Phase Retrieval via Eigenvalue Optimization

By

WILLIAM E. WRIGHT

B.A. (Pennsylvania State University) 2006

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

Michael P. Friedlander

Shiqian Ma

Committee in Charge

2019

To...

Contents

William E. Wright
April 2018
Mathematics

Title Here

Abstract

Insert abstract here.

Acknowledgments

Insert acknowledgements here.

CHAPTER 1

Introduction

1.1. Introduction and contributions

Phase retrieval has a wide range of solution methods, yet few exist for handling noisy observations 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) first introduced and analyzed in [**FM16**],

$$\begin{aligned} &\min_{y} \quad \lambda_1(\mathcal{A}^* y) \\ \text{(PLGD)} \quad &\text{s.t.} \quad \langle b, y\rangle - \epsilon||y|| \geq 1. \end{aligned} \tag{1.1.1}$$

The PLGD model is based on the PhaseLift method [**CESV13**] 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 very large-scale recovery problem. The PLGD model (1.1.1) maintains the convergence guarantees of the convex PhaseLift model, yet also allows for efficient first-order methods such as Algorithm 3 in Section 4.2.

In [**FM16**, Section 5], the authors demonstrate that Algorithm 3 is far more efficient than an alternate method for optimizing the PhaseLift model. This algorithm also returns signals with greater accuracy than wflow, another method for phase retrieval (see Section 2.3.3 for details regarding wflow). Yet Algorithm 3 faces two significant challenges for noisy phase retrieval. Computationally, this algorithm requires a large-scale eigenvalue problem to be solved at each step, leading to significant runtime for modest-sized signal recovery. Additionally, when the phase retrieval problem has nontrivial noise, first-order methods for the PLGD model (1.1.1) such as Algorithm 3 often stagnate before converging.

This dissertation offers two main contributions. First, we address the algorithmic stagnation of Algorithm 3 for noisy phase retrieval. We establish new termination conditions for Algorithm

3 which correspond to stagnation of signal recovery progress. Second, we examine the eigenvalue problem in Algorithm 3, the computational bottleneck of this algorithm. We examine the structure of this problem and develop an adaptive method for handling this problem which decreases the computational costs and overall runtime of Algorithm 3 by about $50 - 80\%$.

This dissertation is organized in the following manner. Chapter 2 introduces the phase retrieval problem and provides a survey of phase retrieval methods. We close Chapter 2 by demonstrating that Algorithm 3 is generally more accurate for noisy phase retrieval than the wflow method (the primary competitor method for unstructured phase retrieval). Chapter 3 examines the PLGD model (1.1.1) and presents the gauge duality theory necessary for examining this model. Chapter 4 then develops Algorithm 3, a first-order method for the PLGD model (1.1.1), and demonstrates the effectiveness of this method for noiseless phase retrieval.

Chapter 5 demonstrates that Algorithm 3 typically fails to converge for noisy phase retrieval and establishes new termination conditions for this algorithm. Chapter 6 demonstrates that the evolving matrix eigenvalue problem (EMEP) in Algorithm 3 is the main computational bottleneck. This chapter examines the structure of the EMEP in Algorithm 3 and reviews a few common eigenvalue methods for handling the EMEP. Chapter 7 presents numerical results for the eigenvalue methods applied to the EMEP and develops an efficient, adaptive method for handling the EMEP. Finally, Chapter 8 presents a revised algorithm for optimizing the PLGD model (1.1.1).

1.2. Notation

This dissertation uses the following notation. Additional notation and definitions related to gauge duality are stated in Sections 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, with $||\cdot|| \equiv ||\cdot||_2$. 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.,

$$||A||_p = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p(A) \right)^{\frac{1}{p}}. \tag{1.2.1}$$

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

$$||A||_F = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A) \right)^{1/2}. \tag{1.2.2}$$

Schatten norms are essential to the multiplicative nature of gauge duality as it is used to develop the PLGD model. In contrast, the EMEP requires measurements with the vector-induced 2-norm. Thus we define the generic *matrix norm* as

$$||A|| = \sup_{||v||_2=1} ||Av||_2 = \sigma_{\max}(A). \tag{1.2.3}$$

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 defined as

$$Diag(d) = Diag(d_1, d_2, \ldots, d_n)_{ij} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{else.} \end{cases} \tag{1.2.4}$$

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

$$\delta_{\mathcal{S}}(x) = \begin{cases} 0 & x \in \mathcal{S} \\ +\infty & x \notin \mathcal{S}. \end{cases} \tag{1.2.5}$$

It is easily seen that if $\mathcal{S}$ is convex, then $\delta_{\mathcal{S}}$ will be convex. Thus the indicator function is useful for tasks like embedding a domain constraint of an optimization model into the objective function and computing the gauge dual of the model (see Chapter 3).

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

$$N_{\mathcal{C}}(y_0) = \{ g \in \mathcal{X} \mid \langle g, y - y_0 \rangle \leq 0 \ \ \forall y \in \mathcal{C} \}. \tag{1.2.6}$$

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

$$S^{\perp} = \{v \mid \langle v, w\rangle = 0 \text{ for all } w \in S\}. \tag{1.2.7}$$

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

$$\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A), \tag{1.2.8}$$

where $\lambda_1(A)$ or simply λ_1 is the largest algebraic eigenvalue of A, and $\lambda_n(A)$ or λ_n is the smallest algebraic 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

$$\langle w, (Au - \theta u)\rangle = 0 \qquad \forall w \in S. \tag{1.2.9}$$

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 that induced by the trace

$$\langle A, B\rangle := \operatorname{tr}(A^*B) = \sum_{i=1}^{n} \sigma_i(A^*B). \tag{1.2.10}$$

For a convex set $\mathcal{C}$, define the projection onto this set as $\Pi_{\mathcal{C}}(y)$. Since the PLGD objective function $f(y) = \lambda_1(\mathcal{A}^*y)$ is not generally differentiable, we consider the subdifferential of f. 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

$$\partial f(y_0) = \{g \in \mathcal{X} \mid f(y) \geq f(y_0) + \langle g, y - y_0\rangle \ \ \forall y \in \mathcal{U}\}, \tag{1.2.11}$$

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

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

$$\langle \mathcal{A}x, y\rangle = \langle x, \mathcal{A}^*y\rangle \text{ for all } x \in \mathcal{X}, y \in \mathcal{Y}. \tag{1.2.12}$$

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 $\mathcal{A}^*$ [**RS80**, Section 6.2]. In this

dissertation, we will be concerned specifically with linear operators $\mathcal{A} : \mathcal{H} \to \mathbb{R}^m$, where $\mathcal{H}$ is the set of $n \times n$ Hermitian matrices. It is easily shown that all such linear operators $\mathcal{A}$ will have the form

$$\mathcal{A}(X) = \begin{bmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{bmatrix}, \tag{1.2.13}$$

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

$$\langle \mathcal{A}(X), y \rangle = \sum_{i=1}^{m} \langle A_i, X \rangle y_i = \sum_{i=1}^{m} \langle y_i A_i, X \rangle = \langle X, \sum_{i=1}^{m} y_i A_i \rangle = \langle X, \mathcal{A}^* y \rangle. \tag{1.2.14}$$

Thus we have $\mathcal{A}^* y = \sum_{i=1}^{m} y_i A_i$.

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

$$f\left(x \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{1.2.15}$$

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

$$f(z) = \frac{1}{\pi} e^{-|z|^2}. \tag{1.2.16}$$

CHAPTER 2

Noisy phase retrieval: a brief survey of applications and methods

2.1. Mathematical model

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

$$\begin{aligned} &\text{find} \quad x \\ &\text{s.t.} \quad |\langle a_i, x\rangle|^2 = b_i = \bar{b}_i + \eta_i \quad 1 \le i \le m. \end{aligned} \tag{2.1.1}$$

If $\eta = 0$, then (2.1.1) is the noiseless phase retrieval problem (from [**Fie82**] and [**CLS15**] among many others). In this dissertation however, we are primarily concerned with nontrivial noise and will refer to (2.1.1) with $||\eta|| > 0$ as *noisy phase retrieval*. Additionally, we are concerned with noise η which has a random Gaussian distribution, as discussed in [**CESV13**] and [**FM16**].

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 [**BB86**, 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 (2.1.1) 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 key fork in the road in terms of choosing an appropriate algorithm.

In this chapter, we begin by discussing a few typical applications of phase retrieval and briefly discuss the experimental models used for numerical testing (Chapter 7). We then review several methods for noisy phase retrieval. These methods are split into three classes: alternating projection methods (Section 2.3.1), structured optimization methods (Section 2.3.2) which rely on additional assumptions like sparsity of the signal x, and unstructured optimization methods (Section 2.3.3) which only require an observation vector b and noise ratio $\epsilon_{\text{rel}} = ||\eta||/||b||$. This final class of methods contains the PlaseLift gauge dual (1.1.1) which is central to this dissertation, and therefore we provide greater detail for theoretical guarantees and numerical performance behavior.

2.2. Applications and experimental models

(1) Phase retrieval has a broad range of applications across the sciences, many of which fall into the general category of coherent diffraction imaging (CDI) [**MCKS99**]. This section provides a brief overview of CDI and closes with a description of the phase retrieval experiment models used in [**CESV13**], [**CLS15**], [**FM16**], and this dissertation.

More specific applications of phase retrieval can be found in astronomy [**FD87**], diffraction and array imaging [**BDP$^+$07**] [**CMP10**], microscopy [**MISE08**], optics [**Wal63**], and x-ray crystallography [**Har93**], [**Mil90**] (for a recent benchmark set of crystallography problems, see [**ELB17**]). For a comprehensive introduction to optical phase retrieval and an overview of recent theory and methods, see the survey [**SEC$^+$15**].

(2) CDI is a method for reconstructing 2- or 3-dimensional nano-structures (e.g., nanotubes, nanocrystals, proteins). 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 2.1 below depicts the CDI observation process.

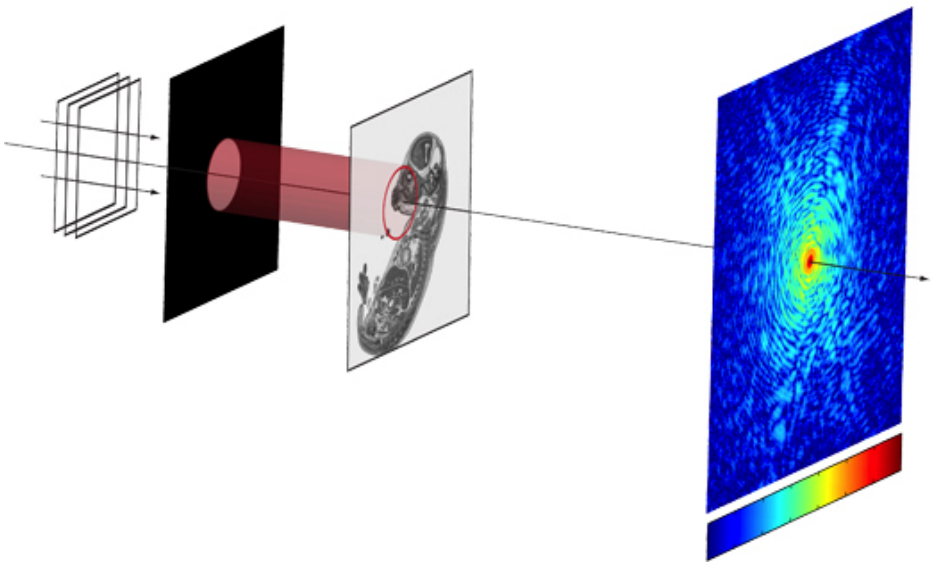

FIGURE 2.1. Depiction of coherent diffractive imaging. Coherent waves (left) are projected at an image (center) which causes a diffraction pattern that is measured by a detector (right). Image from [**GS18b**].

Because CDI does not involve optical lenses, there is no optical aberration (blurring or distorting). The resolution depends is instead dependent on the limits of diffraction and dose. Many efficient methods exist for handling low-noise phase retrieval (see Section 2.3.1 for an overview of a few common methods). However, due to the nonconvexity of the constraints in (2.1.1), these low-noise methods are often 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.

(3) 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 [**DMM$^+$11**] for a discussion of these methods). Our experimental models use the same masking method described in [**CESV13**, Section 2], [**CLS15**, Sections 4.2, 4.3], and [**FM16**, Section 5.1] (see Section 2.3.3 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.

(4) Mathematically, the application of a phase plate to the phase problem 2.1.1 is equivalent to replacing the sensing operator $|Fx|^2$ with $|FC_jx|^2$, where the matrices $C_j \in \mathbb{C}^{n\times n}$ 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

$$\left| \begin{matrix} FC_1x \\ \vdots \\ FC_Lx \end{matrix} \right|^2 = b. \tag{2.2.1}$$

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, making signal recovery difficult for nonconvex methods.

(5) To analyze the effectiveness of various phase retrieval methods, we follow the method for generating experimental models established in [**CLS15**] and extended in [**FM16**]. Figure 2.2 below depicts a test image of size 128×128 and a few particular iterates returned by the method considered in this dissertation (Algorithm 3 in Section 4.2).

FIGURE 2.2. Results from a first-order method (Algorithm 3 in Section 4.2) applied to a test image with oversampling rate $L = 8$ and noise ratio 0.30.

2.3. Survey of noisy phase retrieval methods

Due to the variety of phase retrieval applications and the difficulty of solving the phase retrieval problem (2.1.1), a wide range of phase retrieval methods have been developed for both noisy and noiseless phase retrieval. This section provides a review of methods for noisy phase retrieval (2.1.1 with $||\eta|| > 0$), which we separate into three classes: alternating projection methods (Section 2.3.1), structured optimization methods (Section 2.3.2), and unstructured optimization methods (Section 2.3.3). These three classes were chosen to mirror the historical progression of methods (as the alternating projection methods preceded the others by a few decades) and emphasize the uniqueness of the three unstructured methods discussed in Section 2.3.3. For a recent survey of noiseless phase retrieval methods (2.1.1 with $\eta = 0$), see [**JEH15**].

2.3.1. 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 (2.1.1) and rely on prior information about the signal, such as support constraints or positivity. Originally referred to as *error reduction algorithms*, these methods were later identified as nonconvex alternating projection methods [**LS84**]. Each of these methods relies on projecting an iterate between the object (or time) domain and the frequency domain.

We begin with two common alternating projection methods: the Gershberg-Saxton (GS) algorithm [**GS72**] and the hybrid input-output (HIO) algorithm [**Fie82**], a GS variant still commonly used in practice. Although these methods are usually not effective for noisy phase retrieval and typically require additional prior information, the basic alternating projection algorithm will be useful for our discussion of the wflow algorithm [**CLS15**] (Section 2.3.3), which can also be viewed as an alternating projection method. We then discuss recent alternating projection methods for noisy phase retrieval (oversampling smoothness (OSS) [**RXC$^+$13**], error reduction (ER-) HIO and noise robust (NR-) HIO [**MWL$^+$12**]).

(1) Developed in 1972, the GS algorithm was the first alternating projection method for phase retrieval and serves as an algorithmic foundation which later alternating projection methods would modify to improve the likelihood of convergence.

Algorithm 1 Gershberg-Saxton (GS) algorithm

Input: Frequency measurement vector $b \in \mathbb{R}_+^m$, signal measurement vector $c \in \mathbb{R}_+^m$.

Output: Approximate solution signal x.

1: *Initialize:* Choose a random signal x_0, compute DFT $y_0 = Fx_0$, set $res = |y_0|^2 - b$, $k = 0$.
2: **while** $||res|| >$ tol **do**
3: Compute DFT of x_k and residual: $y_{k+1} = Fx_k$, $res = |y_{k+1}|^2 - b$.
4: Impose frequency magnitude constraints: $[y_{k+1}]_i = \frac{[y_{k+1}]_i}{|[y_{k+1}]_i|}\sqrt{b_i}$ for all $i = 1, \ldots, m$.
5: Compute inverse DFT of y_{k+1}: $x_{k+1} = F^{-1}y_{k+1}$.
6: Impose signal magnitude constraints: $[x_{k+1}]_i = \frac{[x_{k+1}]_i}{|[x_{k+1}]_i|}\sqrt{c_i}$ for all $i = 1, \ldots, m$.
7: $k = k + 1$.
8: **end while**
9: return $x \leftarrow x_k$.

During an iteration of the GS algorithm, knowledge of the object and frequency 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 algorithmic phase retrieval method, this algorithm is also very likely to converge to a local rather than global minima [**JEH15**].

(2) In [**Fie82**], Fienup interpreted the GS algorithm as a nonlinear feedback control system (Figure 2.3), 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 frequence 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 $\bar{x}$, and the object domain constraints 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 object domain constraints.

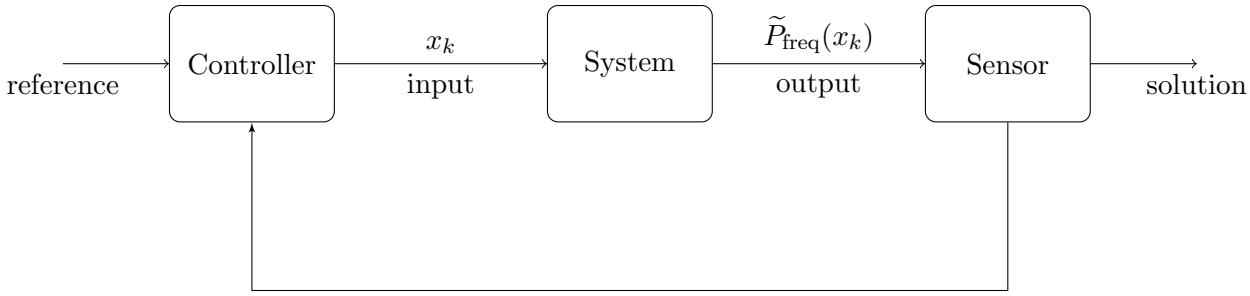

FIGURE 2.3. Phase retrieval as a nonlinear feedback control system

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

$$\widetilde{P}_{\text{freq}}(x + \Delta x) - \widetilde{P}_{\text{freq}}(x) \approx \alpha \Delta x. \tag{2.3.1}$$

Thus 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 (2.3.1). This observation led Fienup to identify three new potential strategies for selecting an update (the *Controller* in Figure 2.3). The most successful of these methods is the HIO algorithm, which has the following index-wise update replacing step 6 in the GS algorithm:

$$[x_{k+1}]_i = \begin{cases} [\widetilde{P}_{\text{freq}}(x_k)]_i & i \notin \mathcal{V} \\ [x_k]_i - \beta [\widetilde{P}_{\text{freq}}(x_k)]_i & i \in \mathcal{V}, \end{cases} \tag{2.3.2}$$

where $\mathcal{V}$ is the set of indices in which the update violates the object domain constraints. This simple corrective step HIO applies to the object domain makes the resulting algorithm much more likely than the GS algorithm to recover a signal from a low-noise observation [**JEH15**]. Nevertheless, HIO has a tendency to converge to local minima and is not robust to noise. In practice, the user will often select a large set of random initial iterates to initialize HIO and take the best resulting signal.

(3) Recent developments in alternating projection methods for phase retrieval have focused on handling noisy observations. In [**RXC$^+$13**], the authors modify the GS algorithm by taking the HIO update (2.3.2) and performing an Gaussian smoothing step in the frequence domain on the indices (pixels) violating the support constraint. Their OSS algorithm takes

the update x_{k+1} from HIO step 6 and applies the additional update:

$$[x'_{k+1}]_i = \begin{cases} [x_{k+1}]_i & i \notin \mathcal{V} \\ \left[F^{-1}\left[F(x_{k+1}) \; .* \; \bar{w}(j,\alpha)\right]\right]_i & i \in \mathcal{V}, \end{cases} \tag{2.3.3}$$

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

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

As $\alpha \to \infty$, the original HIO algorithm is recovered. The authors select α heuristically, with early $\alpha = \mathcal{O}(N)$ and later $\alpha = \mathcal{O}(1/N)$, which causes OSS to behave similar to HIO during early iterations while damping high frequency violations in later iterations. Their experiments [**RXC$^+$13**, 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 $\epsilon_{\text{rel}} = ||\eta||/||b||$). The results indicate the accuracy and consistency of OSS is superior to HIO and two HIO variants: ER-HIO and NR-HIO of [**MWL$^+$12**].

(4) While these HIO-type methods are computationally efficient and still commonly used in practice [**JEH15**], they also requires 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 (2.1.1) as a structured optimization problem.

2.3.2. 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 structures of particular phase retrieval problems. The following survey highlights typical methods, their applications, and theoretical guarantees for exact recovery or error bounds.

