by examining several PhaseLift-type models and discussing the computational costs involved in a typical first-order method for each model. As discussed in equation (2.4.3), any PhaseLift-type model will have as a variable a lifted matrix X of size $n \times n$, where n is the dimension of the desired signal $\mathbf{x}$. Since we are focused on large-scale phase retrieval problems, we seek to avoid repeated partial singular value decompositions (SVDs) of $n \times n$ matrices. For instance, the PhaseLift model (2.4.4) has the objective function $f(X) = ||X||_1$ and constraints $||\mathcal{A}(X) - b||_2 \leq \epsilon$ and $X \succeq 0$. The evaluation of f and its subdifferential require a partial SVD, and projection onto $X \succeq 0$ requries an additional partial SVD. Thus the PhaseLift model (2.4.4) is not well suited for first-order methods.

In contrast with the PhaseLift model (2.4.4), the PLGD model restated from (3.2.1) as

$$\begin{aligned} &\min_{y} && \lambda_1(\mathcal{A}^* y) \\ \text{(PLGD)} \quad &\text{s.t.} && \langle b, y\rangle - \epsilon||y||_2 \geq 1 \end{aligned} \tag{A.0.1}$$

has an objective function $f(y) = \lambda_1(\mathcal{A}^* y)$ whose evaluation and gradient require a standard eigenvalue problem, and projection onto $\mathcal{C} = \{y \mid \langle b, y\rangle - \epsilon||y||_2 \geq 1\}$ is an $\mathcal{O}(n)$ operation (see Section 4.2 for details). Likewise, recovery of the desired approximate signal x from the variable y involves a wflow-like problem which is very efficient for large-scale problems (see Section 4.2). Thus the PLGD model (A.0.1) is well suited for developing first-order methods.

Another PhaseLift-type model is the PhaseLift Lagrange dual

$$\begin{aligned} &\max_{y} && \langle b, y\rangle - \epsilon||y||_2 \\ \text{(PLD)} \quad &\text{s.t.} && I \succeq \mathcal{A}^* y \end{aligned} \tag{A.0.2}$$

(see [**8**, Chapter 5] for derivation). The PLD model (A.0.2) has a simple objective function to evaluate. Yet the PLD constraint is a complicated linear matrix inequality and projection onto the feasible set $\{y \mid I \succeq \mathcal{A}^* y\}$ is a separate eigenvalue optimization problem similar to PLGD model (A.0.1). Thus the PLGD model is better suited for first-order methods than the PLD model (A.0.2).

Another method for dualizing the PhaseLift model (2.4.4) is considered in [**16**] where the authors demonstrate the efficacy of the PhaseLift model by applying a Lagrange multiplier to the constraint $||\mathcal{A}(X) - b||_2 \leq \epsilon$ in (2.4.4), giving the model

$$\begin{aligned} &\min && \tfrac{1}{2}||\mathcal{A}(X) - b||_2^2 + \tau||X||_1 \\ &\text{s.t.} && X \succeq 0. \end{aligned} \tag{A.0.3}$$

To optimize (A.0.3), the authors use the Templates for First-Order Conic Solvers (TFOCS) package [**5**], which dualizes a given model, applies a smoothing, and then solves the smooth dual with a first-order method. For the appropriate value $\tau(\epsilon)$, the model (A.0.3) is equivalent to the PhaseLift model (2.4.4) [**51**, Section 28]. Thus (2.4.4) is solved by maximizing $\tau(\epsilon)$ with a bisection method which requires solving a sequence of models (A.0.3) using TFOCS.

This PhaseLift bisection method successfully demonstrates that the accuracy of the PhaseLift optimal signal improves with oversampling and deceases gradually as the error ϵ increases [**16**, Section 7]. However, the resulting algorithm is computationally expensive, as the function $||X||_1$ in the objective requires a partial SVD and the constraint $||\mathcal{A}(xx^*) - b||_2 \leq \epsilon$ must be embedded into the objective. Additionally, this algorithm tends to fail without sufficient oversampling and is unable to achieve a high level of accuracy for noiseless models [**27**, Section 5, Table 1].

Another potential PhaseLift-type model we may consider optimizing is the *PhaseLift-l_1 model*

$$\begin{array}{lll} (\text{PLP-}l_1) & \min & ||\mathcal{A}(X) - b||_1 \\ & \text{s.t.} & X \succeq 0 \end{array} \tag{A.0.4}$$

discussed in [**13**]. This paper showed that the number of observations required in Theorem 2.4.2 to guaranteed the solution signal error bounds (2.4.7), (2.4.8) are reduced to $\mathcal{O}(n)$ if we instead consider the PhaseLift-l_1 model (A.0.4). Therefore we will investigate whether (A.0.4) or its duals could be used to develop a computationally efficient large-scale first-order method.