These methods are grouped based on the structural property they seek to exploit. *Compressive phase retrieval* methods assume sparsity in the signal x ([**SESS11**], the GrEedy Sparse PhAse

Retrieval (GESPAR) method [**SBE14**], and thresholded wflow [**CLM$^+$16**]). *Robust phase retrieval* methods assume sparsity in either the observation b ([**Kat17**]) or the noise η ([**JSL17**]). *Supervised phase retrieval* methods require an approximate solution signal $\hat{x}$ for initialization ([**GS18a**] and [**BR16**]).

(1) Many methods have been considered for handling compressive phase retrieval. In [**SESS11**], the authors consider sparse signals with noisy observations and proceed by lifting the signal and sensing vectors (for details on lifting, see Section 2.3.3). They construct a rank minimization problem similar to the PhaseLift rank minimization model, 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 authors of [**EM12**] prove signal uniqueness and stable recovery (a property stronger than invertibility) results for a variety of sparsity assumptions. If the signal x is k-sparse and observation b is noiseless, $\mathcal{O}(k\log(n/k))$ observations are necessary for signal uniqueness. If 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 x dense).

The GESPAR method of [**SBE14**] again assumes 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 [**SBE14**, Section 5] indicate that as sparsity increases, GESPAR has a higher recovery probability than PhaseLift and a sparse variant of HIO.

A thresholded version of the wflow algorithm (see Section 2.3.3 for wflow) is considered in [**CLM$^+$16**]. Under the assumption that x is sparse and b is noisy, the authors develop a thresholded gradient descent algorithm by adding $\tau||x||_1$ to the wflow objective function

$$\min_x \quad \tfrac{1}{2m}\sum_{i=1}^{m}\left(|a_i^*x|^2 - b_i\right)^2 + \tau||x||_1. \tag{2.3.4}$$

They show that this phase retrieval analog to the basis pursuit denoising model [**CDS01**] 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 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 [**ZYLX17**]. The authors show that only $\mathcal{O}(k)$ measurements are required for their algorithm to converge within ϵ error in a runtime of $\mathcal{O}(nk\log(1/\epsilon))$.

(2) A few recent papers have considered sparsity in the observation space rather than the signal space, which can be considered robust phase retrieval. In one case, the paper [**JSL17**] assumes the noise η is sparse. Whereas the minimization of the 2-norm is optimal for fitting against Gaussian noise, the 1-norm is optimal for fitting against outliers. Thus the authors suggests minimizing the objective function $||\mathcal{A}(xx^*) - b||_1$ and develop an alternating direction method of multipliers (ADMM) algorithm. Their work provides numerical results demonstrating this method achieves better accuracy than the wflow algorithm when recovering from an observation with Gaussian noise and 10% outliers.

In contrast, the authors of [**Kat17**] consider phase retrieval when the observation vector b is itself sparse and noisy. Their algorithm is infact a 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 wflow and a filtered version of the GS algorithm.

(3) Rather than assuming sparsity of the signal or observations, the authors of [**GS18a**] and [**BR16**] assume the existence of an approximation $\hat{x}$ to the solution signal $\bar{x}$. In this supervised phase retrieval paradigm, the authors define the PhaseMax problem

$$\begin{aligned} \max_{x} \quad & \mathrm{Re}\langle \hat{x}, x\rangle \\ \text{s.t.} \quad & |\langle a_i, x\rangle|^2 \leq b_i, \quad i = 1, \ldots, m \end{aligned} \tag{2.3.5}$$

which has the benefits of being convex and without requiring signal lifting. The dual to this problem is the widely-studied basis pursuit problem, which has several efficient solution techniques as discussed in (2.3.10) below. The authors of [**GS18a**] prove a relationship between the angle between $\bar{x}$ and $\hat{x}$ and the probably of exact recovery, and demonstrate numerically that this probability quickly approaches 1 as oversampling increases.

2.3.3. Unstructured optimization methods. The final class of methods we examine are unstructured optimization methods. Unlike the methods discussed in Section 2.3.2, these methods place no assumptions on the signal x, the sensing vectors a_i, or the observation b other than the knowledge of the noise ratio $\epsilon_{\text{rel}} = ||\eta||/||b||$.

This section begins with an explanation of matrix lifting for the signal and sensing vectors. Next we discuss the PhaseLift method [**CESV13**] and the wflow algorithm [**CLS15**]. Theoretical and numerical results are included to highlight the effectiveness and robustness of these methods (for complete theoretical results, see [**CLS15**], [**SQW16**] for wflow and [**CL14**], [**CSV13**] for PhaseLift). This discussion of the PhaseLift and wflow methods leads to the PhaseLift gauge dual method [**FM16**], another unstructured optimization method which is the subject of Chapter 3.

(1) The PhaseLift model was first introduced in [**CESV13**], with additional theoretical results established in [**CSV13**]. 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 (2.1.1) into a higher-dimensional linear constraint. If we lift the n-dimensional signal x and sensing vectors a_i into rank-one Hermitian matrices $X = xx^*, A_i = a_i a_i^* \in \mathcal{H}$, then the sensing operator $\mathcal{A}$ is defined coordinate-wise to satisfy $[\mathcal{A}(X)]_i = \langle A_i, X\rangle =$

$\text{tr}(a_i a_i^* x x^*) = |\langle a_i, x\rangle|^2$. Thus $\mathcal{A} : \mathcal{H} \to \mathbb{R}^m$ and its adjoint $\mathcal{A}^* : \mathbb{R}^m \to \mathcal{H}$ are defined as

$$\mathcal{A}(xx^*) := \begin{bmatrix} \langle a_1 a_1^*, xx^*\rangle \\ \vdots \\ \langle a_m a_m^*, xx^*\rangle \end{bmatrix} \quad \text{and} \quad \mathcal{A}^* y := \sum_{i=1}^m y_i a_i a_i^*. \tag{2.3.6}$$

In particular, if the experimental model is the one discussed in Section 2.2 then we have

$$\mathcal{A}(xx^*) = \begin{vmatrix} FC_1 x \\ \vdots \\ FC_L x \end{vmatrix}^2 = \text{diag}\left[\begin{pmatrix} FC_1 \\ \vdots \\ FC_L \end{pmatrix} (xx^*) \begin{pmatrix} FC_1 \\ \vdots \\ FC_L \end{pmatrix}^* \right], \tag{2.3.7}$$

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

(2) The lifted rank-one matrix $X = xx^* \in \mathcal{H}$ 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 (2.1.1) is equivalent to the left-most problem in sequence of problems

$$\begin{array}{ll} \text{find} & x \\ \text{s.t.} & \mathcal{A}(xx^*) = b \end{array} \iff \begin{array}{ll} \text{find} & X \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \\ & \text{rank}(X) = 1 \end{array} \iff \begin{array}{ll} \min\limits_{X \in \mathcal{H}} & \text{rank}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0. \end{array} \tag{2.3.8}$$

If a rank-one solution $X = xx^*$ exists then this matrix satisfies the lifted model in the middle of (2.3.8), and the left implication holds. Likewise, the rank minimization model at right in (2.3.8) will have a rank-one solution, and the right implication holds.

The resulting rank minimization problem at right of (2.3.8) is NP hard, as it includes the cardinality minimization problem as a special case (see [**Nat95**], [**RFP10**]). Thus the next step is to relax the discrete, nonconvex objective function $\text{rank}(X)$. To generalize the model for noisy phase retrieval, a norm bound is also applied to the sensing constraint. This gives the following semidefinite program which defines the PhaseLift model [**CESV13**],

[**CSV13**]

$$\begin{aligned} &\min_{X\in\mathcal{H}} && ||X||_1 = \sum_{i=1}^{n}\sigma_i(X) \\ \text{(PhaseLift)}\quad &\text{s.t.} && ||\mathcal{A}(X)-b||_2 \le \epsilon \\ & && X \succeq 0. \end{aligned} \tag{2.3.9}$$

Here the objective function $||X||_1$ refers to the Schatten p-norm (which is expressed as $\sum_{i=1}^{n}\sigma_i(X)$, $\mathrm{tr}(X)$, or $\langle I, X\rangle$ in various literature). Also, the term $\epsilon = ||\eta||$ measures the total noise, whereas $\epsilon_{\text{rel}} = ||\eta||/||b||$ discussed in Section 2.2 measures the noise ratio or relative noise. We choose the Schatten p-norm to highlight the fact that (2.3.9) is the semidefinite analog to the celebrated basis pursuit denoising problem [**CDS01**], [**CRT06**],

$$\begin{aligned} &\min_{x} && ||x||_1 \\ &\text{s.t.} && ||Ax-b||_2 \le \epsilon, \end{aligned} \tag{2.3.10}$$

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

The following two theorems address how much of a relaxation PhaseLift (2.3.9 is from (2.1.1), establishing exact and approximate recovery guarantees for the PhaseLift model [**CL14**], [**CSV13**]. Theorem 2.3.1 applies to the noiseless case (where $\epsilon = 0$ in the PhaseLift model), establishing a probability of equivalence between (2.1.1) and PhaseLift (2.3.9).

THEOREM 2.3.1. *Consider an arbitrary signal $\bar{x}$ 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 $m \ge c_0 n$, where c_0 is a sufficiently large constant. Then in both the real and complex cases, the solution to (2.3.9) is exact with high probability in the sense that the noiseless PhaseLift problem (2.3.9 with $\epsilon = 0$) has a unique solution obeying*

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

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

PROOF. See [**CL14**, Section 2]. □

The assumptions of Theorem 2.3.1 are met by the experimental models used in [**CESV13**], [**CSV13**], [**FM16**], and this dissertation (see Section 2.2, where the masks $C_i \in \mathbb{C}^{n\times n}$ from (2.2.1) have diagonal entries chosen with random Gaussian distribution). Thus, if L is the oversampling rate (i.e., $m = nL$) then Theorem 2.3.1 shows that only $L = \mathcal{O}(1)$ masks are required to guarantee exact signal recover (up to global phase) with high probability.

Theorem 2.3.2 applies to the noisy phase retrieval case ($\epsilon > 0$), where there is no guarantee that the PhaseLift (2.3.9) 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 largest algebraic eigenvalue $\lambda_1 = \lambda_1(X)$, giving

$$x = \sqrt{\lambda_1} v_1. \tag{2.3.11}$$

THEOREM 2.3.2. *Consider an arbitrary signal $\bar{x}$ 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 $\mathbb{C}^n$. And 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 (2.3.9) solution X obeys*

$$||X - \bar{x}\bar{x}^*||_F \leq C_0 \epsilon, \tag{2.3.12}$$

for some positive constant C_0. Additionally, we have

$$||x - e^{i\theta}\bar{x}||_2 \leq C_0 \min(||\bar{x}||_2, \epsilon/||\bar{x}||_2), \tag{2.3.13}$$

where x is defined as in (2.3.11) and we have some $\theta \in [0, 2\pi]$. Both these estimates hold with probability at least $1 - \mathcal{O}(e^{-\gamma \frac{m}{n}})$, where γ is a positive absolute constant.

PROOF. See [**CSV13**, Section 6]. □

The strength of the PhaseLift model lies in its convexity and generality (no signal or observation assumptions are necessary). Yet the weakness is the difficulty in optimizing the objective function $||X||_1$, as its evaluation requires a partial SVD.

(3) Like the PhaseLift method, the recent Wirtinger flow (wflow) algorithm [**CLS15**] also seeks to solve the phase retrieval problem (2.1.1) 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 (2.1.1) as the following least-squares problem

$$\min_x \quad \tfrac{1}{2m}\sum_{i=1}^{m}\left(|a_i^*x|^2-b_i\right)^2 = \tfrac{1}{2m}||\mathcal{A}(xx^*)-b||^2. \tag{2.3.14}$$

While this model is nonconvex, the authors develop an efficient gradient descent-like method which has a provable guarantee on exact recovery when initialized appropriately. Initialization of the wflow algorithm involves a rescaling of the largest algebraic eigenvector v_1 of the sum of the outer products of the measurement vectors, scaled with the observation magnitudes

$$\lambda_1 = \lambda_1(\mathcal{A}^*b), \qquad \mathcal{A}^*b := \sum_{i=1}^{m} b_i a_i a_i^*. \tag{2.3.15}$$

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

$$\mathbb{E}\left[\frac{1}{m}\mathcal{A}^*b\right] = I + 2\bar{x}\bar{x}^*. \tag{2.3.16}$$

Thus for large m, (2.3.15) has a high probability of returning a vector v_1 closely aligned with the solution $\bar{x}$. Intuitively, this initialization can also be seen as maximizing $\langle \mathcal{A}(vv^*), b\rangle = \langle vv^*, \mathcal{A}^*b\rangle = \langle v, [\mathcal{A}^*b]v\rangle$, and hence the eigenvalue problem (2.3.15) will encourage $\mathcal{A}(vv^*)$ to be collinear with b.

With this initialization, the authors recommend a gradient descent-like method with a preset stepsize strategy. Let $f(x) = \frac{1}{2}||\mathcal{A}(xx^*)-b||^2$, and the residual $r = \mathcal{A}(xx^*)-b$. The mapping $f : x \to \frac{1}{2}||\mathcal{A}(xx^*)-b||^2$ from $\mathbb{C}^n$ to $\mathbb{R}$ is not holomorphic, and thus not complex-differentiable. As a result, the authors appeal to Wirtinger derivatives [**CLS15**, Section

6] for the descent direction

$$\begin{aligned} \nabla f(x) &= [\mathcal{A}^*(\mathcal{A}(xx^*) - b)]x \\ &= [\mathcal{A}^* r]x \\ &= \left[\sum_{j=1}^{L} C_j^* F^* \mathrm{Diag}(r_j) F C_j\right] x. \end{aligned} \tag{2.3.17}$$

The stepsize is then chosen using the following heuristics

$$\mu_k = \min\{1 - e^{-k/k_0}, \mu_{\max}\} \qquad k_0 = 330,\ \mu_{\max} = 0.4, \tag{2.3.18}$$

where μ_k starts small and increases [**CLS15**, Section 2]. This descent direction and stepsize choice lead to Algorithm 2.

Algorithm 2 wflow algorithm

Input: Sensing operator $\mathcal{A}$ (2.3.6) with sensing vectors a_i, frequency measurement vector $b \in \mathbb{R}_+^m$.

Output: Approximate solution signal x.

1: *Initialize:* Compute the largest algebraic eigenpair (λ_1, v_1) of $\mathcal{A}^*b$, set $x_0 = \alpha v_1$ where $\alpha^2 = n \sum_{i=1}^m b_i / \sum_{i=1}^m ||a_i||^2$, set $k = 0$.
2: **while** $||res|| >$ tol **do**
3: Compute Wirtinger derivative: $\nabla f(x_x)$ from (2.3.17).
4: Compute stepsize: μ_{k+1} from (2.3.18) and set $\alpha_k = \frac{\mu_{k+1}}{||x_0||^2 m}$.
5: Compute signal update: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$.
6: $k = k + 1$.
7: **end while**
8: return $x \leftarrow x_k$.

In [**CLS15**, 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

$$F(x) = x^* \left(I - \bar{x}\bar{x}^*\right) x - \frac{3}{4}\left(||x||^2 - 1\right)^2. \tag{2.3.19}$$

Interestingly, we observe that this algorithm can also be interpreted as an alternating projection HIO-type method, where computation of the Wirtinger derivative (2.3.17) 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 (2.3.17) 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 (2.3.2) which replaces Algorithm 1, step 6).

Yet unlike other alternating projection methods, the wflow algorithm does not require additional frequency domain information or heuristics. Wflow 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 (2.3.17) provides a heuristic-free frequency domain damping. Consequentially, the wflow algorithm merges the computational simplicity of an alternating projection method (Section 2.3.1) with the broad utility of an unstructured optimization method. This resulting gradient descent-like method is shown to have the following exact recovery probability for noiseless phase retrieval problems.

THEOREM 2.3.3. *Consider an arbitrary signal $\bar{x}$ 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*

$$\mathrm{dist}(x_0, \bar{x}) := \min_{\theta \in [0, 2\pi]} ||x_0 - e^{i\theta}\bar{x}|| \leq \frac{1}{8}||x|| \tag{2.3.20}$$

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 (2.3.20), we have

$$\operatorname{dist}(x_k, \bar{x}) \leq \frac{1}{8}||\bar{x}|| \left(1 - \frac{\mu}{4}\right)^{k/2}. \tag{2.3.21}$$

PROOF. See [**CLS15**, Section 7]. □

Computationally, the wflow algorithm is very efficient at recovering the signal of an oversampled noiseless observation. Numerical experiments in [**CLS15**, Section 4] demonstrate the effectiveness of this algorithm for 1-D random signals, 2-D natural images, and 3-D molecular structures.

More generally, it was proved in [**SQW16**] that if 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 $f(x) = \frac{1}{2}||\mathcal{A}(xx^*) - b||^2$ has the following properties:

- f has no spurious local minima,
- all global minima are equal to $\bar{x}$ up to some phase constant: $x = e^{i\theta}\bar{x}$,
- f has negative directional curvature at all saddle points.

Thus when solving (2.3.14), methods such as wflow do not require specialized initialization, and are likewise guaranteed to find a global minimum. Nevertheless, wflow can diverge when significant noise exists and an insufficient number of observations are present (see Section 5.3).

As we have seen in this section, many phase retrieval methods have been developed to utilize structural properties of the given phase retrieval problem (2.1.1). Yet if the problem is unstructured (only the observation and noise ratio are availalble), then only a few methods are available. We will now examine these methods to establish which is best-suited for handling large-scale, noisy phase retrieval.

2.4. Why the PLGD model? A comparison of methods for large-scale phase retrieval

In this section, we justify the choice of the PLGD model (1.1.1) for handling large-scale, noisy unstructured phase retrieval problems (2.1.1). As discussed in Section 2.3, there are only a few methods for handling unstructured phase retrieval problems. These methods can be grouped into

two sets: the nonconvex methods (HIO and wflow) and the convex-lifted methods (PhaseLift, PLGD, and other PhaseLift variants). We will examine the robustness and computational costs of each method to establish which is best for large-scale, noisy unstructured phase retrieval problems.

(1) xxx here - MERGE THESE 2 SECTIONS

xxx TODOS:

ADD MOTIVATION FOR THIS DISCUSSION:

- no comparison for Gaussian probs (5.2.3) before

- these are two of the most robust methods for unstructured phase retrieval in the noiseless case

demonstrates the efficacy of Algorithm 3 experimentally, showing that this algorithm is more accurate and robust than the wflow algorithm (2.3.14) for noisy phase retrieval problems when minimal oversampling is available

(2) In this appendix we compare the accuracy of Algorithm 3 and wflow for phase retrieval problems with Gaussian noise (5.2.2) with a variety of oversampling and noise levels. Successful signal recovery occurs when the approximate observation $\mathcal{A}(xx^*)$ closely matches the true observation $\bar{b}$ (rather than the noisy observation b). Thus we use the primal true relative error (5.2.7c) to measure the accuracy of these algorithms. A signal is considered successfully recovered if it satisfies the inequality

$$\frac{||\mathcal{A}(xx^*) - \bar{b}||}{||\bar{b}||} \leq \tau \epsilon_{\text{rel}}, \tag{2.4.1}$$

where $\tau = 1$ indicates accuracy within the expected error, and $\tau < 1$ indicates a higher level of accuracy. In Algorithm 3, the accuracy of the primal refinement step (4.2.11) depends on the dual variable y approximating the noise term η. Thus we also measure the angle between the final dual variable y returned by Algorithm 3 and the noise term η

$$\cos \angle(\eta, y) = \frac{\eta^* y}{||\eta||\ ||y||}. \tag{2.4.2}$$

		Algorithm 3				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 2.1. Rate of successful signal recovery and mean residual values for sets of 100 phase retrieval problems with Gaussian noise (5.2.2) with random Gaussian signals of size $n = 128$ with oversampling rate L and relative error ϵ_{rel}. The term $xErr$ is signal relative error (5.2.7a). Recovery is determined successful if the inequality (2.4.1) is satisfied for a given τ. Algorithm 3 is set to terminate at 100 iterations. Numbers n_{-k} are shorthand for $n \times 10^{-k}$.

As we see in Table 2.1, Algorithm 3 generally has a greater likelihood of successful recovery than wflow 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 y approximating η.

(3) The effectiveness of Algorithm 3 is a consequence of the primal refinement step (4.2.11) using the dual variable y to denoise (at least partially) the noisy observation $b = \bar{b} + \eta$. As established in Corollary 3.4.3 (b) and discussed in (3.4.9), if the lifted true signal $X = \bar{x}\bar{x}^*$ is in the set of optimal matrices then the corresponding optimal dual variable $y_\star$ will be a rescaling of the noise term η. As we see in Table 2.1, a dual variable y must simply be sufficiently close to $y_\star$, as measured by $\cos\angle(\eta, y)$, to increase the accuracy of the recovered signal.

(4) Figure 2.4 depicts the ability of Algorithm 3 to recover a much larger signal with moderate noise and minimal oversampling.

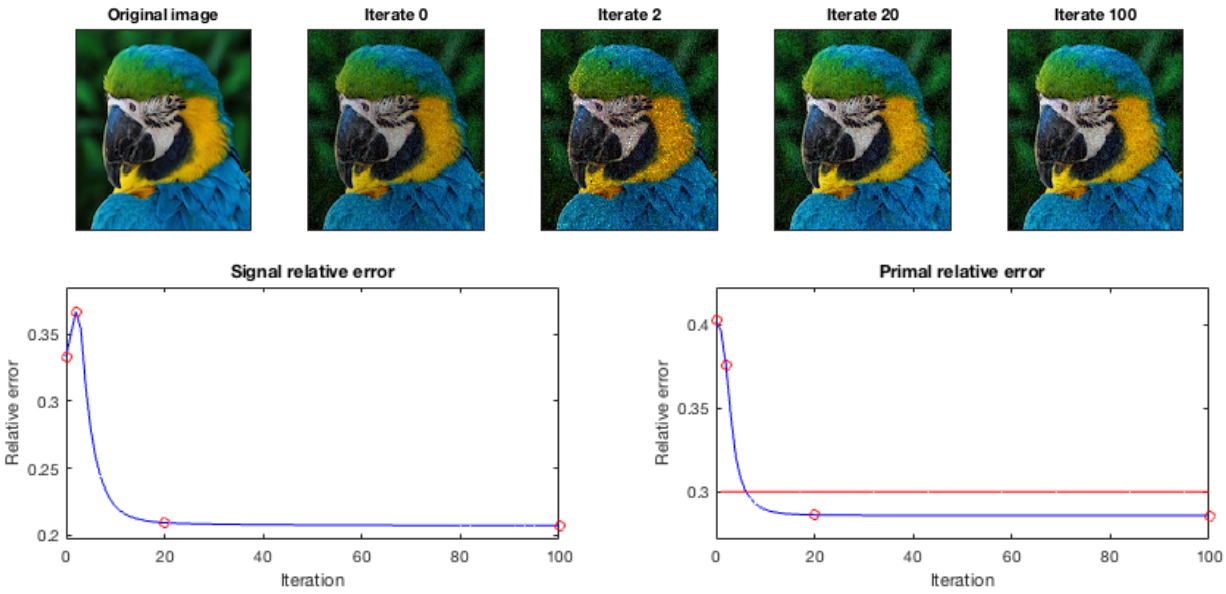

FIGURE 2.4. Results from Algorithm 3 applied to a test image separated into its three RGB channels. Left: signal relative error (5.2.7a). Right: primal relative error (5.2.7d) with red line indicating primal feasibility. Red circles denote the iterate signals pictured above. Original signal is 128×128 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 2.4 demonstrates the tendency of Algorithm 3 to make significant progress during early iterates. When the same set of models in Figure 2.4 were solved with wflow, 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.

(5) Table 2.1 and Figure 2.4 highlight the effectiveness of Algorithm 3 for noisy phase retrieval problems. This algorithm tends to recover signals successfully based on the inequality (2.4.1) when there is minimal oversampling. Additionally, Algorithm 3 is more robust than wflow to minimal oversampling. Yet Figure 2.4 suggests that Algorithm 3 stagnates after making quick initial signal recovery progress. We proceed to examine this stagnation in the next section.

xxx END HERE

(6) We begin by examining the nonconvex methods. As discussed in the previous section, the HIO method (Algorithm 1 with (2.3.2) replacing step 6) and the wflow method (Algorithm 2) are both alternating projection methods which differ only in their choice of frequency and object domain damping (Algorithm 1, steps 4 and 6).

Both algorithms are computationally very simple and ideal for large-scale noiseless phase retrieval, requiring only $2L$ DFTs as the dominant cost per iteration, where $L = m/n$ is the oversampling rate in the phase retrieval model (2.1.1). However, wflow offers two benefits over HIO. The wflow model (2.3.14) has convergence guarantees when oversampling is sufficient, and the wflow method eliminates the need for the additional frequence domain damping parameter, instead using the residual $r = \mathcal{A}(xx^*) - b$. Yet due to the nonconvexity of wflow, this method is not well suited for phase retrieval models with nontrivial noise or low oversampling.

(7) The PhaseLift model (2.3.9) is convex due to the matrix lifting (2.3.6) which defines this model. Thus PhaseLift is well suited for phase retrieval models with noise and low oversampling. The challenge with PhaseLift is the difficulty of developing a first-order optimization method which requires the objective function value and gradient at each step.

To demonstrate the efficacy of the PhaseLift model, the authors of [**CESV13**] apply a Lagrange multiplier to the constraint $||\mathcal{A}(X) = b|| \leq \epsilon$ in (2.3.9), giving the model

$$\begin{aligned} \min \quad & \tfrac{1}{2}||\mathcal{A}(X) - b||^2 + \tau||X||_1 \\ \text{s.t.} \quad & X \succeq 0. \end{aligned} \tag{2.4.3}$$

To optimize (2.4.3), the authors use the TFOCS package (Templates for First-Order Conic Solvers), which dualizes a given model, applies a smoothing, and then solves the smooth dual with a first-order method [**BCG11**]. For the appropriate value $\tau(\epsilon)$, (2.4.3) is equivalent to (2.3.9) [**Roc70**, Section 28]. Thus (2.3.9) is solved by maximizing $\tau(\epsilon)$ with a bisection method which requires solving a sequence of models (2.4.3) using TFOCS.

This PhaseLift bisection method successfully denomstrates that the accuracy of the PhaseLift optimal signal improves with oversampling, and deceases gradually as the noise ratio ϵ_{rel} increases [**CSV13**, Section 7]. However, this method is computationally expensive, as the objective function $||X||_1$ requires a partial SVD and the constraint $||\mathcal{A}(xx^*) - b|| \leq \epsilon$ must be embedded into the objective. Additionally, this method tends to fail without sufficient oversampling and is unable to achieve a high level of accuracy for noiseless models [**FM16**, Section 5, Table 1]. Thus we examine the following convex PhaseLift-type models to determine which is best-suited for first-order optimization.

The PhaseLift model has the following gauge dual and Lagrange dual

$$\begin{array}{llll} & \min\limits_{y} & \lambda_1(\mathcal{A}^* y) & \\ \text{(PLGD)} & \text{s.t.} & \langle b, y\rangle - \epsilon||y|| \geq 1 & \end{array} \qquad \begin{array}{llll} & \max\limits_{y} & \langle b, y\rangle - \epsilon||y|| & \\ \text{(PLD)} & \text{s.t.} & I \succeq \mathcal{A}^* y. & \end{array} \tag{2.4.4}$$

(See Chapter 3 for the derivation of PLGD and [**BV04**, Chapter 5] for PLD.) Evaluation of the PLGD objective function $\lambda_1(\mathcal{A}^* y)$ is a standard eigenvalue problem which also returns the gradient information; and projection onto the constraint set has a simple, closed-form expression (see Section 4.2). The PLD model (2.4.4) 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. Thus the PLGD model is better-suited for first-order methods than the PhaseLift and PLD models.

(8) Another potential PhaseLift-type model we may consider optimizing is an l_1 variant discussed in [**CL14**]. This paper showed that the number of observations required in Theorem 2.3.2 for a guaranteed solution signal error bound can be reduced to $\mathcal{O}(n)$ if we instead consider the l_1 optimization problem

$$\begin{array}{lll} \text{(PLP-}l_1\text{)} & \min & ||\mathcal{A}(X) - b||_1 \\ & \text{s.t.} & X \succeq 0. \end{array} \tag{2.4.5}$$

The oversampling rate of $L = \mathcal{O}(1)$ for the l_1 PhaseLift model (2.4.5) is lower than the rate of $L = \mathcal{O}(\log n)$ required for the PLP model (3.2.1), making (2.4.5) a desirable model

to optimize. Thus we investigate whether (2.4.5) or its duals are suitable for large-scale first-order optimization.

To develop a first-order method for the PLP-l_1 model, we find that projection onto the positive semidefinite constraint requires a partial SVD which is prohibitive for large-scale problems. Thus we consider the PLP-l_1 gauge dual and Lagrange dual

$$\begin{array}{llll} & \min\limits_{y} & ||y||_\infty & \\ \text{(PLGD-}l_1\text{)} & \text{s.t.} & -\mathcal{A}^* y \succeq 0 & \\ & & \langle b, y\rangle \geq 1 & \end{array} \qquad \begin{array}{llll} & \max\limits_{y} & \langle b, y\rangle & \\ \text{(PLD-}l_1\text{)} & \text{s.t.} & -\mathcal{A}^* y \succeq 0 & \\ & & ||y||_\infty \leq 1. & \end{array} \tag{2.4.6}$$

(See Chapter A for the derivation of PLGD-l_1 and [**BV04**, Chapter 5] for PLD-l_1.) Both of these models have a linear matrix inequality in the constraint which again makes them prohibitively difficult to optimize. In fact, projection onto this constraint set involves a separate eigenvalue optimization problem similar to the PLGD model.

In terms of large-scale, noisy unstructured phase retrieval problems, the PLGD model is best suited for first-order optimization methods. In the following two chapters we present the theory behind this model and develop a first-order method for optimizing this model.

CHAPTER 3

Gauge duality theory and the PhaseLift gauge dual model

3.1. Introduction

This chapter examines the PLGD model for solving the phase retrieval problem (2.1.1) and presents the gauge duality theory necessary for examining this model. The PLGD model is the following constrained eigenvalue optimization problem

$$\begin{aligned} \min_{y} \quad & \lambda_1(\mathcal{A}^* y) \\ \text{s.t.} \quad & \langle b, y \rangle - \epsilon ||y||_2 \geq 1. \end{aligned} \tag{3.1.1}$$

We begin in Section 3.2 by establishing the PhaseLift primal and gauge dual (PLP, PLGD) pair of models along with some basic definitions. We then contrast gauge duality with Lagrange duality and provide a brief history of gauge duality theory. Section 3.3 develops a sequence of propositions which are used to prove the gauge duality theorem (Theorem 3.3.1) along with weak duality, strong duality, and optimality conditions for an inequality constrained gauge dual pair. Finally, Section 3.4 uses these results to establish key properties of the primal and gauge dual models which are necessary for developing an efficient method of optimizing the PLGD model and recovering the signal x from the dual variable y.

All of the following gauge properties and gauge duality results were previously established in [**Roc70**], [**Fre87**], [**FMP14**], and [**FM16**]. Gauge functions were first analyzed in [**Roc70**]. Gauge duality was then introduced in [**Fre87**], where the author focused on quadratic programming applications. In [**FMP14**], the authors developed a broad set of antipolar calculus results for the analysis of gauge duality. These results were then applied to the PLP-PLGD pair to develop a first-order method in [**FM16**].

Our contribution is to provide a self-contained, comprehensive treatment of gauge duality theory focused solely on establishing the duality theorem and optimality conditions for the PLP-PLGD pair. Prior to this treatment, the following results were spread throughout the original texts

([**Roc70**], [**FMP14**], and [**FM16**]), occasionally with differing notation or style. In this chapter, we present a single notation and style (Section 4.1) which we use to develop gauge duality theory (Section 3.3) and then apply this theory to the PLP-PLGD pair (Section 3.4). The results of this chapter provide the theoretical foundation for developing a first-order optimization method for the PLGD model in Chapter 4.

3.2. Definitions and primal-dual pairs

(1) The PhaseLift primal semidefinite program (restated from (2.3.9) in Section 2.3.3) and its gauge dual are

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

If $\epsilon > 0$ then we refer to (3.2.1) as the *noisy PLP-PLGD pair*, where the observation vector $b = \bar{b} + \eta$ is the sum of the true observation $\bar{b}$ with some nontrivial noise η as described in the phase retrieval problem (2.1.1). 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.

Following the convention of these major theoretical advances in gauge duality ([**Fre87**], [**FM16**], [**ABD$^+$17**]), all notation and definitions in this section are based on [**Roc70**]. All unproven results in this section are accompanied by a reference to the appropriate section of [**Roc70**]. Along the way, we present three additional, more general primal-gauge dual models which have PLP-PLGD (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 pair (3.2.1).

(2) The PLP-PLGD pair (3.2.1) are examples of a more general primal-gauge dual pair which we define as the *nonlinearly-constrained* pair

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

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). Thus 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 $\kappa, \kappa^\circ : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ 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. The gauge function generalizes the notion of norm, 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 $\rho(y) := ||y||$ 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 [**Roc70**, 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

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

The *preimage* of A 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 $\operatorname{cl}(\mathcal{S})$. The *affine hull* of $\mathcal{S}$ is the set of all affine combinations of elements of $\mathcal{S}$, or

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

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

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

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 $\mathcal{C} = \{x \mid \kappa(x) \leq 1\}$ [**Roc70**, Section 15]. Given a function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$, the *epigraph* of f is defined as

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

Note that f is convex if and only if $\text{epi}(f)$ is convex [**Roc70**, Section 7]. Thus the function f is said to be *closed* if $\text{epi}(f)$ 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 $\text{dom}(f)$) [**Roc70**, Section 7]. Also, f is *proper* if the domain of f, $\text{dom}(f) = \{x \mid f(x) < +\infty\}$ is nonempty.

If κ is a closed gauge, then its polar may also be expressed as [**Roc70**, Section 15]

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

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 [**Roc70**, Section 15]

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

(3) 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 [**BV04**, Chapter 5] for a comprehensive introduction to Lagrange duality, and [**Roc70**, Section 28] or [**BTN01**, Chapter 2] for a treatment of Lagrange duality theory.

The nonlinear pair (3.2.2) demonstrates that the gauge dual model GD-nonlin can be described simply using the polar function κ° and the antipolar set $\mathcal{C}'$. Similarly, the Lagrange dual model 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 [**Roc70**, 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 simple forms

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

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 [**BV04**, 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.

(4) In 1970, Rockafellar thoroughly analyzed gauge functions and their polars [**Roc70**, Part III]. The concept of gauge duality was then introduced by Freund in 1987 [**Fre87**]. This seminal work focuses 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 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 [**FMP14**], where the authors consider the *inequality-constrained* primal and gauge dual pair

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

As we will see in Section 3.4, the PLP-PLGD pair (3.2.1) pair is recovered when we set $\kappa(X)=||X||_1+\delta_{(\cdot)\succeq 0}(X)$ and $\rho(y)=||y||$. The authors of [**FMP14**] 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 gauge dual pair (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 [**ABD$^+$17**] 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. Theory for linearly and nonlinearly constrained models

(1) In this section we develop gauge duality theory for the inequality-constrained primal and gauge dual pair (3.2.21). We begin by establishing the propositions necessary to show the gauge duality of (3.2.21) is a consequence of the duality of both the nonlinear (3.2.2) and linear (3.2.20) models. This two-track proof strategy as depicted in Figure 3.1 leads to Theorem 3.3.1, (i) and (ii), respectively.

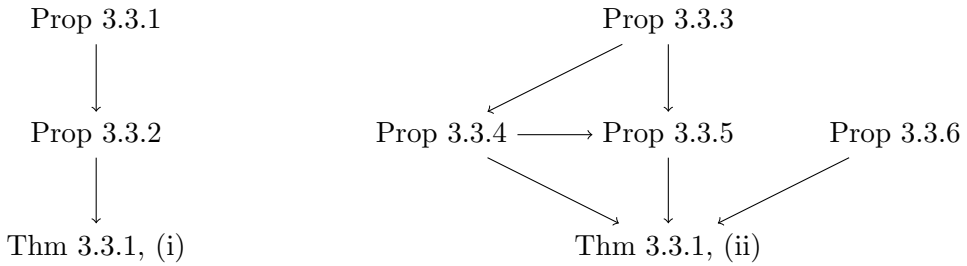

FIGURE 3.1. Dependency chart for proofs of Theorem 3.3.1, (i) and (ii).

Next we establish weak duality, strong duality, and optimality conditions for (3.2.21). In Section 3.4 we return to the PhaseLift model (3.2.1), where these optimality conditions provide a method for recovery of the primal signal x from a dual solution y.

The results in this section are based on [**Roc70**], [**Fre87**], and especially [**FMP14**], and rely on the analysis of polarity relations rather than perturbation analysis as discussed in [**ABD$^+$17**]. In particular, the two-track proof strategy depicted in Figure 3.1 was first developed in [**FMP14**] as part of a larger treatment of antipolar calculus, and refers to [**Roc70**] for more elementary results (e.g., Propositions 3.3.3, 3.3.4, and 3.3.6). This section provides a self-contained development of gauge duality for (3.2.21).

(2) The following two propositions are used in Theorem 3.3.1, (i) to show the inequality pair (3.2.21) is an instance of the nonlinear pair (3.2.2). Since (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 particular constraint set $\mathcal{C} = \{x \mid \rho(Ax - b) \leq \epsilon\}$. 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*

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

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

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

PROOF. The first result is proved in [**FMP14**, Proposition 3.3] and the second in [**FMP14**, 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 the following proposition.

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

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

□

(3) The next four propositions allow us to derive the inequality-constrained primal-gauge dual pair (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 (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. 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 $x_1 \in \text{dom } \kappa_1$ and $\langle x_2, z_2 \rangle \leq \kappa_2(x_2)$ for all $x_2 \in \text{dom } \kappa_2$. 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 $x_1 \in \text{dom } \kappa_1$ and $x_2 \in \text{dom } \kappa_2$, 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 such 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 $\rho(Ax - b) \leq \epsilon$ from (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

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

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 $\langle (x,\sigma),(y,\tau)\rangle \leq 1$ for all $(x,\sigma) \in \operatorname{epi}\rho$. Since ρ is a gauge, $(x,\sigma) \in \operatorname{epi}\rho$ is equivalent to $\rho(\alpha x) \leq \alpha\sigma$ for all $\alpha > 0$. Then $(\alpha x, \alpha\sigma) \in \operatorname{epi}\rho$ for all $\alpha > 0$. Dividing the inequality by α and taking the limit, we have

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

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 $\langle (x,\sigma),(y,\tau)\rangle \leq \mu\delta_{\operatorname{epi}\rho}(x,\sigma)$. Thus $(\delta_{\operatorname{epi}\rho})^{\circ}(y,\tau) = +\infty$.

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

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

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

(4) We are now prepared to prove the essential results which underly our phase retrieval method and signal recovery strategy. We begin with the following gauge duality result.

THEOREM 3.3.1. *Let P-ineq and GD-ineq be the inequality-constrained pair of models from (3.2.21). Also let* $\mathcal{C} = \{x \mid \rho(Ax - b) \leq \epsilon\}$ *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 $\phi(x, r, \sigma) = \kappa(x) + \delta_{\operatorname{epi}\rho}(r, \sigma)$. Since the epigraph of a gauge is closed under positive scaling, the indicator function $\delta_{\operatorname{epi}\rho}$ 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. □

(5) With the gauge duality of (3.2.21) established, we proceed by establishing weak duality, strong duality, and optimality conditions for this pair of models. These properties will play a central role in the first-order method for optimizing the PLGD model, 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 (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 in (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 (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 (3.2.21), we appeal to the strong duality properties of the Lagrange primal and dual pair (see [**Roc70**, Section 28]).

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

This proof strategy also requires that ρ° be continuous in order to identify an appropriate Lagrange dual feasible sequence which may be rescaled to a gauge dual feasible sequence.

PROPOSITION 3.3.8. (strong duality) *Assume the primal model P-ineq in (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, 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), [**Roc70**, 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, [**Roc70**, 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

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

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

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

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 [**Roc70**, 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).

□

(6) We close this section by establishing the optimality conditions of (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 in (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. Optimality conditions and primal recovery for the PLGD model

(1) This section establishes optimality conditions and a primal signal recovery strategy for the PLGD model (3.2.1), two essential tools for developing the method 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 inequality-constrained primal-dual model (3.2.21). Thus Proposition 3.3.9 applies to the PLGD model, supporting the development of 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 first-order method established in the next chapter.

(2) 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 $\mathcal{H}$ to $\mathbb{R}^m$ (see (2.3.6) 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

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

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

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

Then by Theorem 3.3.1, the GD-ineq model has the objective $\kappa^\circ(\mathcal{A}^* y)$ and constraint $\langle b, y\rangle - \epsilon||y|| \geq 1$. We may further simplify κ° by noting that $\kappa(X)$ is strictly positive and finite for all feasible $X \in \mathcal{H}$. 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.

(3) 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 $\mathcal{X} \times \mathcal{Y} = \mathcal{H} \times \mathbb{R}^m$. 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 admit a simultaneous ordered SVD [**Gri91**].