A first-order method for the PhaseLift-l_1 model (A.0.4) will require projection onto the positive semidefinite constraint $X \succeq 0$, again requiring a partial SVD which is prohibitive for large-scale problems. We may also consider the gauge dual and Lagrange dual

$$\begin{array}{llll} & \min\limits_{y} & ||y||_\infty & \\ (\text{PLGD-}l_1) & \text{s.t.} & -\mathcal{A}^* y \succeq 0 & \\ & & \langle b, y \rangle \geq 1 & \end{array} \qquad \begin{array}{llll} & \max\limits_{y} & \langle b, y \rangle & \\ (\text{PLD-}l_1) & \text{s.t.} & -\mathcal{A}^* y \succeq 0 & \\ & & ||y||_\infty \leq 1 & \end{array} \tag{A.0.5}$$

to the PhaseLift-l_1 model (A.0.4). For a derivation of PLD-l_1, see [**8**, Chapter 5]. For a derivation of PLGD-l_1, we have the following proof.

PROOF. To determine the gauge dual of PLP-l_1, first note that the objective function $f(X) = ||\mathcal{A}(X) - b||_1$ is not a gauge function, as f is not positively homogeneous. Using the substitution $z = b - \mathcal{A}(X)$ and extending the linear operator $\mathcal{A}$, we may restate PLP-l_1 as

$$\begin{array}{ll} \min\limits_{X,z} & \kappa(X, z) := ||z||_1 + \delta_{\succeq 0}(X) \\ \text{s.t.} & \overline{\mathcal{A}}(X, z) := \mathcal{A}(X) + z = b, \end{array} \tag{A.0.6}$$

which is now in the form of the primal-gauge dual pair (3.2.21). The gauge functions $\kappa_1(z) = ||z||_1$ and $\kappa_2(X) = \delta_{\succeq 0}(X)$ have polars $\kappa_1^\circ(w) = ||w||_\infty$ and $\kappa_2^\circ(V) = \delta_{\succeq 0}(-V)$. Thus, by Proposition 3.3.5, κ has the polar

$$\kappa^\circ(V, w) = \max\{||w||_\infty,\ \delta_{\succeq 0}(-V)\} = ||w||_\infty + \delta_{\succeq 0}(-V). \tag{A.0.7}$$

Additionally, the adjoint of $\overline{\mathcal{A}}$ is $\overline{\mathcal{A}}^* y = (\mathcal{A}^* y, y)$, giving

$$\kappa^\circ(\overline{\mathcal{A}}^* y) = ||y||_\infty + \delta_{\succeq 0}(-\mathcal{A}^* y). \tag{A.0.8}$$

Passing $\delta_{\succeq 0}(-\mathcal{A}^* y)$ into the constraint set, we see that PLGD-l_1 is the gauge dual of PLP-l_1.

□

Like the PhaseLift Lagrange dual (A.0.2), both of these models (A.0.5) have a linear matrix inequality in the constraint. As a result, projection onto this constraint $\{y \mid -\mathcal{A}^* y \succeq 0\}$ will again require a separate eigenvalue optimization problem similar to PLGD model (A.0.1), making both models (A.0.5) computationally prohibitive to optimize. Thus, among the PhaseLift-type models discussed above, the PLGD model (A.0.1) is the best suited for developing a first-order method.

Next, we demonstrate that the GDD algorithm (Algorithm 3 of Section 4.2) is typically more accurate and robust than the wflow algorithm (Algorithm 2) for noisy phase retrieval problems (1.0.1) with minimal oversampling.

To compare these two algorithms, we generate a set of noisy phase retrieval problems using the method described in Section 5.2 (the phase retrieval problem with Gaussian noise (5.2.3)). Successful signal recovery occurs when the approximate observation $\mathcal{A}(xx^*)$ as defined in (2.4.2) closely matches the true observation $\mathbf{b} = \mathcal{A}(\mathbf{x}\mathbf{x}^*)$ rather than the noisy observation b. Thus we use the primal true relative error

$$\frac{||\mathcal{A}(xx^*) - \mathbf{b}||_2}{||\mathbf{b}||_2} \leq \tau \epsilon_{\text{rel}} \tag{A.0.9}$$

to measure the accuracy of these algorithms. A signal is considered successfully recovered if it satisfies the inequality (A.0.9), where $\tau = 1$ indicates accuracy within the expected error, and $\tau < 1$ indicates a higher level of accuracy.