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

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

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

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

(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* $\mathcal{A}^*y$ *admit a simultaneous ordered eigendecomposition.*

PROOF. Since $\rho^\circ(\cdot) = ||\cdot||$ 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 model (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

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

Thus for all i, if $\lambda_i(X) > 0$ then we must have $\lambda_i(\mathcal{A}^*y) = \lambda_1(\mathcal{A}^*y)$ and the matrices X and $\mathcal{A}^*y$ admit a simultaneous ordered eigendecomposition. □

(4) In addition to Corollary 3.4.1, the following first-order optimality condition for convex optimization also 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 method. In the next chapter 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 $\mathcal{C} = \{y \in \mathbb{R}^m \mid \langle b, y \rangle - \epsilon||y|| \geq 1\}$.

The following result can be viewed as a fixed-point optimality condition, where y is optimal if $y = \Pi_{\mathcal{C}}(y - \alpha g)$ for some $g \in \partial f(y)$ 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 (1.2.5) 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 convex, subdifferentiable 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_0 is optimal for the minimization problem

$$\min_y F(y) := f(y) + \delta_{\mathcal{C}}(y) \tag{3.4.3}$$

if $y_0 = \Pi_{\mathcal{C}}(y_0 - \alpha g)$ for some nonzero $g \in \partial f(y_0)$ and all $\alpha > 0$.

PROOF. We begin by demonstrating the first-order optimality condition for unconstrained convex subdifferentiable functions. A variable $y_\star$ is optimal for (3.4.3) if and only if

$$F(y_\star) = F(y_\star) + 0^T(y - y_\star) \leq F(y) \quad \forall y \in \mathcal{Y}. \tag{3.4.4}$$

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

$$y_\star \text{ is optimal if and only if } 0 \in \partial F(y_\star). \tag{3.4.5}$$

Thus it suffices to show $0 \in \partial F(y_0)$. Note that $-g$ is not a descent direction for f, since $f(y_0) = f(\Pi_{\mathcal{C}}(y_0 - \alpha g))$ for all $\alpha > 0$. Then $-g$ is in the normal cone (1.2.6) of $\mathcal{C}$ at y_0. Additionally, the normal cone $N_{\mathcal{C}}(y_0)$ is equivalent to the subdifferential of the indicator function $\partial\delta_{\mathcal{C}}(y_0)$, since

$$\begin{aligned} \partial\delta_{\mathcal{C}}(y_0) &= \{g \mid \delta_{\mathcal{C}}(y) \geq \delta_{\mathcal{C}}(y_0) + \langle g, y - y_0\rangle \ \ \forall y\} \\ &= \{g \mid 0 \geq \langle g, y - y_0\rangle \ \ \forall y \in \mathcal{C}\} \\ &= N_{\mathcal{C}}(y_0). \end{aligned} \tag{3.4.6}$$

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 $\partial F(y_0) = \partial f(y_0) + \partial\delta_{\mathcal{C}}(y_0)$ [**Roc70**, Theorem 23.8]. Then $g \in \partial f(y_0)$ and $-g \in \partial\delta_{\mathcal{C}}(y_0)$ imply that $0 \in \partial F(y_0)$, and thus y_0 is optimal for (3.4.3).

□

Thus $y = \Pi_{\mathcal{C}}(y - \alpha g)$ is another optimality condition for the PLGD model (3.2.1).

(5) 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 $y \in \mathbb{R}^m$ be an arbitrary optimal solution for (3.2.1), $U \in \mathbb{C}^{n\times r}$ the eigenvectors corresponding to the largest algebraic eigenvalue λ_1 of $\mathcal{A}^*y$, and r the multiplicity of λ_1. Then $X \in \mathbb{C}^{n\times n}$ is a solution to (3.2.1) if and only if there exists an $r \times r$ matrix $S \succeq 0$ such that
(a) $X = USU^*$,
(b) $b - \mathcal{A}(X) \in \epsilon\partial||\cdot||(y)$.

PROOF. If X is a solution to (3.2.1), then (e) of Corollary 3.4.1 implies that rank of X is no larger than the multiplicity r of $\lambda_1(\mathcal{A}^*y)$, and (f) implies that X has the form $X = USU^*$ 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 show that the optimality conditions (a)-(d) of Corollary 3.4.1 also hold. The optimality of y gives $\langle b, y\rangle - \epsilon||y|| = 1$. The next two conditions may be derived from definitions of the dual norm $\rho^\circ(\cdot) = ||\cdot||$, 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

$$\langle b - \mathcal{A}(X), y\rangle = \epsilon||y||_*. \tag{3.4.7}$$

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

$$\langle b - \mathcal{A}(X), y\rangle = ||b - \mathcal{A}(X)|| \cdot ||y||_*. \tag{3.4.8}$$

The pair (3.4.7) and (3.4.8) give optimality conditions (a) and (c).

The following equality establishes optimality condition (d):

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

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 largest algebraic eigenvalue λ_1 of $\mathcal{A}^*y$. Since the diagonalization of $X = USU^*$ only requires a unitary transformation of U, this transformation will not affect the eigenspace of $\mathcal{A}^*y$. Thus X and $\mathcal{A}^*y$ admit a simultaneous ordered SVD. Finally, the third equality is a result of X having rank at most r and λ_1 having multiplicity r.

Thus optimality conditions (a)-(d) hold, and X is a solution to (3.2.1).

□

This primal recovery property allows us to develop a recovery strategy for obtaining a signal x from a dual variable y. If $y \neq 0$ then the 2-norm used in the PLP-PLGD pair (3.2.1) gives $\partial||\cdot||(y) = \nabla||\cdot||_2(y) = y/||y||$. For a given PLGD model, if the lifted true signal $X = \bar{x}\bar{x}^*$ is in the set of optimal matrices, then Corollary 3.4.3 indicates that the corresponding dual optimal variable $y_\star$ will have the relation

$$\eta = (\bar{b} + \eta) - \bar{b} = b - \mathcal{A}(X_\star) = \epsilon\nabla||\cdot||_2(y_\star) = \epsilon\frac{y_\star}{||y_\star||}. \tag{3.4.9}$$

Thus the noise term η will be in the set of rescaled optimal dual variables.

In the case of noisy phase retrieval (2.1.1), there is no guarantee that the lifted true signal $X = \bar{x}\bar{x}^*$ will be an optimal matrix for the PLGD model (3.2.1). In this case, the gauge dual variable y can be viewed instead 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 the optimal matrix 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 a first-order method for optimizing the PLGD model (3.2.1) and recovering a primal signal x from the PLGD solution y.

CHAPTER 4

A first-order method for the PLGD model

4.1. Introduction

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

The algorithm presented in this chapter was first established in [**FM16**]. 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 first-order method. Thus, due to the effectiveness of this particular method for noiseless phase retrieval and the existence of a comprehensive, specialized software package, we use this method to examine the behavior of first-order methods for optimizing (3.2.1). All implementation details in this chapter are identical to the original software package unless otherwise noted.

4.2. Algorithm and implementation details

(1) We begin by examining the features of the PLGD model (3.2.1) which are essential to developing a first-order descent method. Recall that first-order methods like gradient descent 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 with some policy, and $\Pi_{\mathcal{C}}$ is projection onto the feasible region $\mathcal{C}$. Since we are primarily concerned with large-scale signal recovery, the descent direction must be constructed using only first-order information. 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. Additionally, the PLGD model

(3.2.1) is a convex problem, having both convex objective function $\lambda_1(\mathcal{A}^*y)$ and convex constraint domain $\mathcal{C} = \{y \in \mathbb{R}^m \mid \langle b, y\rangle - \epsilon||y|| \geq 1\}$. Thus, any descent method 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. Some applicable methods include projected gradient, quasi-Newton, and spectral bundle methods. Our first-order method of choice for solving (3.2.1) is a projected gradient descent method with adaptive steplength based on the local differentiability of $\lambda_1(\mathcal{A}^*y)$.

(2) Construction of this projected gradient method requires the following sequence of computational steps. First, a descent direction is chosen from the gradient or subdifferential (1.2.11) of the objective function. Next, an initial steplength is computed and used to initialize a linesearch (backtracking) method for determining the update y_+. The linesearch and update both require a method for projecting onto the feasible region $\mathcal{C}$. Finally, a recovery method is used to recover the primal signal update x_+ from the dual update y_+.

The steps described above are sufficient for a projected gradient method. However, the authors of [**FM16**] found that two additional refinement steps tend to accelerate convergence greatly for this descent method. Thus our algorithm follows the primal signal recovery with a primal refinement step and a dual refinement step.

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

$$\partial\lambda_1(\mathcal{A}^*y) = \{\mathcal{A}(V_1 T V_1^*) \mid T \succeq 0,\ \mathrm{tr}(T) = 1\}, \tag{4.2.1}$$

where $V_1 \in \mathbb{C}^{n\times r_1}$ are the eigenvectors corresponding to the largest eigenvalue λ_1 of $\mathcal{A}^*y$ and r_1 is the multiplicity of λ_1 [**PB**$^+$**14**, 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

$$g = \nabla\lambda_1(\mathcal{A}^*y) = \mathcal{A}(v_1 v_1^*). \tag{4.2.2}$$

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}.

(4) 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 [**BB88**]

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

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 [**NW06**, 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 [**ZH04**]. 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.

(5) 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., [**Ber16**, Proposition 1.2.3 and Section 3.3.1]). A typical choice for this steplength sequence is $\alpha_k = \mathcal{O}(1/k)$.

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

expression

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

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

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

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 [**BV04**, Chapter 5]).

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

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

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 $\epsilon y/||y||$ to the noise term η as shown in (3.4.9). If $\lambda_1(\mathcal{A}^*y)$ is differentiable, then λ_1 is unique and (4.2.9) simplifies to the closed-form expression

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

(8) The previous steps constitute a complete 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.

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

unconstrained nonconvex problem

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

which is initialized with the primal recovery solution $\hat{x}$ from (4.2.10). This problem can be interpreted as a partially denoised version of the wflow least-squares problem (2.3.14), where b has been replaced by b_ϵ.

(10) 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 $X_\star = \bar{x}\bar{x}^*$ is rank-one, then Corollary 3.4.3 (a) indicates that the largest algebraic 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 $\bar{x}$, and Corollary 3.4.1 (d) (strong duality) gives this rescaling

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

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

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

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 [**Ber16**, Proposition 1.2.5], is guaranteed not to interfere with the convergence behavior of the underlying method.

(11) The above sequence of steps and methods leads to the following projected gradient descent method for optimizing the PLGD model (3.2.1):

Algorithm 3 Projected gradient descent method with refinement

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

Output: Approximate solution signal x.

while not converged **do**
  Compute largest algebraic eigenpairs: (λ_1, v_1) and (λ_2, v_2) from $\mathcal{A}^* y$.
  Compute (sub)gradient: $g = \mathcal{A}(v_1 v_1^*)$ based on (4.2.1).
  Determine differentiability of $\lambda_1(\mathcal{A}^* y)$ based on (4.2.3).
  if $\lambda_1(\mathcal{A}^* y)$ is differentiable **then**
    Linesearch: Perform linesearch (4.2.5) with initial step α (4.2.4) to obtain y_+.
  else
    Projected subgradient step: Set $y_+ = \Pi_{\mathcal{C}}(y - \alpha g)$ with α from (4.2.6).
  end if
  Primal recovery: Compute $\hat{x}$ based on (4.2.10).
  Primal refinement: Find x_+ as the solution to (4.2.11) initialized with $\hat{x}$ and y_+.
  if $\epsilon = 0$ **then**
    Dual refinement: Find $\widehat{y}$ based on (4.2.12).
    if $\lambda_1(\mathcal{A}^* \widehat{y}) < \lambda_1$, set $y_+ = \widehat{y}$
  end if
  Update: $x = x_+$, $y = y_+$.
end while
Return: x.

(12) We close this section by describing the implementation details of Algorithm 3. This algorithm was original established in [**FM16**] and implemented in the accompanying software package `low-rank-opt`[1] (with the identical clone [2]). All tests and experiments in this dissertation are available for reproduction in a branch of the original package[3]. This branch

[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`

includes a few minor changes to the original algorithm which are noncritical to the behavior of this algorithm (see `README.md` in the main directory for details).

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

$$\epsilon = ||\eta|| = ||b - \bar{b}||$$

of the model (2.1.1). Convergence of Algorithm 3 (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 entire algorithm, only primal feasibility and strong duality must be measured. Thus the algorithm terminates when both of the following conditions are met:

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

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

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 this algorithm, steps 5-9 are skipped. This strategy has the result of avoiding an additional eigenvalue computation (step 6) during initialization. Additionally, the 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 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 algorithm performs a projected subgradient step (step 8) with 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 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 $l_k = 1/k$.

(13) Computationally, most of the steps in Algorithm 3 are very simple. The three key exceptions are the eigenvalue computation (steps 2 and 6), 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), where only the first two largest algebraic 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 [**Sch05**]. 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 [**Sch08**], [**SBFM09**].

In each of these three subroutines, the primary computational cost comes from $\mathcal{A}$-products (2.3.7), 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 [**CLS15**] and [**FM16**].

(14) Theoretically, the steplength strategy in Algorithm 3 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) [**ZH04**], [**Ber16**, Proposition 1.2.3 and Section 3.3.1]. Yet the rate of convergence differs greatly for noiseless and noisy phase retrieval problems. This 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. Efficient recovery for noiseless phase retrieval

(1) This section examines the behavior of Algorithm 3 for noiseless phase retrieval problems. The efficiency of this algorithm for noiseless problems is well-established in [**FM16**, 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 (2.3.14) discussed in Section 2.3.3. As proved in [**SQW16**] and restated at the end of Section 2.3.3, 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 Algorithm 3 (i.e., x typically has a relative error $||\bar{x}\bar{x}^* - xx^*||_F/||\bar{x}||_2^2$ 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 Algorithm 3 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. 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 $||\bar{x}\bar{x}^* - x_0x_0^*||_F/||\bar{x}||_2^2$.

Table 4.1 demonstrates that primal refinement is necessary for Algorithm 3 to converge, and dual refinement is essential to the efficiency of this algorithm. With only primal refinement, this 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, this algorithm fails to converge as the dual problem (4.2.12) is initialized with an inaccurate signal x. And when Algorithm 3 is run without either refinement step, it again fails to converge.

However, when both the primal and dual refinement subroutines are used, Algorithm 3 rarely takes more than a couple iterations. Since steps 5-9 are skipped during the 0-th iterate, Algorithm 3 serves as an improvement to the wflow algorithm. Both algorithms set the initial signal as the eigenvector corresponding to the largest algebraic eigenvalue of $\mathcal{A}^*b$ (where Algorithm 3 first rescales b by projecting it onto $\mathcal{C}$). Yet Algorithm 3 uses better 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.

Algorithm 3 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 [**FM16**, Section 5.1.1, 5.1.3]. Note that the optimality of y is unnecessary if we simply want to minimize the noiseless problem $||\mathcal{A}(xx^*) - b||$.

To this end, Algorithm 3 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 Algorithm 3 typically converges in 0-1 iterations with about the same computational cost as the wflow algorithm.

This dissertation addresses the challenges noisy phase retrieval poses for Algorithm 3 with the following contributions.

Chapter 5 addresses the convergence challenges Algorithm 3 experiences for noisy problems. We begin by demonstrating that this algorithm stagnates prior to convergence for noisy problems and determining the cause of this stagnation. Although Algorithm 3 does not generally converge for noisy phase retrieval problems, we demonstrate that this algorithm is more robust and accurate than the wflow method 2 for noisy problems. Section 5.4 establishes new termination conditions which indicate stagnation, so we may treat Algorithm 3 as a black-box and focus our attention on handling the EMEP.

Chapters 6 and 7 examine the computational challenges Algorithm 3 experiences for noisy problems. We identify the EMEP as the dominant computational cost and develop new methods for handling this problem. Chapters 6 provides a detailed explanation of three common methods for large-scale eigenvalue problems. The unique structure and properties of each method inform our strategy for handling the EMEP. Chapter 7 then examines the evolving spectral structure of the matrices $\mathcal{A}^* y_k$ across sequences of iterates k to develop an optimal method for handling the EMEP.

Chapter 8 presents a revised version of Algorithm 3 with an optimized method for handling the EMEP and new termination conditions for identifying stagnation.

CHAPTER 5

New termination conditions for the first-order PLGD algorithm for noisy phase retrieval

5.1. Introduction

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

We begin in Section 5.2 by describing how we construct experimental noisy PLGD models and discussing potential residuals for measuring the progress of Algorithm 3. Section 5.3 then examines the tendency of Algorithm 3 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 methods like Algorithm 3 from converging. To handle this issue, Section 5.4 establishes new termination conditions based on heuristic evidence indicating Algorithm 3 has stopped making signal recovery progress. These new conditions allow us to treat Algorithm 3 as a black-box solver in the remainder of this dissertation so we may develop optimal methods for handling the EMEP, the computational bottleneck of this algorithm.

5.2. Experimental models and potential residuals

(1) This section describes two methods for creating experimental noisy phase retrieval problems and presents a set of potential residuals to measure the progress of 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 Algorithm 3. The set of potential residuals is a combination of those discussed in Chapter 4 and new residuals based on the variables available in Algorithm 3.

(2) We begin by describing experimental noisy phase retrieval problems which have *Gaussian noise*. Recall that the noisy phase retrieval problem (2.1.1) involves an observation $b = \bar{b}+\eta$, where the true observation $\bar{b}$ is contaminated by some nontrivial noise η. To mimic the typical experimental noisy phase retrieval scenario, we begin with an unknown signal $\bar{x}$. A sensing operator $\mathcal{A}$ (2.3.7) is then chosen with diagonal mask matrices C_j whose diagonal elements have complex standard Gaussian distribution (1.2.16) (see Section 2.2 for an explanation of masks). The sensing operator is then used to create the true observation $\bar{b} = \mathcal{A}(\bar{x}\bar{x}^*)$, and a noise term η is chosen with real standard Gaussian distribution (1.2.15). Finally, the noisy observation is set as $b = \bar{b} + \eta$. Altogether, given a signal $\bar{x}$, a sensing operator $\mathcal{A}$ (2.3.7), and noise ratio ϵ_{rel}, we create the phase retrieval problem with Gaussian noise experimentally with the steps

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

where η is rescaled in the third step to satisfy the noise ratio $\epsilon_{\text{rel}} = ||\eta||/||b||$. 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 $b = \bar{b}+\eta$ 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)

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

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

(3) To demonstrate how the differentiability of the dual objective $\lambda_1(\mathcal{A}^* y)$ impacts the behavior of Algorithm 3, 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 (1.2.15) 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 $\bar{x}$ is set as the eigenvector corresponding to the largest algebraic eigenvalue of $\mathcal{A}^* y_\star$ per Corollary 3.4.3 (a). This signal is then used to create a true observation $\bar{b}$ which is noised with a rescaling of $y_\star$ per Corollary 3.4.3 (b) and (3.4.9). Altogether, given a noise ratio ϵ_{rel} we have the following steps for constructing the phase retrieval problem with synthetic noise

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

where $\epsilon = \epsilon_{\text{rel}}||b||$. Steps one and two in (5.2.4) guarantee the PLP-PLGD optimality conditions (Corollary 3.4.1) will hold for a known pair ($X_\star = \bar{x}\bar{x}^*$, $y_\star$) with rank-one optimal matrix $X_\star$. Additionally, step four in (5.2.4) satisfies the primal recovery equation (3.4.9), guaranteeing the primal refinement strategy (4.2.11) used in Algorithm 3 will recovery the true signal $\bar{x}$ 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 $b = \bar{b} + \epsilon \frac{y_\star}{||y_\star||}$ 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)

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

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

(4) Next we establish an exhaustive list of residuals which are available for measuring the progress of Algorithm 3. 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 Algorithm 3 maintains dual feasibility (each gradient descent step projects the dual iterate y onto the feasible set).

(5.2.7a)	signal relative error	$\|\|xx^* - \bar{x}\bar{x}^*\|\|_F / \|\|\bar{x}\bar{x}^*\|\|_F$
(5.2.7b)	dual relative error	$\|\|y - y_\star\|\| / \|\|y_\star\|\|$
(5.2.7c)	primal true relative error	$\|\|\mathcal{A}(xx^*) - \bar{b}\|\| / \|\|\bar{b}\|\|$
(5.2.7d)	primal relative error	$\rho := \|\|\mathcal{A}(xx^*) - b\|\| / \|\|b\|\|$
(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	$\|\|y - \hat{y}\|\| / \|\|y\|\|$
(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 Algorithm 3 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 Algorithm 3 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 Algorithm 3, and thus used to establish new termination conditions in that section.

5.3. Stagnation of the first-order PLGD algorithm

(1) In this section we demonstrate the tendency of 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 largest algebraic eigenvalue of the optimal dual matrix $\mathcal{A}^* y_\star$ will also be greater than one, causing the dual objective $\lambda_1(\mathcal{A}^* y)$ to be nondifferentiable in a neighborhood of $y_\star$. Additionally, since $X_\star \neq \bar{x}\bar{x}^*$, the primal refinement strategy (4.2.11) used in Algorithm 3 is not guaranteed to recover the true signal $\bar{x}$.

In order to describe the behavior of Algorithm 3 for noisy phase retrieval problems, we use the following terminology to make the distinction between Algorithm 3 converging to an optimal solution and converging to a nonoptimal variable. Recall that Algorithm 3 *converges* at a given iterate if the conditions (4.2.13) and (4.2.14) are satisfied. We say that Algorithm 3 *fails to converge* if conditions (4.2.13) or (4.2.14) are not satisfied for any iterate within a maximum number of iterations (typically 1,000). In contrast with converging, we say Algorithm 3 *stagnates* at a given iterate if the progress from the previous iterate is trivial. Specifically, Algorithm 3 stagnates at a given iterate if a chosen subset of difference residuals (5.2.7e, g-j) are below a required set of tolerances.

(2) The authors of [**FM16**] use PLGD models with synthetic noise (5.2.6) to demonstrate that Algorithm 3 is able to converge and accurately recover the true signal $\bar{x}$ when the optimal matrix in the PLGD model is rank-one (i.e., $X_\star = \bar{x}\bar{x}^*$). Table 5.1 depicts the results of

Algorithm 3 applied to the PLGD model with synthetic noise (5.2.6), corroborating the results in [**FM16**, Section 5.1.2].

	$L = 5$		$L = 10$		$L = 15$	
	Iterations	xErr	Iterations	xErr	Iterations	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 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 wflow (Algorithm 2) for the same problem.

Table 5.1 demonstrates that Algorithm 3 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 Algorithm 3 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 Algorithm 3 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 Algorithm 3 applied to PLGD models with Gaussian noise (5.2.3). Algorithm 3 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, Algorithm 3 fails to converge for PLGD models with Gaussian noise (5.2.3).

The next experiment demonstrates that Algorithm 3 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` [**TTT99**] 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.

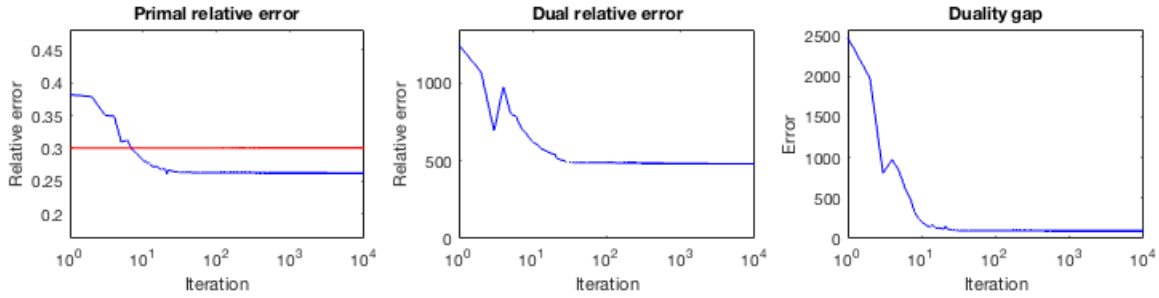

FIGURE 5.1. Primal relative error (5.2.7d), dual relative error (5.2.7b), and duality gap (5.2.7f) for 10,000 iterates of Algorithm 3 applied to a natural noise model with $n = 16$, $L = 6$ observations, and noise ratio 0.30. The horizontal axis is log-plotted to highlight early progress along with later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

Figure 5.1 demonstrates that Algorithm 3 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 Algorithm 3 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 Algorithm 3. 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 Algorithm 3.

		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
$\epsilon_{\text{rel}} = 0.05$	Algo. 3 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
$\epsilon_{\text{rel}} = 0.15$	Algo. 3 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
$\epsilon_{\text{rel}} = 0.30$	Algo. 3 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. 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 Algorithm 3 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 Algorithm 3 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 Algorithm 3 to stagnate. Algorithm 3 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 largest algebraic 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 Algorithm 3 will stagnate prior to approaching $y_\star$. As a result, this 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 Algorithm 3 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 largest algebraic eigenvalue, making the dual objective function $\lambda_1(\mathcal{A}^* y)$ differentiable at $y_\star$ and allowing Algorithm 3 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 (3.4.9) successfully denoises the noisy observation $b = \bar{b} + \epsilon y_\star / ||y_\star||$ because the synthetic noise steps (5.2.4) guarantee an exact relation $\eta = \epsilon y_\star / ||y_\star||$ between the noise term η and optimal dual variable $y_\star$. Thus Algorithm 3 is able to return a matrix X which accurately approximates the optimal matrix $X_\star = \bar{x}\bar{x}^*$.

(3) 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 Algorithm 3 with and without dual refinement for three PLGD models with Gaussian noise (5.2.3).

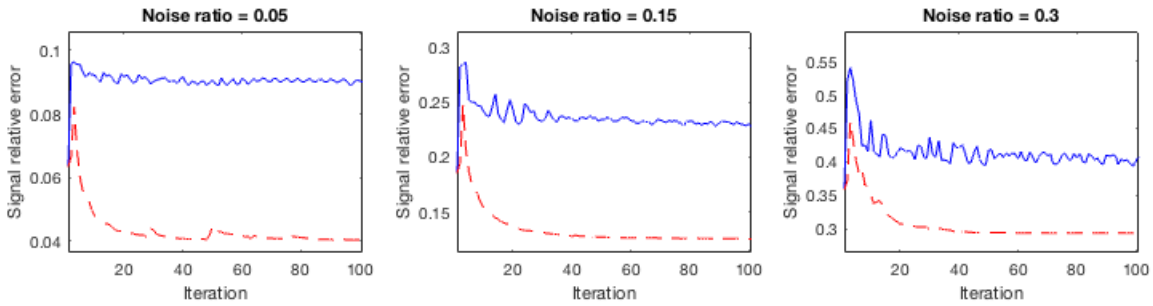

FIGURE 5.2. A comparison of Algorithm 3 with and without dual refinement (4.2.12) for PLGD models with Gaussian noise (5.2.3) with various noise levels, with random Gaussian signal of size $n = 128$ and $L = 10$ observations. The solid blue line indicates dual refinement, and the dashed red line indicates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

Figure 5.2 demonstrates that the dual refinement step (4.2.12) inhibits the progress otherwise made by Algorithm 3. 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 Algorithm 3 is not beneficial for signal recovery in phase retrieval problems with Gaussian noise (5.2.2).

5.4. New termination conditions

(1) In Section 5.3, we saw that 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 [**FM16**]. This section establishes new termination conditions for Algorithm 3 for PLGD models with Gaussian noise (5.2.3) and demonstrates the effectiveness of these conditions for identifying stagnation of Algorithm 3. (For a comparison of the unused residuals from (5.2.7), see Appendix II, Chapter B.)

(2) We begin by presenting the new termination conditions for Algorithm 3 for PLGD models with Gaussian noise (5.2.3). To identify stagnation of Algorithm 3 and declare termination, the primal difference (5.2.7e) is set to

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

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

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

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.

(3) To demonstrate the effectiveness of these termination conditions, we consider a set of six PLGD models with Gaussian noise (5.2.3) solved with Algorithm 3. The signal relative error (5.2.7a) serves as a control measurement to identify when Algorithm 3 has stagnated

and should terminate. Figure 5.3 depicts the signal relative error (5.2.7a) for each of these models.

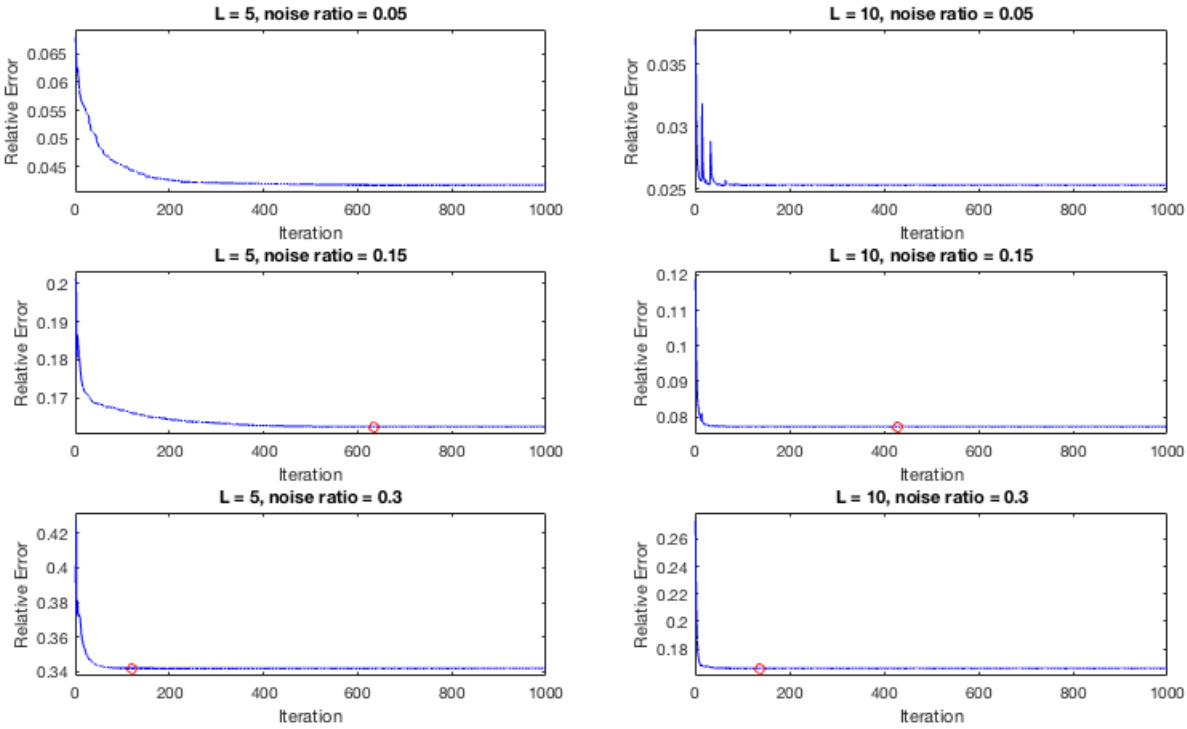

FIGURE 5.3. Signal relative errors (5.2.7a) for six PLGD models with Gaussian noise (5.2.3) solved with Algorithm 3. The image in Figure 2.2 was resized to 64×64 pixels, made grayscale, and six models were generated with oversampling rates of $L = 5, 10$ and noise ratios $\epsilon_{\text{rel}} = 0.05, 0.15, 0.30$. These models were then solved using Algorithm 3 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 Algorithm 3 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 Algorithm 3 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 Algorithm 3

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 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 Algorithm 3 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 Algorithm 3 appears to stagnate for models from Figure 5.3.

(4) In order to demonstrate that the primal difference (5.2.7e) and dual variable difference (5.2.7i) are indicators of the stagnation of Algorithm 3, 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 Algorithm 3 would satisfy this tolerance.

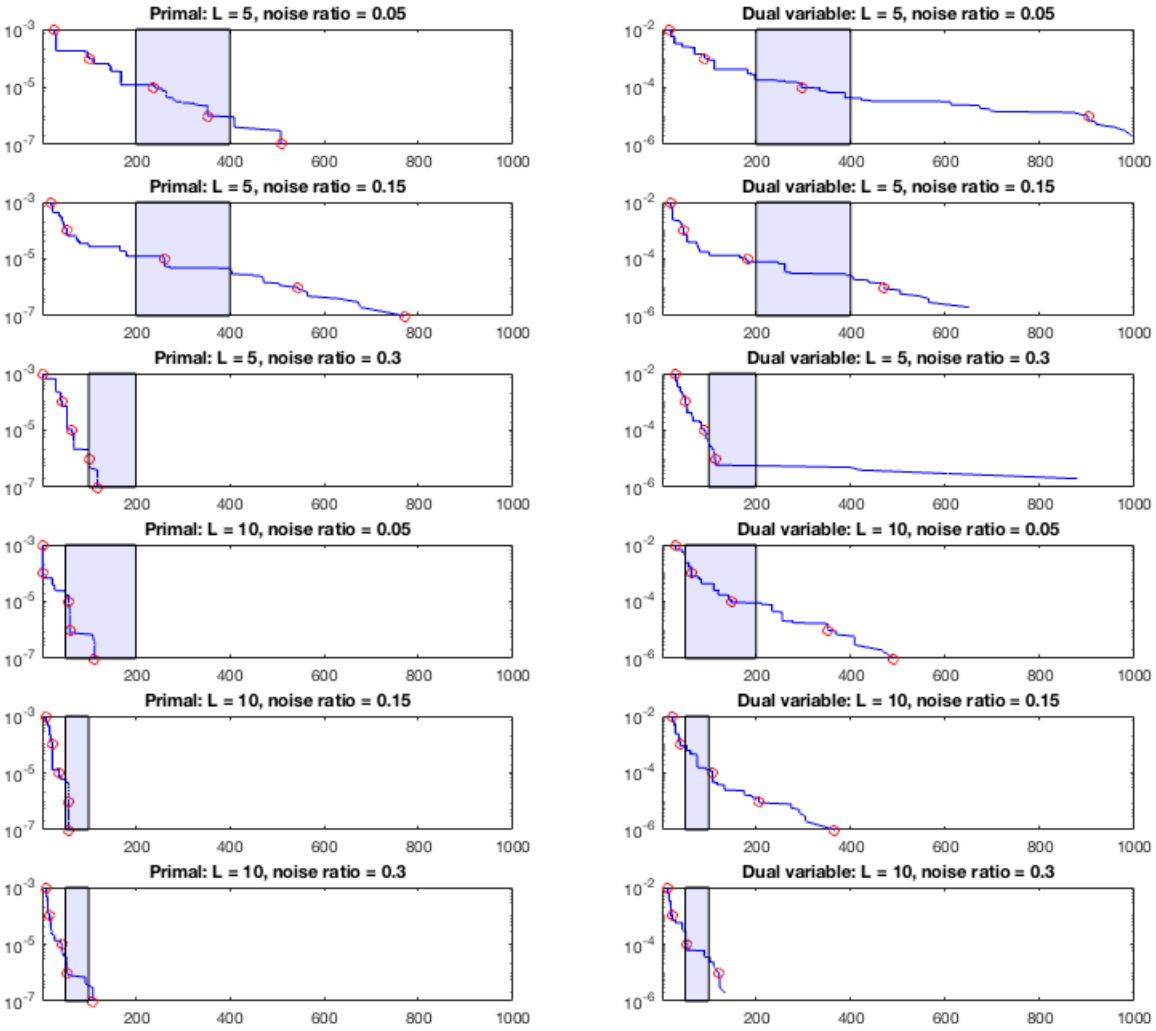

FIGURE 5.4. Plots of tolerance values against the iterate at which Algorithm 3 first satisfies this tolerance for the models discussed in Figure 5.3. Tolerances depicted are difference values for the primal objective (5.2.7e) and dual variable (5.2.7i). Red circles are placed at tolerances 10^{-n}. The 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 $L = 5$, Algorithm 3 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, Algorithm 3 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 Algorithm 3 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

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

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

(5) One additional concern for termination conditions is the nonmonotonic nature of Algorithm 3. In Figure 5.3, the model with $L = 10$ and $\epsilon_{\text{rel}} = 0.05$ demonstrates that Algorithm 3 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.

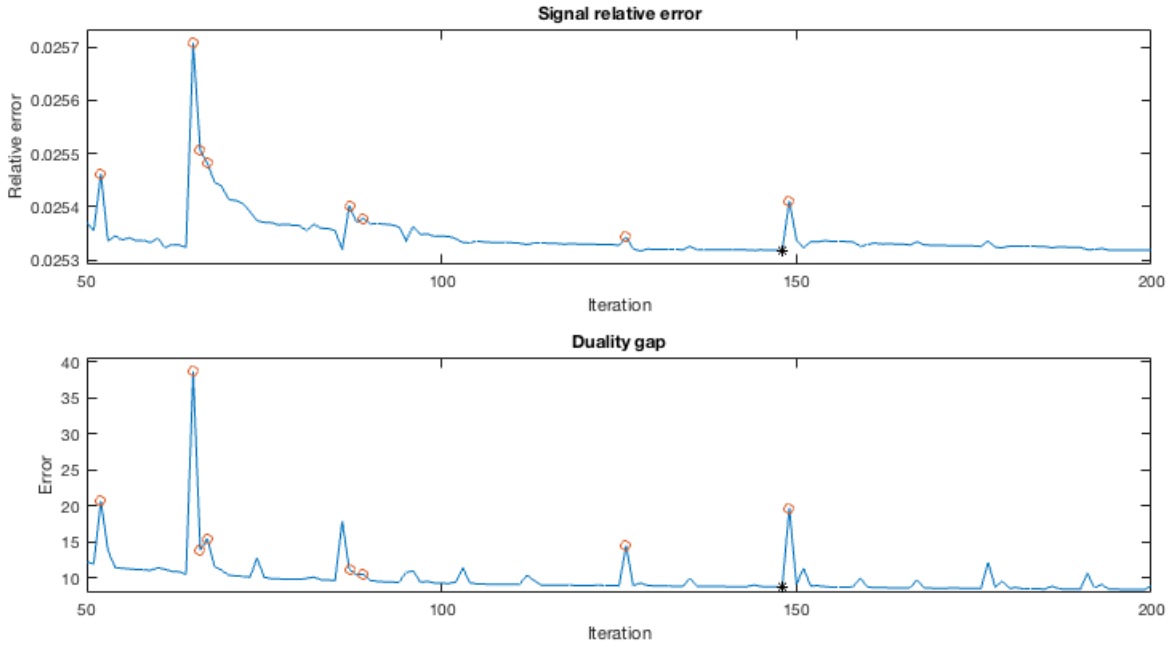

FIGURE 5.5. Signal relative error (5.2.7a) and duality gap (5.2.7f) for the model from Figure 5.3 with oversampling $L = 10$ and noise ratio $\epsilon_{\text{rel}} = 0.05$. Red circles indicate signals with relative error at least 0.05% above the mean of their ten neighbors, and the black asterisk indicates 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 Algorithm 3 terminates, we select the signal among the last 20 with the lowest duality gap value.

(6) 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 Algorithm 3 would terminate for each PLGD model with Gaussian noise from Figure 5.3.

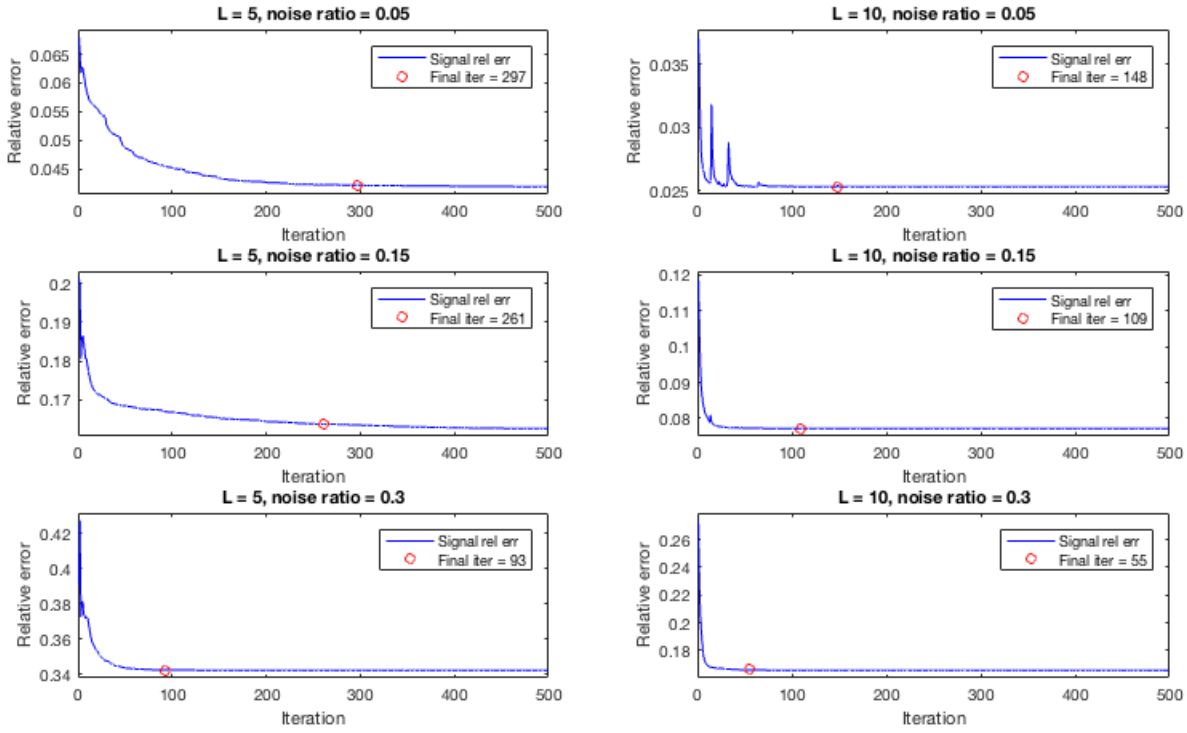

FIGURE 5.6. Iterate at which Algorithm 3 terminates for the models from Figure 5.3 based on new termination conditions.