The ability of the GDD algorithm to denoise the noisy observation $b = \mathbf{b} + \eta$ is a consequence of the property that the PLGD variable y_k approximates the noise term η with increasing accuracy as the GDD algorithm converges (see Section 3.4 for details regarding the primal refinement method). Thus we also measure the angle between η and the final dual variable y returned by the GDD algorithm

$$\cos \angle(\eta, y) = \frac{\eta^* y}{||\eta||_2 \; ||y||_2}. \tag{A.0.10}$$

Table A.1 displays the results of the GDD algorithm and wflow algorithm.

		GDD				wflow		
				% success			% success	
L	ϵ_{rel}	$\cos \angle(\eta, y_{100})$	xErr	$\tau = 1.0$	$\tau = 0.8$	xErr	$\tau = 1.0$	$\tau = 0.8$
4	0.050	1.12_{-1}	1.50_{-1}	0.86	0.00	3.92_{-1}	0.01	0.01
4	0.150	3.57_{-1}	6.21_{-1}	0.07	0.00	5.58_{-1}	0.00	0.00
4	0.300	5.65_{-1}	1.17_{0}	0.30	0.00	1.00_{0}	0.04	0.00
6	0.050	1.89_{-1}	6.92_{-2}	1.00	0.96	1.25_{-1}	0.64	0.64
6	0.150	4.27_{-1}	2.58_{-1}	1.00	0.93	2.40_{-1}	0.49	0.49
6	0.300	6.12_{-1}	6.72_{-1}	1.00	0.20	4.21_{-1}	0.64	0.32
8	0.050	4.00_{-1}	4.61_{-2}	1.00	1.00	4.53_{-2}	1.00	1.00
8	0.150	5.32_{-1}	1.55_{-1}	1.00	1.00	1.42_{-1}	0.98	0.98
8	0.300	6.68_{-1}	4.01_{-1}	1.00	1.00	2.97_{-1}	0.98	0.94

TABLE A.1. Rate of successful signal recovery and mean residual values for sets of 100 noisy phase retrieval problems with random Gaussian signals of size $n = 128$ with oversampling rate L and relative error ϵ_{rel}. The term *xErr* is signal relative error $||xx^* - \mathbf{x}\mathbf{x}^*||_F / ||\mathbf{x}\mathbf{x}^*||_F$. Recovery is determined successful if the inequality (A.0.9) is satisfied for a given τ. The GDD algorithm (Algorithm 3) is set to terminate at 100 iterations. Numbers n_{-k} are shorthand for $n \times 10^{-k}$.

As we see in Table A.1, the GDD algorithm generally has a greater likelihood of successful recovery than the wflow algorithm when observations have a lower rate of oversampling, regardless of the noise level. Additionally, if models have greater noise and greater oversampling, this increases

the accuracy of the PLGD variable y approximating the noise term η. Figure A.1 depicts the ability of the GDD algorithm to recover a much larger signal with moderate noise and minimal oversampling.

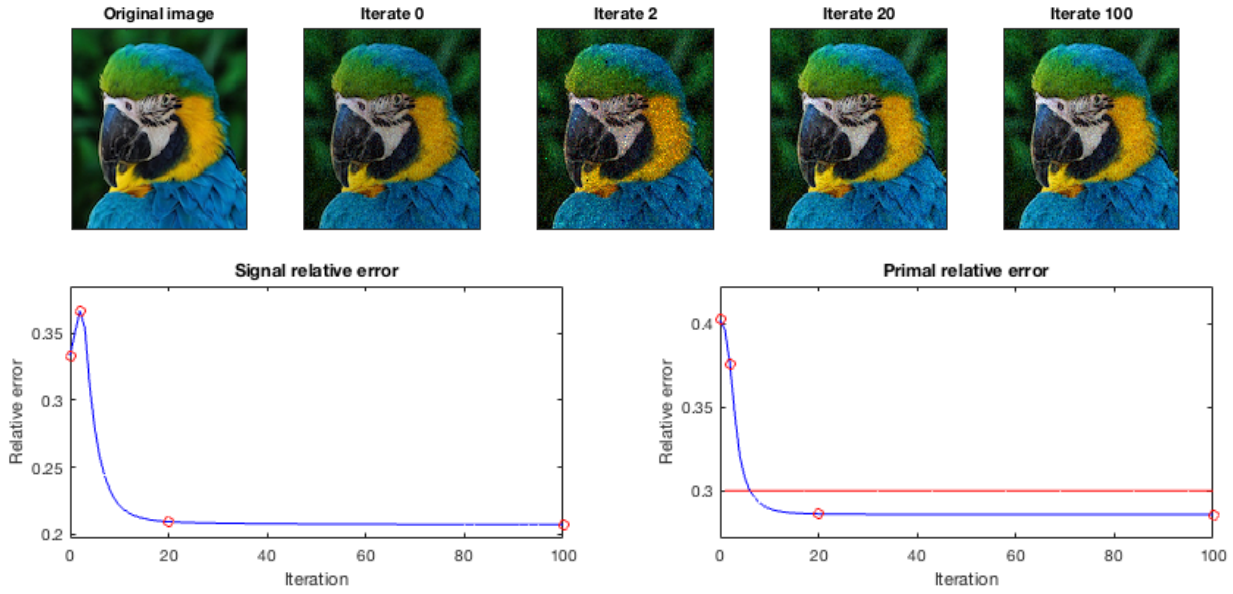

FIGURE A.1. Results from the GDD algorithm (Algorithm 3) applied to a test image separated into its three RGB channels. Left: signal relative error $||xx^* - \mathbf{x}\mathbf{x}^*||_F/||\mathbf{x}\mathbf{x}^*||_F$. Right: primal relative error $||\mathcal{A}(xx^*) - b||_2/||b||_2$ with red line indicating primal feasibility. Red circles denote the iterate signals pictured above. Original signal is 128×128 pixels, with an oversampling of $L = 8$ and noise ratio $\epsilon_{\text{rel}} = 0.30$. Measurements use the mean of the three color channel values.

Figure A.1 demonstrates the tendency of the GDD algorithm to make significant progress during early iterates. When the same set of models in Figure A.1 were solved with the wflow algorithm, the red channel converged to an infeasible solution (with primal residual 0.300068), the green channel converged to a feasible solution (with primal residual 0.288473), and the blue channel diverged.

Table A.1 and Figure A.1 highlight the effectiveness of the GDD algorithm for noisy phase retrieval problems. The GDD algorithm is typically is more robust than the wflow algorithm and tends to recover signals successfully when there is minimal oversampling. Thus, among the phase retrieval models and methods considered in Chapter 2 and this appendix, the PLGD model (A.0.1) is the best suited for large-scale, unstructured noisy phase retrieval problems.

APPENDIX B

Further Justification of New Termination Conditions in Section 5.4

Section 5.4 demonstrated that the primal difference (5.2.7e) and dual variable difference (5.2.7i) effectively identify the point at which the GDD algorithm (Algorithm 3) stagnates for PLGD models with Gaussian noise (5.2.3). This appendix demonstrates that all of the remaining residuals from (5.2.7) are either unreliable or effectively a duplicate of another computationally cheaper residual.

One residual that may be eliminated is the primal relative error (5.2.7d). The original termination conditions for the GDD algorithm include the feasibility requirement (4.2.13)

$$||\mathcal{A}(xx^*) - b||_2 \leq \epsilon + \text{tol}_{\text{feas}}(1 + ||b||_2),$$

which is equivalent to the primal relative error (5.2.7d) using the relation $\epsilon_{\text{rel}} = \epsilon/||b||_2$

$$\frac{||\mathcal{A}(xx^*) - b||_2}{||b||_2} \leq \epsilon_{\text{rel}} + \text{tol}_{\text{feas}} \left(\frac{1 + ||b||_2}{||b||_2} \right). \tag{B.0.1}$$

As indicated in Table 5.4, the GDD algorithm typically requires only a few iterations before a primal feasible signal x is found. Yet in certain cases the GDD algorithm may never identify a feasible signal. Figure B.1 demonstrates plots the primal relative error (5.2.7d) for the particular model from Figure 5.3 which never attained primal feasibility.

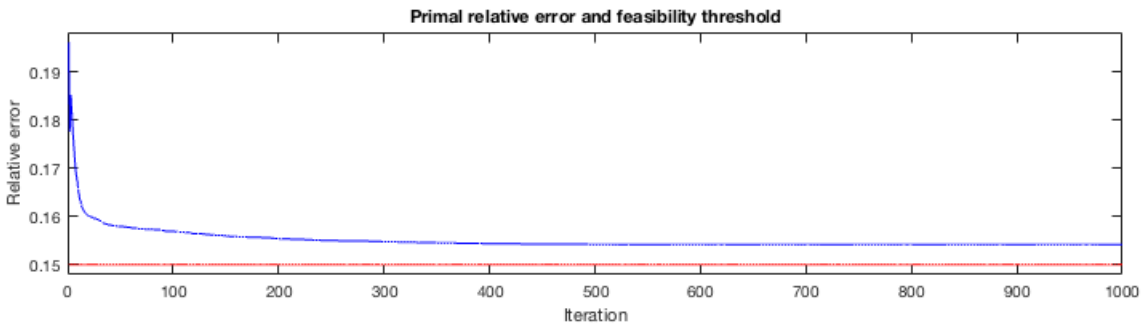

FIGURE B.1. Primal relative error (5.2.7d) (blue) and noise ratio threshold (red) for noise ratio $\epsilon_{\text{rel}} = 0.15$ and oversampling rate $L = 5$. Iterate 1,000 remains infeasible with a primal relative error of 0.1541.