For each model in Figure 5.6, Algorithm 3 successfully terminates within or nearby the interval of stagnation established in Table 5.5.

CHAPTER 6

Two methods for the evolving matrix eigenvalue problem

6.1. Introduction

In this chapter we examine the *evolving matrix eigenvalue problem* in Algorithm 3, which we define as the *EMEP*, and develop two common eigenvalue methods for handling large-scale eigenvalue problems like the EMEP. The eigenvalue methods presented in this chapter will allow us to develop new, efficient methods for handling the EMEP in Chapter 7.

Section 6.2 defines the EMEP and examines its evolving spectral structure across matrix iterates A_k. In particular, we observe that the largest algebraic eigenvalues tend to cluster as the algorithm proceeds, leading to more difficult eigenvalue problems in later matrix iterates. To handle these difficult eigenvalue problems, we develop two common eigenvalue methods: the *implicitly restarted Arnoldi method* (IRAM) [**Sor92**] in Section 6.3, and the *inverse-free preconditioned Krylov subspace method* (EIGIFP) [**GY02**] in Section 6.4.

The IRAM and EIGIFP are based on elementary eigenvalue methods which have inherent strengths and weaknesses related to the spectral structure of the given eigenvalue problem. To understand and exploit the theoretical or empirical convergence behavior of the IRAM and EIGIFP, we develop these methods systematically from their elementary eigenvalue methods.

6.2. The EMEP: computational costs and spectral properties

(1) In this section we define the *evolving matrix eigenvalue problem* (EMEP) and identify the EMEP in Algorithm 3 as the most computationally expensive subroutine in Algorithm 3. We see that this EMEP evolves computationally and structurally from early to later iterates in Algorithm 3. The high computational cost and changing structure of this EMEP leads us to explore various eigenvalue methods in the remainder of this chapter.

Generally speaking, an EMEP is a sequence of eigenvalue problems in which each eigenvalue problem is dependent on the results of the previous problems. For each iterate

k of the EMEP, we have a Hermitian matrix iterate $A_k \in \mathcal{H}$ and find its j largest algebraic eigenvalues $\Lambda_j^{(k)}$ and the matrix of corresponding eigenvectors $V_j^{(k)}$. Next, the previous set of basis matrices $\mathcal{V}_k = \{V_j^{(i)}\}_{i=0}^{k}$ are used to find the next matrix iterate A_{k+1}. To define the EMEP formally, let $A(\cdot)$ be a function which maps $\mathcal{V}_k$ to $\mathcal{H}$ for all finite $k \geq 0$ and let $V_0 \in \mathbb{C}^{n \times j}$ be an initial basis matrix. Then the EMEP is defined as

$$\begin{aligned} &\text{for} && k = 1, 2, \ldots, K \\ &\text{find} && \left(\Lambda_j^{(k)},\ V_j^{(k)}\right) \text{ of } A_k \\ &\text{s.t.} && A_k = A(\mathcal{V}_{k-1}) \\ & && \mathcal{V}_{k-1} = \left\{V_j^{(i)}\right\}_{i=0}^{k-1} \end{aligned} \tag{6.2.1}$$

where $\Lambda^{(k)}$ are the j largest algebraic eigenvalues of A_k and V_k is the matrix of corresponding eigenvectors.

xxx CITE 2 or 3 EMEP examples

To see that Algorithm 3 is an EMEP, first note that each eigenvalue problem in Algorithm 3 (steps 2, 6, and 14) requires the $j = 2$ largest algebraic eigenvalues of the matrix iterate $A_k = \mathcal{A}^* y_k$. (Also note that the EMEP iterate k does not correspond to the Algorithm 3 iterate since step 6 involves a linesearch which may involve more than one eigenvalue problem.) We will show that each matrix iterate $A_k = \mathcal{A}^* y_k$ in Algorithm 3 is computed using a variable y_k which is dependent on the previous set of basis vectors $\mathcal{V}_{k-1} = \{v_1^{(i)}\}_{i=0}^{k-1}$. It can be show inductively that y_k is a function of $\mathcal{V}_{k-1}$. The initial EMEP iterate $k = 0$ corresponds to the first eigenvalue problem in Algorithm 3, where $y_0 = \Pi_{\mathcal{C}}(b)$ is initialized using the observation vector b and is independent of any eigenvector. Thus we initialize $v_1^{(0)}$ as the empty set. If $k > 0$ then the update y_k is computed as $y_k = \Pi_{\mathcal{C}}(y_{k-1} - \alpha_{k-1} g_{k-1})$, where $g_{k-1} = \mathcal{A}(v_{k-1} v_{k-1}^*)$ is a function of v_{k-1} and α_{k-1} is determined using a linesearch on the minimization problem

$$\min_{\alpha} \quad \lambda_1 \left(\mathcal{A}^*(\Pi_{\mathcal{C}}(y_{k-1} - \alpha g_{k-1}))\right). \tag{6.2.2}$$

Thus y_k is dependent on v_{k-1} and y_{k-1} and Algorithm 3 is an EMEP.

The *Algorithm 3 EMEP* is defined as

$$\begin{aligned} &\text{for} \quad k = 1, 2, \ldots, K \\ &\text{find} \quad \left(\lambda_1^{(k)},\ v_1^{(k)}\right) \text{ and } \left(\lambda_2^{(k)},\ v_2^{(k)}\right) \text{ of } A_k \\ &\text{s.t.} \quad A_k = \mathcal{A}^* y_k \end{aligned} \tag{6.2.3}$$

where $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$ are the two largest algebraic eigenvalues of the *matrix iterate* A_k, and y_k is the previous dual variable generated by Algorithm 3 (from either step 2, 6 or 14).

(2) We many now examine the computational costs of the Algorithm 3 EMEP (6.2.3) for PLGD models with Gaussian noise (5.2.3).

As discussed in Section 4.2, the main computational costs in Algorithm 3 are the EMEP, the primal refinement (step 11), and the dual refinement (step 13).

The eigenvalue computation for the EMEP is performed using the MATLAB function `eigs` (described in Section 6.3 for details), and the primal refinement step involves solving (4.2.11) using `minFunc`, a quasi-Newton solver for unstrained optimization, with the descent direction determined using the limited-memory (l-)BFGS method for Hessian approximation [**Sch05**].

Since we are focused on PLGD models with Gaussian noise 5.2.3, the dual refinement step of Algorithm 3 is skipped (see the end of Section 5.3 for an explanation).

In both the Algorithm 3 EMEP (6.2.3) and the primal refinement step, the primary computational cost comes from $\mathcal{A}$-products (2.3.7), 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 [**CLS15**] and [**FM16**].

Also note that the computation of the eigenpair (λ_1, v_1) must be very accurate in order to determine an accurate descent step $g = \mathcal{A}(v_1 v_1^*)$ in Algorithm 3.

Table 6.1 depicts the computational costs of Algorithm 3 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. Algorithm 3 runtime and number of DFTs (with percentage of the total in parentheses) for the Algorithm 3 EMEP (6.2.3), primal refinement (solving (4.2.11) in step 11) and all other operations. Here n is signal size (i.e., number of pixels in the image from Figure 2.2), L is number of observations, and ϵ_{rel} is the noise ratio.

The results in Table 6.1 demonstrate the essential computational challenges of Algorithm 3. 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 non-trivial amount of computation. All other operations accounted for 0.00% of the overall runtime.

(3) Figure 6.1 depicts the computational costs the EMEP (6.2.3) for each of the six smaller models from Table 6.1. For each model, the computational cost of the k-th EMEP iterate is measured by the number of matrix-vector products $[\mathcal{A}^* y_k]x$.

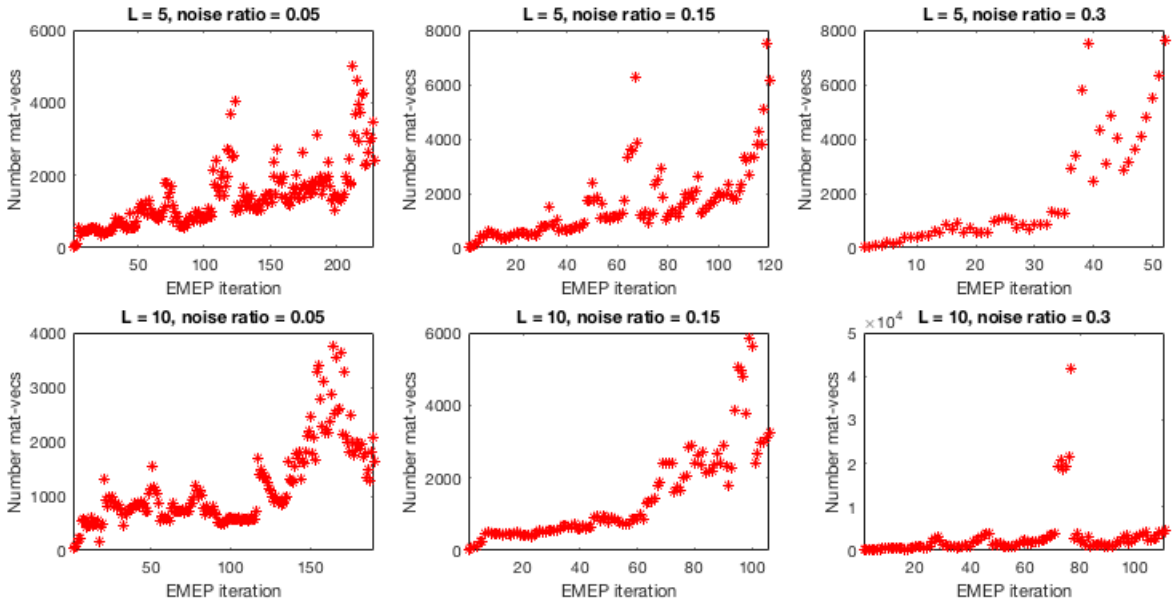

FIGURE 6.1. Number of matrix-vector products for each iteration in the EMEP (6.2.3)for the six smaller models from Table 6.1 .

Figure 6.1 demonstrates that the computational cost of the EMEP (6.2.3) varies greatly from earlier to later iterates. In each model, the later iterates account for the majority of the computational cost of the EMEP (6.2.3).

(4) To understand why the computational cost increases as the EMEP (6.2.3) progresses, we must examine the evolving structure of these eigenvalue problems. In Sections 6.3 and 6.4 we will discuss the impact the spectrum of a matrix has on the (theoretic or empirically expected) convergence rate of various eigenvalue methods. For now, we will briefly examine how the spectrum varies for the matrix iterates in the EMEP (6.2.3). Figure 6.2 depicts the spectrum of earlier and later iterates for a particular model from Table 6.1.

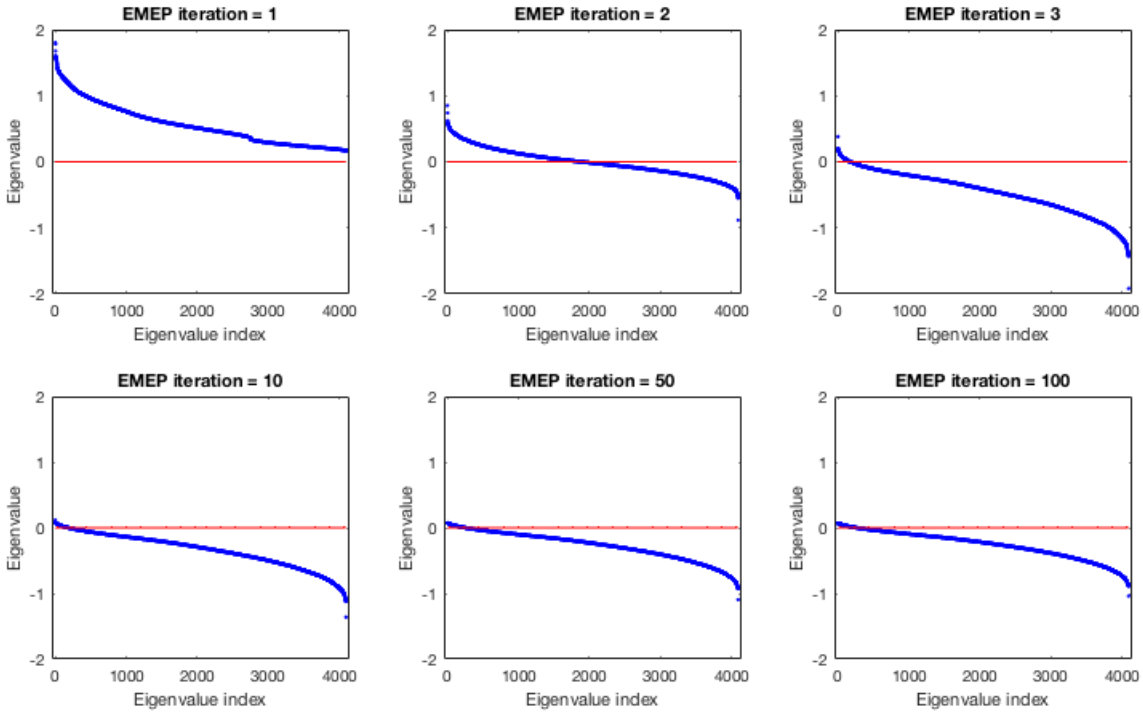

FIGURE 6.2. Spectrum of specific EMEP (6.2.3) matrix iterates A_k for the model from Table 6.1 with signal size $n = 4,096$, oversampling $L = 5$, and noise ratio $\epsilon_{\text{rel}} = 0.15$.

As we see in Figure 6.2, the spectrum of the matrix iterates A_k in the EMEP (6.2.3) 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 (3.2.1). The first matrix iterate $A_1 = \mathcal{A}^* b$ will always be positive-semidefinite because the observations b_i 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 Algorithm 3 minimizes the objective function $\lambda_1(\mathcal{A}^* y_k)$, the largest algebraic eigenvalue $\lambda_1^{(k)}$ of $\mathcal{A}^* y_k$ can be expected to decrease for later iterates k.

As we will see in Sections 6.3 and 6.4, the convergence rate of eigenvalue methods often depends on the distance between the desired eigenvalue λ_j and the next largest eigenvalue

λ_{j+1}. Figure 6.3 depicts the 20 largest algebraic eigenvalues of the EMEP (6.2.3) iterates from Figure 6.2.

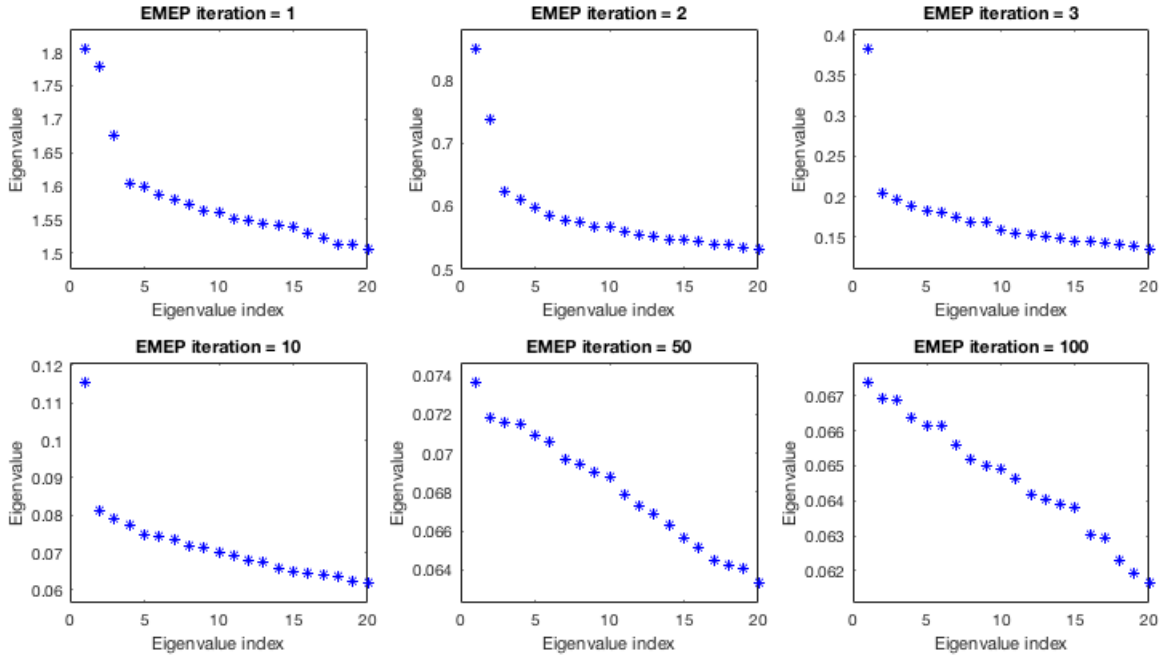

FIGURE 6.3. Twenty largest algebraic eigenvalues of specific EMEP (6.2.3) matrix iterates A_k for the model from Table 6.1 with signal size $n = 4,096$, oversampling $L = 5$, and noise ratio $\epsilon_{\text{rel}} = 0.15$.

Figure 6.3 demonstrates that the largest algebraic eigenvalues of the matrix iterates A_k cluster together as the EMEP (6.2.3) progresses. In general, this clustering can be expected. Section 4.2 established that the PLGD model (3.2.1) objective function $\lambda_1(\mathcal{A}^*y)$ is nondifferentiable if the two largest algebraic eigenvalues of $\mathcal{A}^*y$ are equal. And Section 5.3 demonstrated that PLGD models with Gaussian noise (5.2.3) typically have nondifferentiable optimal objectives $\lambda_1(\mathcal{A}^*y_\star)$. Thus the two largest algebraic eigenvalues $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$ of the EMEP (6.2.3) can be expected to have a decreasing relative difference

$$\frac{\lambda_1^{(k)} - \lambda_2^{(k)}}{\lambda_1^{(k)}}.$$

(5) Given the high computational cost of the EMEP (6.2.3) and the evolving structure of this problem, we now proceed to develop a few common methods for handling eigenvalue

problems. The convergence behavior and the strengths and weaknesses of these methods will help us develop more efficient ways of handling the EMEP (6.2.3) in Chapter 7.

6.3. The implicitly restarted Arnoldi method

(1) In this section we develop the *implicitly restarted Arnoldi method* (IRAM), a common large-scale eigenvalue method which is the default method for handling the EMEP (6.2.3). First proposed by Sorensen [**Sor92**], [**Sor97**], 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 following order.

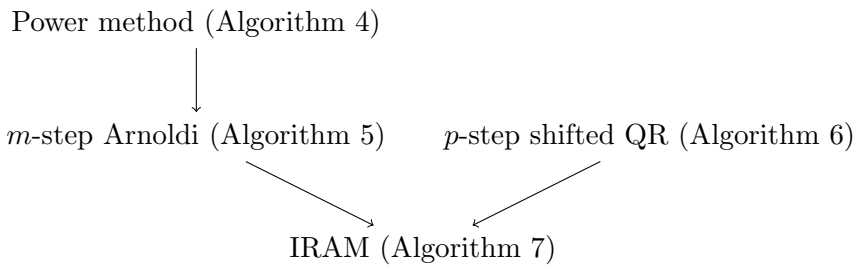

FIGURE 6.4. Dependency chart for the IRAM.

This section follows the treatment found in [**GVL12**, Chapters 8, 10], with occasional minor changes in notation.

(2) 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

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

will have v_1 as its limit. Formally, the power method is the following algorithm [**GVL12**, 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.

1: *Initialize:* $q^{(0)} = q^{(0)}/||q^{(0)}||$, $\rho^{(0)} = [q^{(0)}]^* A q^{(0)}$, $r^{(0)} = Au^{(0)} - \rho^{(0)} q^{(0)}$, $i = 1$.

2: **while** *not converged:* $||r^{(i)}||/(||Aq^{(i)}|| + |\rho^{(i)}|) > \text{tol}_{\text{rel}}$ **do**

3: $\quad z^{(i)} = Aq^{(i-1)}$

4: $\quad q^{(i)} = z^{(i)}/||z^{(i)}||$

5: $\quad \rho^{(i)} = [q^{(i)}]^* z^{(i)}$

6: $\quad r^{(i)} = Aq^{(i)} - \rho^{(i)} q^{(i)}$, $i = i + 1$

7: **end while**

8: *Return:* $(\lambda, v) = (\rho^{(i-1)}, q^{(i-1)})$.