The experiment in Figure B.1 demonstrates a model where both the exact tolerance $\epsilon_{\text{rel}} = 0.15$ and the relaxed tolerance $\epsilon_{\text{rel}} + \text{tol}_{\text{feas}} \left(\frac{1+||b||_2}{||b||_2} \right) = 0.1502$ as suggested in (4.2.13) are unattainable by the GDD algorithm. Nevertheless, the GDD algorithm makes significant progress in recovering an approximate signal. Notably, the wflow algorithm applied to the same model recovers a signal with a primal relative error (5.2.7d) of 0.3165. Thus we ignore the primal feasibility condition (4.2.13).

We may also disregard the duality gap (5.2.7f) as a termination condition. The experiments in Table 5.3 demonstrate that the duality gap tends to stagnate at different values for differing noise level and oversampling rate. Table B.1 indicates the final duality gap value for each experiment from Figure 5.3, further demonstrating this residual is not a reliable indicator of stagnation.

	$L = 5$	$L = 10$
$\epsilon_{\text{rel}} = 0.05$	8.24	8.10
$\epsilon_{\text{rel}} = 0.15$	39.23	29.41
$\epsilon_{\text{rel}} = 0.30$	38.93	59.73

TABLE B.1. Final values of duality gap (5.2.7f) after 1,000 iterations of the GDD algorithm (Algorithm 3) with indicated noise ratios and oversampling rates.

The duality gap (5.2.7f) is not reliable because this measurement is dependent on the accuracy of both the dual objective value λ_1 and the approximate signal norm $||xx^*||_1$ as compared to

their respective optimal values. As established in Section 5.3, the GDD algorithm cannot achieve optimality for most noisy phase retrieval models. Thus the complementarity condition $||xx^*||_1 \cdot \lambda_1 = 1$ (Corollary 3.4.1, d) cannot be achieved. As a result, the duality gap (5.2.7f) stagnates unpredictably for varying oversampling rates and noise levels and is not used as a termination condition for the GDD algorithm.

Next we demonstrate that the duality gap difference (5.2.7g) is a redundant measurement. Denote the primal objective value as $p = ||xx^*||_1$ and its update with a hat. As the GDD algorithm stagnates, $\hat{p}/p$ approaches 1. Additionally, note that typical optimal signals have fairly large norms (i.e., $||\mathbf{x}\mathbf{x}^*||_F = \mathcal{O}(10^3)$ or larger). If we assume $\hat{p}/p \approx 1$ and $\lambda_1 - 1/p \approx \lambda_1$ for later iterates then we have

$$\begin{aligned} \frac{|\gamma - \hat{\gamma}|}{\gamma} &= \frac{|(p\lambda_1 - 1) - (\hat{p}\hat{\lambda}_1 - 1)|}{p\lambda_1 - 1} \\ &= \frac{\left|\lambda_1 - \frac{\hat{p}}{p}\hat{\lambda}_1\right|}{\lambda_1 - \frac{1}{p}} \\ &\approx \frac{|\lambda_1 - \hat{\lambda}_1|}{\lambda_1}. \end{aligned} \tag{B.0.2}$$

Thus the duality gap difference (5.2.7g) behaves similarly to the dual objective difference (5.2.7h) and may be disregarded.

Finally, the dual matrix difference (5.2.7j) is also a redundant measurement which may be disregarded. Note that the norm difference of dual matrices is bounded above by the dual variable difference (5.2.7i), since

$$||A - \hat{A}|| = ||\mathcal{A}^*(y - \hat{y})|| \leq ||\mathcal{A}^*||\ ||y - \hat{y}||_2,$$

where we have the induced norm of $\mathcal{A}^*$

$$||\mathcal{A}^*|| = \sup_{||w||_2 = 1} ||\mathcal{A}^* w||.$$

And computationally, the dual variable difference (5.2.7i) is computed with a vector norm, where as the dual matrix difference (5.2.7j) requires an additional eigenvalue computation (in this case

the largest magnitude eigenvalue). Thus the dual matrix difference (5.2.7j) can be ignored due to its excessive computational cost and close relationship to the dual variable difference (5.2.7i).

Finally, we may also eliminate the dual objective difference (5.2.7h) as a candidate residual for termination. Figure B.2 depicts the behavior of this residual for the models in Figure 5.3. As with Figure 5.4, the vertical axis indicates specific tolerances and the horizontal axis indicates the first iterate at which the GDD algorithm would satisfy this tolerance.

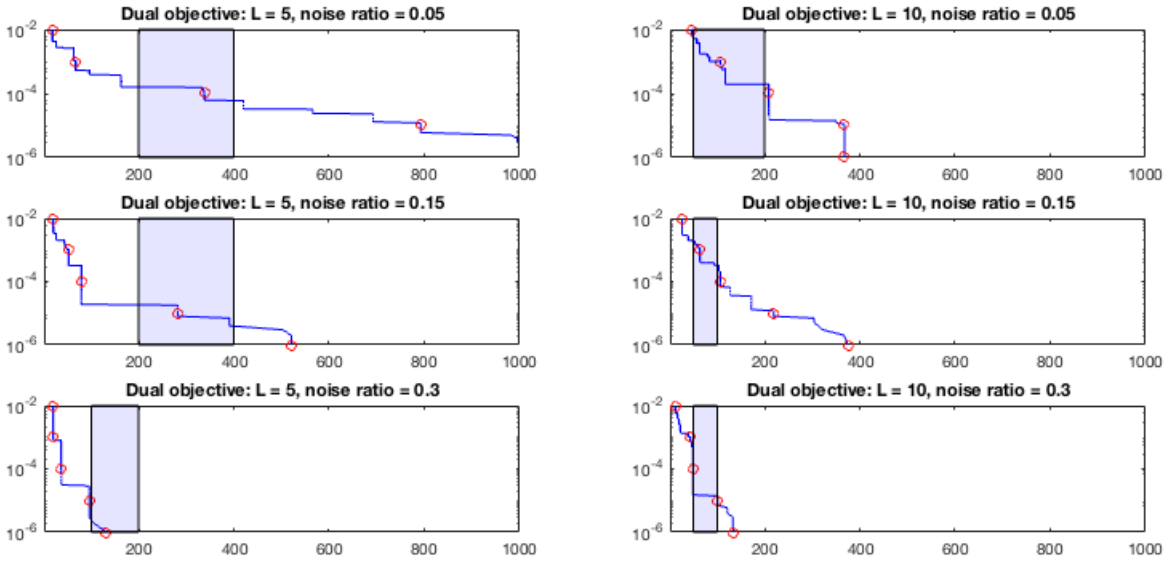

FIGURE B.2. Plots of dual objective difference values (5.2.7h) against the iterate at which the GDD algorithm (Algorithm 3) first satisfies this tolerance for the models discussed in Figure 5.3. Red circles are placed at tolerances 10^{-n}. The blue rectangles indicate the proposed intervals of termination from Table 5.5.

Figure B.2 demonstrates that the dual objective difference (5.2.7h) behaves erratically, decreasing too slowly for low-noise models and too quickly for high-noise models. To simplify our discussion, label the dual objective tolerance as tol_{DO}. If we set $\text{tol}_{\text{DO}} = 10^{-5}$, then the GDD algorithm will terminate far too late for the top-left model in Figure B.2. However, if we set $\text{tol}_{\text{DO}} = 10^{-4}$ then the GDD algorithm may terminate far too early for the middle-left and bottom-left models. Likewise, the models at right in Figure B.2 do not have a consistent tolerance value tol_{DO} which reliably selects for termination within the desired intervals. Thus the tolerance tol_{DO} is not a reliable indicator of stagnation of the GDD algorithm.

Bibliography

[1] A. Y. Aravkin, J. V. Burke, D. Drusvyatskiy, M. P. Friedlander, and K. J. MacPhee, *Foundations of gauge and perspective duality*, SIAM Journal on Optimization, 28 (2018), pp. 2406–2434.

[2] S. Bahmani and J. Romberg, *Phase retrieval meets statistical learning theory: A flexible convex relaxation*, arXiv preprint arXiv:1610.04210, (2016).

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, vol. 1, SIAM, 2000.

[4] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA Journal of Numerical Analysis, 8 (1988), pp. 141–148.

[5] S. R. Becker, E. J. Candès, and M. C. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Mathematical Programming Computation, 3 (2011), p. 165.

[6] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, vol. 2, SIAM, 2001.

[7] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, third ed., 2016.

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

[9] R. N. Bracewell and R. N. Bracewell, *The Fourier Transform and Its Applications*, vol. 31999, McGraw-Hill New York, third ed., 1986.

[10] O. Bunk, A. Diaz, F. Pfeiffer, C. David, B. Schmitt, D. K. Satapathy, and J. F. van der Veen, *Diffractive imaging for periodic samples: Retrieving one-dimensional concentration profiles across microfluidic channels*, Acta Crystallographica Section A: Foundations of Crystallography, 63 (2007), pp. 306–314.