The simplicity of power method allows for elegant, insightful convergence results like the following theorem, 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*

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

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 [**GVL12**, 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.

(3) 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 (1.2.9) (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 [**GVL12**, 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:* $q_1 = q_1/||q_1||, \quad z = Aq_1, \quad \alpha_1 = q_1^* z, \quad r_1 = z - \alpha_1 q_1, \quad Q_1 = [q_1], \quad H_1 = [\alpha_1]$.

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

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

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 [**GVL12**, Algorithm 10.1.1]. In this case, we have the following theorem. Note that this theorem involves *Chebyshev polynomials* [**Saa11**, 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 largest algebraic 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 [**GVL12**, Theorem 10.1.2]. □

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

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 [**GVL12**, 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 [**GVL12**, 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 largest algebraic 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 (6.3.3) 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 (6.2.3), 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.

(4) 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 [**Sor92**] and discussed in [**GVL12**, Sections 10.5.2-3]. To develop this algorithm, assume we are seeking the j 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 [**GVL12**, 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.

The following proposition 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. First we must show that

$$VR = (H_m - \mu_1 I) \cdots (H_m - \mu_p I). \tag{6.3.13}$$

Using induction, we see that if $p = 1$ then clearly (6.3.13) holds. If $p > 1$, then let $\tilde{V} = V_1 \cdots V_{p-1}$ and $\tilde{R} = R_{p-1} \cdots R_1$, and note that $H^{(p-1)} = \tilde{V}^* H_m \tilde{V}$. Then we have

$$\begin{aligned} VR &= \tilde{V}(V_p R_p)\tilde{R} = \tilde{V}\left(H^{(p-1)} - \mu_p I\right)\tilde{R} = \tilde{V}\left(\tilde{V}^* H_m \tilde{V} - \mu_p I\right)\tilde{R} \\ &= (H_m - \mu_p I)\,\tilde{V}\tilde{R} = (H_m - \mu_p I)(H_m - \mu_1 I)\cdots(H_m - \mu_{p-1} I) \\ &= (H_m - \mu_1 I)\cdots(H_m - \mu_p I), \end{aligned}$$

and (6.3.13) holds.

Next, note that if the m-step Arnoldi decomposition (6.3.4) is right-multiplied by e_1, then for all μ we have

$$(A - \mu I)\, Q_m e_1 = Q_m\, (H_m - \mu I)\, e_1. \tag{6.3.14}$$

Then we have

$$\begin{aligned} q_+ = Q_m V(:,1) &= Q_m \left[\frac{1}{R(1,1)} VRe_1\right] \\ &= Q_m p(H_m) e_1 = p(A) Q_m e_1 \\ &= p(A) q_1, \end{aligned}$$

where the third equality is given by (6.3.13) and the fourth is given by (6.3.14). □

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

$$AQ_+ = Q_+ H_+ + r_m e_m^* V, \tag{6.3.15}$$

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 (6.3.15) 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

$$r_m e_m^* V(:, 1 : j) = V(m, j) r_m e_j^*. \tag{6.3.16}$$

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

$$\begin{aligned} Q_+H_+(:, 1:j) &= Q_+(:, 1:j)H_+(1:j, 1:j) + Q_+(:, j+1:m)H_+(j+1:m, 1:j) \\ &= Q_+(:, 1:j)H_+(1:j, 1:j) + H_+(j+1, j)Q_+(:, j+1)e_j^*. \end{aligned} \tag{6.3.17}$$

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 (6.3.15-6.3.17) give the new j-step Arnoldi decomposition

$$\begin{aligned} AQ_j &= AQ_+(:, 1:j) \\ &= Q_+(:, 1:j)H_+(1:j, 1:j) + \left[V(m, j)r_m + H_+(j+1, j)Q_+(:, j+1)\right] e_j^* \\ &= Q_j H_j + r_j e_j^*, \end{aligned} \tag{6.3.18}$$

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

(5) Combining Algorithms 5 and 6 as described above, we have Algorithm 7 as presented in [**GVL12**, 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 largest algebraic eigenvalues j, maximum Arnoldi decomposition (6.3.4) size m.

Output: Approximate largest algebraic 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 (6.3.18).

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 largest algebraic 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 one of the two eigenvalue methods we use in Chapter 7 to handle the EMEP (6.2.3). The choice of parameters m (the Arnoldi decomposition size) and j (number of requested eigenvalues) can greatly impact the efficiency of IRAM (see Section 7.2). 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 largest algebraic 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 7.2, 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 numerical software package `ARPACK` (the ARnoldi PACKage) is an implementation of IRAM in FORTRAN 77 [**LSY98**]. 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. The inverse-free preconditioned Krylov subspace method

(1) In this section we develop the *inverse-free preconditioned Krylov subspace method* (EIGIFP), another large-scale eigenvalue method which may be used for problems like the EMEP (6.2.3). First proposed by Golub and Ye in [**GY02**], the EIGIFP is a method for solving the *generalized eigenvalue problem* (GEP)

$$\begin{aligned} &\text{find} \quad (\lambda, v) \\ &\text{s.t.} \quad Av = \lambda Bv, \end{aligned} \tag{6.4.1}$$

where $A \in \mathbb{C}^{n\times n}$ is Hermitian and $B \in \mathbb{C}^{n\times n}$ is positive definite. We say that a pair (λ, v) are an *eigenpair of* (A, B) if $Av = \lambda Bv$. Additionally, for a vector x, the GEP has the *residual*

$$r(x) = \left(A - \frac{x^*Ax}{x^*Bx}B\right)x, \tag{6.4.2}$$

where the condition $r(x) = 0$ is equivalent to $(r(x), x)$ being an eigenpair of (A, B).

The EIGIFP has a two-phase algorithmic structure similar to the IRAM (Algorithm 7) discussed in the last section. First, a Krylov subspace (6.3.3) $\mathcal{K}$ is formed with Ritz values (1.2.9) for A with respect to $\mathcal{K}$ which tend to approximate the desired eigenvalues of A. Next, the Ritz values for A with respect to $\mathcal{K}$ are used to restart the Krylov subspace. Yet the IRAM and EIGIFP differ in both the choice of Krylov subspace (6.3.3) and method for restarting this subspace.

To develop the EIGIFP, we will frame this algorithm as the logical extension of a few more elementary eigenvalue methods. We begin with the *steepest descent* (SD) method for (6.4.1) and show how this method can be extended to the *locally-optimal conjugate gradient* (LOCG) method for (6.4.1). Next, we show that the LOCG method may also be extended to develop the Krylov subspace (6.3.3) used in EIGIFP. We also present the *m-step Lanczos iteration* for restarting the EIGIFP Krylov subspace, and conclude with the *block inverse-free preconditioned Krylov subspace method* (BLEIGIFP). Altogether, the algorithms in this section are presented in the following order.

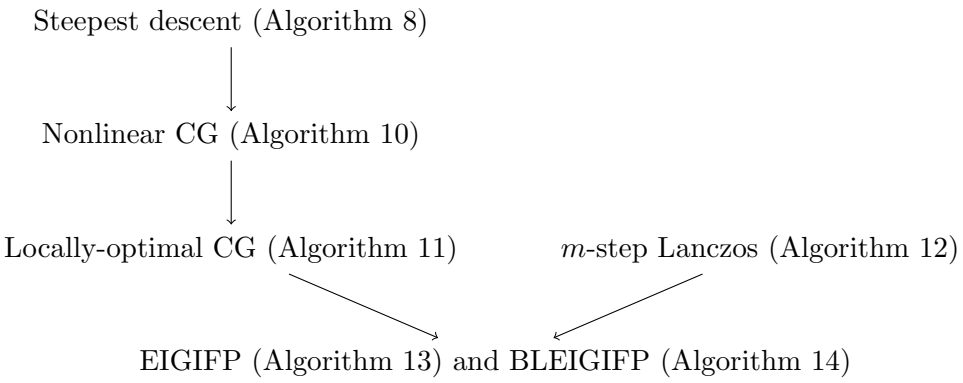

FIGURE 6.5. Dependency chart for the EIGIFP and BLEIGIFP.

The Krylov subspace method underlying the EIGIFP was first introduced with a different preconditioning strategy in [**Kny87**]. We consider the version proposed and analyzed in [**GY02**] and implemented based on the technical report [**MY05**]. The BLEIGIFP was first presented in [**QY10**]. For a treatment of SD and LOCG methods, the reader may reference [**Li15**].

(2) We begin by developing the *steepest descent* (SD) method for the GEP (6.4.1). Recall that a steepest descent algorithm involves a function $f(x)$ which is minimized iteratively. First, an iterate x is used to determine the steepest descent direction p of $f(x)$. A linesearch is then performed to solve the problem

$$\min_{0<t<\infty} f(x+tp),$$

and the solution corresponds to a new iterate x_+.

To develop a steepest descent algorithm for finding the smallest algebraic eigenvalue of (A,B), we may frame this eigenvalue problem as the *Rayleigh quotient* (RQ) minimization problem

$$\min_{x\in\mathbb{R}^n} \rho(x) := \frac{x^TAx}{x^TBx}, \tag{6.4.3}$$

where $\rho(x) := x^TAx/x^TBx$ is the *Rayleigh quotient* (RQ), $A \in \mathbb{R}^{n\times n}$ is symmetric, and $B \in \mathbb{R}^{n\times n}$ is positive definite. Note that we define (6.4.3) as a minimization problem to frame the following methods as "descent" method. If $A = -A$, then (6.4.3) will return the largest algebraic eigenvalue of (A,B).

Since A and B in (6.4.3) are real, the RQ has a well-defined gradient

$$\nabla\rho(x) = \frac{2}{x^TBx}\left[Ax - \rho(x)Bx\right] \tag{6.4.4}$$

and a corresponding steepest descent direction $p := -\nabla\rho(x)$. Given an iterate x of (6.4.3) and steepest descent direction p, we must now perform a linesearch by solving the problem

$$\min_{0<t<\infty} \rho(x+tp). \tag{6.4.5}$$

If x and p are linearly independent, then we may perform an optimal linesearch simply by projecting the problem (6.4.5) onto the appropriate subspace of $\mathbb{R}^n$. Set $t = \beta/\alpha$, $w = [\alpha;\beta]$, let Q_1 be a matrix with orthonormal columns spanning $\{x,p\}$, and set $A_1 =$

$Q_1^T A Q_1$, $B_1 = Q_1^T B Q_1$. Then we have

$$\begin{aligned}\min_{0<t<\infty} \rho(x+tp) &= \min_{0<t<\infty} \frac{(x+tp)^T A(x+tp)}{(x+tp)^T B(x+tp)} \cdot \frac{\alpha^2}{\alpha^2} \\ &= \min_{\alpha,\beta>0} \frac{(\alpha x+\beta p)^T A(\alpha x+\beta p)}{(\alpha x+\beta p)^T B(\alpha x+\beta p)} \\ &= \min_{\alpha,\beta>0} \rho(\alpha x+\beta p) \\ &= \min_{||w||\neq 0} \frac{w^T A_1 w}{w^T B_1 w}.\end{aligned} \tag{6.4.6}$$

Finally, note that the steepest descent direction $p = -\nabla\rho(x)$ from (6.4.4) is parallel to the GEP residual (6.4.2) $r(x) = Ax - \rho(x)Bx$, and thus we may replace p with $r(x)$ in (6.4.6). Then the SD method for the RQ minimization problem (6.4.3) has the following sequence of steps.

Algorithm 8 Steepest descent (SD) method for GEP

Input: Matrices $A, B \in \mathbb{R}^{n\times n}$ from RQ problem (6.4.3), initial iterate x_0.

Output: Approximate largest algebraic eigenpair (λ_1, v_1).

1: *Initialize:* Set $x_0 = x_0/||x_0||$, $r_0 = r(x)$ from (6.4.2), $x_{-1} = 0$, $i = 0$.
2: **while** not converged **do**
3: $Q_1 = \text{orth}(\{x_i, r_i\})$.
4: $A_1 = Q_1^T A Q_1$, $B_1 = Q_1^T B Q_1$.
5: Find the largest algebraic eigenpair (ρ_{i+1}, w_{i+1}) of (A_1, B_1).
6: $x_{i+1} = Q_1 w_{i+1}$.
7: $r_{i+1} = r(x_{i+1})$ from (6.4.2).
8: $i = i + 1$.
9: **end while**
10: *Return:* $\lambda_1 = \rho_i$, $v_1 = x_i$.

(3) While the SD method (Algorithm 8) is computationally very simple and requires minimal memory, it can also be very slow for ill-conditioned problems, exhibiting the zig-zagging behavior typical of steepest descent algorithms. This problem may be remedied by extending the search subspace (Q_1 from step 3 of Algorithm 8) and thus modifying the search

direction. One choice for a modified search direction is to apply the *nonlinear conjugate gradient* (CG) method [**FR64**] to the RQ minimization problem (6.4.3), resulting in the *locally-optimal conjugate gradient* (LOCG) method. We will briefly review the linear and nonlinear CG methods in order to develop the locally-optimal linesearch strategy used in the LOCG method.

We begin with a brief summary of the *linear cojugate gradient* (CG) method (for a more complete treatment, see [**GVL12**, Section 11.3] or [**NW06**, Chapter 5]). First developed in [**HS52**], the linear CG method is a Krylov subspace method for solving the linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is positive definite and the desired solution is $x_\star := A^{-1}b$. Since A is positive definite, it induces an *A-inner product* and *A-norm*, respectively

$$\langle x, y\rangle_A := x^T Ay, \qquad ||x||_A := \sqrt{x^T Ax}. \tag{6.4.7}$$

The linear CG method generates a sequence of iterates x_i which are each optimal in terms of minimizing the function

$$\phi(x) := \frac{1}{2}||x - x_\star||_A^2 = \frac{1}{2}x^T Ax + b^T x + \frac{1}{2}b^T A^{-1}b \tag{6.4.8}$$

over the Krylov subspace

$$\mathcal{K}_i(A, q_1) = \text{span}\left\{q_1, Aq_1, \ldots, A^{i-1}q_1\right\}, \tag{6.4.9}$$

where $q_1 = b$ if the initial iterate is set to $x_0 = 0$. Note that $\phi(x)$ is convex and its gradient $\nabla\phi(x) = Ax - b$ is equal to the residual of the system $Ax = b$. To optimize $\phi(x)$ over $\mathcal{K}_i(A, b)$, a sequence of search directions $\{p_0, \ldots, p_{n-1}\}$ are created which are *conjugate with respect to* A, meaning

$$p_i^T Ap_j = 0 \ \text{ for all } \ i \neq j. \tag{6.4.10}$$

An iteration of the linear CG method has the following sequence of steps. Given a normalized initial iterate x_0, the initial residual $r_0 = Ax_0 - b$ is used to create the initial search direction $p_0 = -r_0$, and we set $i = 0$. The function $\phi(x_i + tp_i)$ from (6.4.8)

is minimized over t. Setting $\frac{\partial}{\partial t}\phi(x_i + tp_i)$ equal to zero and solving for t, we find the closed-form solution

$$t_i = -\frac{p_i^T r_i}{p_i A p_i}. \tag{6.4.11}$$

Next, we update the iterate $x_{i+1} = x_i + t_i p_i$ and residual $r_{i+1} = \nabla\phi(x_{i+1}) = Ax_{i+1} - b$. Finally, we determine a new search direction p_{i+1} which is conjugate to the previous set of search directions. To satisfy this requirement, we set $p_{i+1} = -r_{i+1} + \beta_i p_i$ and require that

$$0 = p_i^T A p_{i+1} = p_i^T A(-r_{i+1} + \beta_i p_i),$$

giving the parameter

$$\beta_i = \frac{r_{i+1}^T A p_i}{p_i A p_i}. \tag{6.4.12}$$

This sequence of steps constitutes an iteration of the linear CG method. Yet equations (6.4.11) and (6.4.12) may be simplified to arrive at expressions for t_i and β_i in common literature (e.g., [**GVL12**, Algorithm 11.3.3], [**NW06**, Algorithm 5.2]). To simplify (6.4.11), assume $\beta_{-1} = 0$ and $p_{-1} = 0$. Then for $i = 0, \ldots, n-1$ we have the search direction update $p_i = -r_i + \beta_{i-1} p_{i-1}$. Note that r_i is orthogonal to p_{i-1} since $\phi(x_{i-1} + tp_{i-1})$ was minimized along p_{i-1} to find t_{i-1}, and then we set $r_i = \nabla\phi(x_{i-1} + t_{i-1}p_{i-1})$. Then $-p_i^T r_i = (r_i - \beta_{i-1}p_{i-1})^T r_i = r_i^T r_i - \beta_{i-1} p_{i-1}^T r_i = r_i^T r_i$, and (6.4.11) is equivalent to

$$t_i = \frac{r_i^T r_i}{p_i A p_i}. \tag{6.4.13}$$

To simplify (6.4.12), assume $x_0 = 0$. By construction of the linear CG method, each iterate x_{i+1} minimizes ϕ over $\mathcal{K}_{i+1}(A, b)$, and thus $r_{i+1} \in \mathcal{K}_{i+1}(A, b)^{\perp}$. Since $r_i \in \mathcal{K}_{i+1}(A, b)$, we have

$$r_{i+1}^T r_i = 0 \quad \text{for all} \quad i = 1, \ldots, n-1. \tag{6.4.14}$$

Additionally, note that the update $r_{i+1} = Ax_{i+1} - b = A(x_{i+1} - x_i) + Ax_i - b = t_i Ap_i + r_i$ implies

$$Ap_i = \frac{1}{t_i}(r_{i+1} - r_i). \tag{6.4.15}$$

Then the numerator in (6.4.12) becomes

$$r_{i+1}^T Ap_i = \frac{1}{t_i}(r_{i+1} - r_i)^T r_{i+1} = \frac{1}{t_i} r_{i+1}^T r_{i+1}, \tag{6.4.16}$$

and the denominator in (6.4.12) becomes

$$p_i^T Ap_i = (-r_i + \beta_{i-1} p_{i-1})^T Ap_i = -r_i Ap_i = \frac{1}{t_i} r_i^T r_i. \tag{6.4.17}$$

And thus (6.4.12) simplifies to

$$\beta_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}. \tag{6.4.18}$$

Altogether, given the steps and simplifications above, we have Algorithm 9.

Algorithm 9 Linear conjugate gradient (CG) method

Input: Differentiable function $\phi(x)$, initial iterate x_0.

Output: Solution x_i.

Initialize: Set $x_0 = x_0/||x_0||$, $r_0 = \nabla\phi(x_0) = Ax_0 - b$, $p_0 = -r_0$, $i = 0$.
while not converged **do**
  $t_i = \arg\min_t \phi(x_i + tp_i) = \frac{r_i^T r_i}{p_i Ap_i}$.
  $x_{i+1} = x_i + t_i p_i$.
  $r_{i+1} = \nabla\phi(x_{i+1}) = Ax_{i+1} - b$.
  $\beta_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$.
  $p_{i+1} = -r_{i+1} + \beta_i p_i$.
  $i = i + 1$.
end while
Return: x_i.

As a consequence of the Cayley-Hamilton theorem [**DF04**, Chapter 12, Proposition 20], the linear CG method in exact arithmetic will terminate in n steps, giving $x_{n-1} = x_\star = A^{-1}b \in \mathcal{K}_n(A,b)$ even if $\mathcal{K}_n(A,b) \neq \mathbb{R}^n$. The Cayley-Hamilton theorem indicates that if $p(\lambda) = \det(\lambda I - A) = \lambda^n + a_1\lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n$ is the characteristic polynomial of A, then $p(A) = A^n + a_1A^{n-1} + \cdots + a_nI = 0$. Then $A^{-1}p(A) = A^{n-1} + a_1A^{n-2} + \cdots + a_{n-1}I + a_nA^{-1}$, which can be solved for A^{-1} giving

$$x_\star = A^{-1}b = -\frac{1}{a_n}\left(A^{n-1}b + a_1A^{n-2}b + \cdots + a_{n-1}b\right) \in \mathcal{K}_n(A,b). \tag{6.4.19}$$

Still assuming exact precision, the linear CG method has the following convergence rate

$$\frac{||x_k - x_\star||_A}{||x_\star||_A} \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k, \tag{6.4.20}$$

dependent on the condition number $\kappa(A) = \lambda_1/\lambda_n$ [**TBI97**, Theorem 38.5].