[11] T. T. Cai, X. Li, Z. Ma, et al., *Optimal rates of convergence for noisy sparse phase retrieval via thresholded Wirtinger flow*, The Annals of Statistics, 44 (2016), pp. 2221–2251.

[12] E. J. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski, *Phase retrieval via matrix completion*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 199–225.

[13] E. J. Candès and X. Li, *Solving quadratic equations via phaselift when there are about as many equations as unknowns*, Foundations of Computational Mathematics, 14 (2014), pp. 1017–1026.

[14] E. J. Candès, X. Li, and M. Soltanolkotabi, *Phase retrieval via Wirtinger flow: Theory and algorithms*, IEEE Transactions on Information Theory, 61 (2015), pp. 1985–2007.

[15] E. J. Candes, J. K. Romberg, and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Communications on Pure and Applied Mathematics, 59 (2006), pp. 1207–1223.

[16] E. J. Candes, T. Strohmer, and V. Voroninski, *Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming*, Communications on Pure and Applied Mathematics, 66 (2013), pp. 1241–1274.

[17] A. Chai, M. Moscoso, and G. Papanicolaou, *Array imaging using intensity-only measurements*, Inverse Problems, 27 (2010), p. 015005.

[18] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM Review, 43 (2001), pp. 129–159.

[19] P. Comon and G. H. Golub, *Tracking a few extreme singular values and vectors in signal processing*, Proceedings of the IEEE, 78 (1990), pp. 1327–1343.

[20] X. G. Doukopoulos and G. V. Moustakides, *Fast and stable subspace tracking*, IEEE Transactions on Signal Processing, 56 (2008), pp. 1452–1465.

[21] H. Duadi, O. Margalit, V. Mico, J. A. Rodrigo, T. Alieva, J. Garcia, and Z. Zalevsky, *Digital holography and phase retrieval*, in Holography, Research and Technologies, InTech, 2011.

[22] Y. C. Eldar and S. Mendelson, *Phase retrieval: Stability and recovery guarantees*, Applied and Computational Harmonic Analysis, 36 (2014), pp. 473–494.

[23] V. Elser, T.-Y. Lan, and T. Bendory, *Benchmark problems for phase retrieval*, arXiv preprint arXiv:1706.00399, (2017).

[24] J. R. Fienup, *Phase retrieval algorithms: A comparison*, Applied Optics, 21 (1982), pp. 2758–2769.

[25] J. R. Fienup and J. C. Dainty, *Phase retrieval and image reconstruction for astronomy*, Image Recovery: Theory and Application, 231 (1987), p. 275.

[26] R. M. Freund, *Dual gauge programs, with applications to quadratic programming and the minimum-norm problem*, Mathematical Programming, 38 (1987), pp. 47–67.

[27] M. P. Friedlander and I. Macedo, *Low-rank spectral optimization via gauge duality*, SIAM Journal on Scientific Computing, 38 (2016).

[28] M. P. Friedlander, I. Macedo, and T. K. Pong, *Gauge optimization and duality*, SIAM Journal on Optimization, 24 (2014), pp. 1999–2022.

[29] R. W. Gerchberg and W. O. Saxton, *A practical algorithm for the determination of phase from image and diffraction plane pictures*, Optik, 35 (1972), pp. 237–250.

[30] T. Goldstein and C. Studer, *Phasemax: Convex phase retrieval via basis pursuit*, IEEE Transactions on Information Theory, (2018).

[31] G. H. Golub and C. F. Van Loan, *Matrix Computations*, JHU Press, fourth ed., 2013.

[32] G. H. Golub and Q. Ye, *An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems*, SIAM Journal on Scientific Computing, 24 (2002), pp. 312–334.

[33] R. D. Grigorieff, *A note on von Neumann's trace inequality*, Mathematical News, 151 (1991), pp. 327–328.

[34] M. Guizar-Sicairos, *Phase retrieval for x-ray coherent lensless imaging*, 2018.

[35] R. W. Harrison, *Phase problem in crystallography*, Journal of the Optical Society of America A, 10 (1993), pp. 1046–1055.

[36] K. Jaganathan, Y. C. Eldar, and B. Hassibi, *Phase retrieval: An overview of recent developments*, arXiv preprint arXiv:1510.07713, (2015).

[37] X. Jiang, H.-C. So, and X. Liu, *Robust phase retrieval via ADMM with outliers*, arXiv preprint arXiv:1702.06157, (2017).

[38] V. Katkovnik, *Phase retrieval from noisy data based on sparse approximation of object phase and amplitude*, arXiv preprint arXiv:1709.01071, (2017).

[39] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, vol. 6, SIAM, 1998.

[40] A. Levi and H. Stark, *Image restoration by the method of generalized projections with application to restoration from magnitude*, in IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '84, San Diego, California, USA, March 19-21, 1984, 1984, pp. 88–91.

[41] A. V. Martin, F. Wang, N. Loh, T. Ekeberg, F. R. Maia, M. Hantke, G. van der Schot, C. Y. Hampton, R. G. Sierra, A. Aquila, et al., *Noise-robust coherent diffractive imaging with a single diffraction pattern*, Optics Express, 20 (2012), pp. 16650–16661.

[42] J. Miao, P. Charalambous, J. Kirz, and D. Sayre, *Extending the methodology of x-ray crystallography to allow imaging of micrometre-sized non-crystalline specimens*, Nature, 400 (1999), p. 342.

[43] J. Miao, T. Ishikawa, Q. Shen, and T. Earnest, *Extending x-ray crystallography to allow the imaging of noncrystalline materials, cells, and single protein complexes*, Annual Review of Physical Chemistry, 59 (2008), pp. 387–410.

[44] R. P. Millane, *Phase retrieval in crystallography and optics*, Journal of the Optical Society of America A, 7 (1990), pp. 394–411.

[45] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM Journal on Computing, 24 (1995), pp. 227–234.

[46] T. Ngo and Y. Saad, *Scaled gradients on grassmann manifolds for matrix completion*, in Advances in Neural Information Processing Systems, 2012, pp. 1412–1420.

[47] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, second ed., 2006.

[48] N. Parikh, S. Boyd, et al., *Proximal algorithms*, Foundations and Trends in Optimization, 1 (2014), pp. 127–239.

[49] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Review, 52 (2010), pp. 471–501.

[50] M. Reed and B. Simon, *Functional Analysis, Vol. I*, Academic Press, San Diego, 1980.

[51] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[52] J. A. Rodriguez, R. Xu, C.-C. Chen, Y. Zou, and J. Miao, *Oversampling smoothness: An effective algorithm for phase retrieval of noisy diffraction intensities*, Journal of Applied Crystallography, 46 (2013), pp. 312–318.

[53] Y. Saad, *Numerical Methods for Large Eigenvalue Problems: Revised Edition*, vol. 66, SIAM, 2011.

[54] ———, *Analysis of subspace iteration for eigenvalue problems with evolving matrices*, SIAM Journal on Matrix Analysis and Applications, 37 (2016), pp. 103–122.

[55] Y. Saad, J. R. Chelikowsky, and S. M. Shontz, *Numerical methods for electronic structure calculations of materials*, SIAM Review, 52 (2010), pp. 3–54.

[56] M. Schmidt, *minfunc: Unconstrained differentiable multivariate optimization in MATLAB*. `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`, 2005.

[57] ———, *minconf: Projection methods for optimization with simple constraints in MATLAB*. `http://www.cs.ubc.ca/~schmidtm/Software/minConf.html`, 2008.

[58] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy, *Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm*, in Artificial Intelligence and Statistics, 2009, pp. 456–463.

[59] Y. Shechtman, A. Beck, and Y. C. Eldar, *GESPAR: Efficient phase retrieval of sparse signals*, IEEE Transactions on Signal Processing, 62 (2014), pp. 928–938.

[60] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, *Phase retrieval with application to optical imaging: A contemporary overview*, IEEE Signal Processing Magazine, 32 (2015), pp. 87–109.

[61] Y. Shechtman, Y. C. Eldar, A. Szameit, and M. Segev, *Sparsity based sub-wavelength imaging with partially incoherent light via quadratic compressed sensing*, Optics Express, 19 (2011), pp. 14807–14822.

[62] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 357–385.

[63] ———, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations*, in Parallel Numerical Algorithms, Springer, 1997, pp. 119–165.

[64] G. W. Stewart, *An updating algorithm for subspace tracking*, IEEE Transactions on Signal Processing, 40 (1992), pp. 1535–1541.

[65] J. Sun, Q. Qu, and J. Wright, *A geometric analysis of phase retrieval*, in Information Theory (ISIT), 2016 IEEE International Symposium on, IEEE, 2016, pp. 2379–2383.

[66] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3 - A MATLAB software package for semidefinite programming, version 1.3*, Optimization Methods and Software, 11 (1999), pp. 545–581.

[67] A. Walther, *The question of phase retrieval in optics*, Optica Acta: International Journal of Optics, 10 (1963), pp. 41–49.

[68] B. Yang, *Projection approximation subspace tracking*, IEEE Transactions on Signal Processing, 43 (1995), pp. 95–107.

[69] H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM Journal on Optimization, 14 (2004), pp. 1043–1056.

[70] H. Zhang, S. You, Z. Lin, and C. Xu, *Fast compressive phase retrieval under bounded noise*, in AAAI, 2017, pp. 2884–2890.