We will now consider the *nonlinear CG* method, which seeks to minimize a differentiable function $\phi(x)$. The nonlinear CG method is nearly identical to the linear CG method (Algorithm 9) except for the absence of conjugate search directions $\{p_i\}_{i=1}^n$. Recall the linear CG method (Algorithm 9) minimizes the convex quadratic function $\phi(x)$ defined in (6.4.8). If instead $\phi(x)$ is not quadratic then this function will not have a fixed positive definite Hessian, and thus any sequence of search directions $\{p_i\}_{i=1}^n$ cannot be conjugate with respect to any inner product. Thus the convergence properties (6.4.19) and (6.4.20) will not hold for the nonlinear CG method. Despite the lack of convergence guarantees, the nonlinear CG method has been well-studied, and several options exist for choosing the search direction update parameter β_i (Algorithm 9, step 6). The first paper to consider the nonlinear CG method [**FR64**] used the parameter

$$\beta_i = \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}, \tag{6.4.21}$$

where $r_i = \nabla\phi(x_{i+1})$. The author of [**PR69**] found that the parameter

$$\beta_i = \frac{r_{i+1}^T (r_{i+1} - r_i)}{r_i^T r_i} \tag{6.4.22}$$

had superior performance to (6.4.21) in some cases. Given a differentiable function $\phi(x)$ which we seek to minimize and these choices for the search direction update parameter β_i, we have Algorithm 10.

Algorithm 10 Nonlinear conjugate gradient (CG) method

Input: Differentiable function $\phi(x)$, initial iterate x_0.

Output: Solution x_i.

Initialize: Set $x_0 = x_0/||x_0||$, $r_0 = \nabla\phi(x_0)$, $p_0 = -r_0$, $i = 0$.
while not converged **do**
  $t_i = \arg\min_t \phi(x_i + tp_i)$.
  $x_{i+1} = x_i + t_i p_i$.
  $r_{i+1} = \nabla\phi(x_{i+1})$.
  Compute conjugate steplength: β_i, e.g. (6.4.21) or (6.4.22).
  $p_{i+1} = -r_{i+1} + \beta_i p_i$.
  $i = i + 1$.
end while
Return: x_i.

xxx here

If $\phi(x) = \rho(x)$ is the Rayleigh quotient of (6.4.3), then the nonlinear CG method seeks to satisfy the nonlinear optimality condition

$$\nabla\rho(x) = \frac{2}{x^T Bx}\left[Ax - \rho(x)Bx\right] = 0. \tag{6.4.23}$$

Yet when nonlinear CG is applied to the Rayleigh quotient function $\phi(x) = \rho(x)$, the update x_{i+1} may be optimized over both steplengths t_i and β_{i-1}.

In the nonlinear CG method, step 4 has the form

$$\begin{aligned} x_{i+1} &= x_i + t_i\left(-r_i + \beta_{i-1}p_{i-1}\right) \\ &\in \operatorname{span}\left\{x_i, r_i, p_{i-1}\right\} \\ &= \operatorname{span}\left\{x_i, r_i, x_{i-1}\right\}. \end{aligned} \tag{6.4.24}$$

xxx ABOVE USES: step 7 in Algorithm 10, THEN step 4 in same algo

Then if we set $Q_1 = \operatorname{orth}([x_{i-1}, x_i, r_i])$, $A_1 = Q_1^T A Q_1$, and $B_1 = Q_1^T B Q_1$, an argument similar to (6.4.6) demonstrates that the optimal linesearch for t_i, β_{i-1} will be the solution to the problem

$$\min_{||w||\neq 0} \frac{w^T A_1 w}{w^T B_1 w}. \tag{6.4.25}$$

Applying the optimal linesearch for the RQ problem, the nonlinear CG method becomes the locally-optimal conjugate gradient (LOCG) for the RQ problem. This algorithm can also be viewed as the SD method with the addition of x_{i-1} in the search space.

Algorithm 11 Locally-optimal conjugate gradient (LOCG) for GEP

Input: Matrices $A, B \in \mathbb{R}^{n\times n}$ from RQ problem (6.4.3), initial iterate x_0.

Output: Approximate largest algebraic eigenpair (λ_1, v_1).

1: *Initialize:* Set $x_0 = x_0/||x_0||$, $r_0 = r(x_0)$ from (6.4.2), $x_{-1} = 0$, $i = 0$.
2: **while** not converged **do**
3: $Q_1 = \operatorname{orth}(\{x_{i-1}, x_i, r_i\})$.
4: $A_1 = Q_1^T A Q_1$, $B_1 = Q_1^T B Q_1$.
5: Find the largest algebraic eigenpair (ρ_{i+1}, w_{i+1}) of (A_1, B_1).
6: $x_{i+1} = Q_1 w_{i+1}$.
7: $r_{i+1} = r(x_{i+1})$ from (6.4.2).
8: $i = i + 1$.
9: **end while**
10: *Return:* $\lambda_1 = \rho_i$, $v_1 = x_i$.

(4) xxx MAKE CLEAR WE JUSTIFY shift-and-invert strategy before stating algo

The inverse-free preconditioned Krylov subspace (eigifp) method extends the SD subspace $Q_1 = \text{span}\{x, (A-\rho B)x\}$ to the Krylov subspace

$$\mathcal{K}_m(A-\rho B, x) = \text{span}\{x, (A-\rho B)x, (A-\rho B)^2 x, \ldots, (A-\rho B)^{m-1}x\}. \tag{6.4.26}$$

The choice of this Krylov subspace comes from the effectiveness of shift-and-invert preconditioning strategies for solving eigenvalue problems [**Saa11**, Chapter 8]. To simplify notation, we briefly consider the standard eigenvalue problem. If $\rho < \lambda_1$ is a current estimate of λ_1, then we may instead consider the shifted and inverted matrix

$$C = (A-\rho I)^{-1}. \tag{6.4.27}$$

The largest algebraic eigenvalue of C will have the form $\theta_1 = 1/(\lambda_1 - \rho)$, so ρ and θ_1 may be used to recover $\lambda_1 = \rho + 1/\theta_1$. Additionally, if ρ is close to λ_1, then θ_1 will be very large and the problem of finding $\lambda_1(C)$ may be much better conditioned that the original problem $\lambda_1(A)$. Also, if $\lambda_2 < \rho < \lambda_1$, then the shifted and inverted eigenvalue problem will have θ_1 as its only positive eigenvalue.

For the GEP, the shifted and inverted matrix

$$C = B(A-\rho B)^{-1}B \tag{6.4.28}$$

replaces the eigenvalue problem $\lambda_1 = \lambda_1(A, B)$ with $\theta_1 = \lambda_1(C)$. In this case

$$Cx = \theta_1 Bx \quad \Longrightarrow \quad Ax = \left(\rho + \frac{1}{\theta_1}\right) Bx, \tag{6.4.29}$$

and thus if ρ is sufficiently close to λ_1 we again recover $\lambda_1 = \rho + 1/\theta_1$.

If the shift-and-invert technique is used exactly, then the shifted linear system

$$(A-\rho B)\, x_+ = Bx \tag{6.4.30}$$

must be solved directly at each step. Yet for large eigenvalue problems like the PLGD EMEP, the factorization of $A-\rho B$ is prohibitive and $C-$products must be approximated.

If we use an inverse iteration to approximate $x_+ = CBx = (A - \rho B)^{-1}Bx$, then x_+ will be in the subspace $\mathcal{K}_m(A-\rho B, x)$ for some iterate count m. Eigifp uses this "inverse-free" Krylov subspace, applying the corresponding projection to the shifted pencil $(A - \rho B, B)$ and updating $\rho_+ = \rho + 1/\theta_1$.

While this is theoretically equivalent to projecting (A, B) directly onto Q_m, the authors of [**GY02**] observe that in practice this saves matrix multiplications by reusing $(A-\rho B)Q_m$ computed in Algorithm 12 and is also potentially more stable (see [**GY02**, Section 4, Example 1]).

(5) The eigifp method has to main steps: constructing a basis for the Krylov subspace $\mathcal{K}_m(A - \rho B, x)$ and solving the eigenvalue problem on this subspace to compute eigenpair updates. Construction of the Krylov subspace is performed with Algorithm 12, which is computationally identical to the Arnoldi iteration if the Hessenberg matrix H_m were not formed.

Algorithm 12 m-step Lanczos iteration

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

Output: Basis Q_m.

1: *Initialize:* Set $q_1 = q_1/||q_1||, \quad Q_1 = [q_1]$.
2: **for** $i = 1, \ldots, m-1$ **do**
3: $\quad z = Cq_i$.
4: $\quad h = Q_i^* z, \quad r_i = z - Q_i h$.
5: $\quad \beta_i = ||r_i||, \quad q_{i+1} = r_i/\beta_i$.
6: $\quad Q_{i+1} = [Q_i \mid q_{i+1}]$.
7: **end for**
8: *Return:* Q_m.

The Lanczos iteration provides the inner iteration of the single-vector inverse-free preconditioned Krylov subspace method. Note that step 3 of eigifp includes the previous iterate which we will label $\overline{x}$ in the search space. This blending of the inverse-free preconditioned Krylov subspace with the CG method has been shown to accelerate the algorithm while adding minimal cost [**MY05**].

Algorithm 13 Inverse-free preconditioned Krylov subspace method (eigifp)

Input: Matrices $A, B \in \mathbb{C}^{n\times n}$ with B positive definite, initial approximate eigenvector x_0, maximum Lanczos basis size m, convergence tolerance $\text{tol}_{\text{rel}} > 0$.

Output: Approximate largest algebraic eigenpair (λ_1, v_1).

1: *Initialize:* Set $x_0 = x_0/||x_0||$, $\rho_0 = x_0^* A x_0 / x_0^* B x_0$, $\overline{x} = 0$, and $i = 0$.
2: **while** not converged **do**
3: *Algorithm 12:* Perform the m-step Lanczos iteration on $C = A - \rho_i B$ with initial vector $q_1 = x_i$ to obtain Q_m.
4: Orthogonalize $\overline{x}$ against Q_m to obtain q_{m+1} and set $\widehat{Q}_m = [Q_m \mid q_{m+1}]$.
5: Form $A_m = \widehat{Q}_m^*(A - \rho_i B)\widehat{Q}_m$ and $B_m = \widehat{Q}_m^* B \widehat{Q}_m$.
6: Compute the largest algebraic eigenpair (θ_1, u_1) of (A_m, B_m).
7: Update $\rho_{i+1} = \rho_i + \theta_1$, $x_{i+1} = \widehat{Q}_m u_1$, $\overline{x} = x_{i+1} - x_i$, and $i = i + 1$.
8: **end while**
9: Set $\lambda_1 = \rho_i$, $v_1 = x_i$.
10: *Return:* (λ_1, v_1).

If a preconditioner L_k is provided at iterate k, then eigifp only differs in the inner Lanczos iteration. In this case, the vector $L_k q_1$ is used to construct a Krylov subspace for the transformed eigenvalue problem $(L_k^{-1} A L_k^{-*}, L_k^{-1} B L_k^{-*})$, and then this subspace is premultiplied by L_k^{-*} and orthogonalized to return Q_m. For complete implementation details, see [**GY02**, Section 5]. Since the matrix A in the PLGD EMEP is represented implicitly, this preconditioning is not an option.

The PLGD EMEP requires the computation of the two largest algebraic eigenvalues, yet eigifp only computes one eigenvalue at a time. In order to compute multiple eigenvalues, eigifp is implemented with a deflation strategy. The basic idea behind deflation is that the k largest algebraic eigenvalues and eigenvectors

$$\Lambda_k = \text{diag}(\lambda_1, \ldots, \lambda_k), \qquad V_k = [v_1, \ldots, v_k] \tag{6.4.31}$$

are used to construct a deflated matrix

$$\widehat{A} = A - V_k \Lambda_k V_k^T. \tag{6.4.32}$$

Passing $\widehat{A}$ to eigifp will then return $\lambda_1(\widehat{A}) = \lambda_{k+1}$. Thus eigifp must be called k times to return the k largest algebraic eigenvalues. Additionally, if these eigenvalues of A are clustered, then this algorithm can converge slowly.

The typical remedy for these challenges is to replace the single vector iterate in eigifp with a block of vectors. The resulting block inverse-free preconditioned Krylov subspace method (bleigifp) was developed and analyzed in [**QY10**].

Algorithm 14 Block inverse-free preconditioned Krylov subspace method (bleigifp)

Input: Matrices $A, B \in \mathbb{C}^{n\times n}$ with B positive definite, initial approximate eigenvectors $Q_j = [q_1, \ldots, q_j]$, maximum Lanczos basis size m, convergence tolerance $\text{tol}_{\text{rel}} > 0$.

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

Initialize: For $i = 1, \ldots, j$, set $q_i = q_i/||q_i||$, $\rho_i = q_i^* A q_i / q_i^* B q_i$, and $\overline{q}_i = 0$.
while not converged **do**
  for $i = 1, \ldots, j$ **do**
    Algorithm 12 with CG step: Construct a basis $Q^{(i)}$ of $\{\overline{q}_i, \mathcal{K}_m(A - \rho_i B, q_i)\}$.
  end for
  Orthonormalize $\left\{Q^{(1)}, \ldots, Q^{(j)}\right\}$ to obtain Q_m.
  Form Krylov subpace problem with $A_m = Q_m^* A Q_m$ and $B_m = Q_m^* B Q_m$.
  Compute the j eigenpairs $(\rho_i, w_i)_{i=1}^j$ of (A_m, B_m).
  Update $\overline{q}_i = q_i$ and $q_i = Q_m w_i$ for $i = 1, \ldots, j$.
end while
Set $\Lambda_j = \text{diag}(\rho_1, \ldots, \rho_j)$, $V_j = [q_1, \ldots, q_j]$.
Return: (Λ_j, V_j)..

Black-box implementations of these inverse-free preconditioned Krylov subspace methods are available in MATLAB[1]. The single-vector implementation `eigifp` is discussed

[1] `https://www.ms.uky.edu/~qye/software.html`

in [**MY05**], and its block extension `bleigifp` is discussed in [**QY10**]. In the next section, we examine the effectiveness of these methods and IRAM on the PLGD EMEP.

CHAPTER 7

Numerics

7.1. Experimental models and software

This chapter presents a new algorithm for solving the EMEP (6.2.3) using the IRAM (Algorithm 7, Section 6.3). Section 7.2 develops the new algorithm based on intuition regarding the behavior of the IRAM and the structure of the EMEP (6.2.3). Section 7.3 then demonstrates the efficiency of this algorithm and provides heuristics suggesting the reason for this efficiency. All experiments in this chapter are available for reproduction[1]

7.2. Adaptive inner iteration method for the EMEP

(1) In this section we develop a new algorithm for solving the EMEP (6.2.3). This new algorithm uses the IRAM (Algorithm 7) to handle each EMEP matrix iterate A_k while adaptively changing the IRAM parameters based on the results from the EMEP iterates. As discussed in Section 6.3, the IRAM has only two parameters: the number of requested eigenvalues j and the Arnoldi decomposition (6.3.4) size m. As we will see, the proper choice of these parameters can greatly reduce computational costs in the EMEP (6.2.3).

(2) To determine how the IRAM parameters should be changed adaptively, we must understand and exploit the correlation between the spectrum of a matrix iterate A_k and behavior of the IRAM in finding the two largest algebraic eigenvalues of A_k required for the EMEP (6.2.3). In the original implementation of Algorithm 3, all matrix iterates A_k were handled using the IRAM with $j = 2$ requested eigenvalues and Arnoldi decomposition size $m = 20$. Yet as we saw in Section 6.2, Figure 6.3, the later EMEP (6.2.3) matrix iterates can have largest algebraic eigenvalues which cluster together, i.e., $\lambda_1 \approx \lambda_2 \approx \cdots \approx \lambda_t$ for some t. If these eigenvalues are very close and some subset of them $\{\lambda_{j+1}, \lambda_{j+2}, \ldots, \lambda_t\}$ are used in the shifted QR iteration (Algorithm 6) to restart the Arnoldi decomposition

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

in the IRAM, then the shifted QR iteration may have the reverse effect of shifting the Arnoldi decomposition away from the desired eigenvectors. To avoid this phenomenon, we propose an adaptive method for choosing the number of requested eigenvalues j_k for each matrix iterate A_k based on the behavior of the IRAM for previous matrix iterates.

To develop an adaptive method for choosing the number of requested eigenvalues j_k, we begin by selecting a fixed Arnoldi decomposition size m and initializing $j_1 = j_{min}$ and $j_2 = j_1 + 1$ (with default values $m = 40$, $j_{min} = 2$, and $j_{max} = m - 1$). At each step $k \geq 2$ we update $j_{k+1} = \delta + j_k$, where $\delta \in \{-2, -1, 1, 2\}$ is a shift based on the number of requested eigenvalues $j_k, j_{k-1}, \ldots$ and number of matrix-vector products $mv_k, mv_{k-1}, \ldots$ for the previous eigenvalues problems. The shift δ is computed using two basic comparisons. First, we determine a 2-*step shift value* $\delta_2 \in \{-1, 1\}$ based j_{k-1}, j_k and mv_{k-1}, mv_k. If $j_k > j_{k-1}$ and $mv_k < mv_{k-1}$ then the computational cost of the EMEP (6.2.3) decreased as the number of requested eigenvalues was increased, suggesting we should shift j_k by $\delta_2 = 1$. By the same reasoning for the other three inequality cases, we define the 2-*step shift value* as

$$\delta_2 = \text{sign}(j_k - j_{k-1}) \cdot \text{sign}(mv_{k-1} - mv_k), \tag{7.2.1}$$

where sign(0) is defined as 1. Next, if $k \geq 4$ then we compute a linear interpolation of the past four requested eigenvalue numbers and matrix-vector products by solving

$$\min_{\alpha,\beta} ||y - \alpha e - \beta x||, \tag{7.2.2}$$

where x is the vector of matrix-vector product values $mv_{k-3}, mv_{k-2}, mv_{k-1}, mv_k$, y is the vector of the number of requested eigenvalues $j_{k-3}, j_{k-2}, j_{k-1}, j_k$, and $e = [1; 1; 1; 1]$. If the solution to (7.2.2) has $\beta > 0$ then the past four eigenvalue problems suggest that mv_i increases with j_i, and thus we should decrease j_k. Thus we have the 4-*step shift value*

$$\delta_4 = -\text{sign}(\beta), \tag{7.2.3}$$

where β is determined by (7.2.2). If $\delta_2 = \delta_4$, then the 2-step (7.2.1) and 4-step equations (7.2.3) both suggest we should shift in the value 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 (7.2.1) and select the shift $\delta = \delta_2$. Finally, if $j_k = j_{min}$ then we set $\delta = 1$ and if $j_k = j_{max}$ the we set $\delta = -1$. Altogether, these steps lead to Algorithm 15.

Algorithm 15 Adaptive inner iteration method for the EMEP (6.2.3)

Input: Sequence of matrices $\{A_k\}_{k=1}^{maxit}$ from the EMEP (6.2.3), Arnoldi decomposition (6.3.4) size m (default parameter $m = 40$).

Output: EMEP (6.2.3) solution eigenpairs $\{(\lambda_1^{(k)}, v_1^{(k)})\}_{k=1}^{maxit}$ and $\{(\lambda_2^{(k)}, v_2^{(k)})\}_{k=1}^{maxit}$.

Initialize: $j_{min} = 2$, $j_{max} = m - 1$, $j_1 = j_{min}$, $mv_0 = -1$, $k = 1$.
while $k \leq maxit$ **do**
    Algorithm 7: Perform IRAM with matrix A_k, number of requested largest algebraic eigenvalues j_k, and maximum Arnoldi decomposition size m. Return eigenpairs $(\lambda_1^{(k)}, v_1^{(k)})$, $(\lambda_2^{(k)}, v_2^{(k)})$ and number of matrix-vector products mv_k.
    if $j_k = j_{min}$ **then**
        $j_{k+1} = j_k + 1$
    else if $j_k = j_{max}$ **then**
        $j_{k+1} = j_k - 1$
    else if $k < 4$ **then**
        Compute 2-step shift value δ_2 from (7.2.1) and set $\delta = \delta_2$
        $j_{k+1} = j_k + \delta$
    else
        Compute 2-step shift value δ_2 from (7.2.1) and 4-step shift value δ_4 from (7.2.3)
        if $\delta_2 = \delta_4$ **then**
            Set $\delta = 2\delta_2$
        else
            Set $\delta = \delta_2$
        end if
        $j_{k+1} = \min\{\max\{j_k + \delta, j_{min}\}, j_{max}\}$
    end if
    $k = k + 1$
end while
Return: $\{(\lambda_1^{(k)}, v_1^{(k)})\}_{k=1}^{maxit}$ and $\{(\lambda_2^{(k)}, v_2^{(k)})\}_{k=1}^{maxit}$.

Note that the only parameter in Algorithm 15 is 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 $AQ_m = Q_m H_m + r_m e_m^*$. As we will see in Section 7.3, m must be sufficiently large for the shifted QR iteration (Algorithm 6) in the IRAM to handle the EMEP (6.2.3) efficiently. However, each column of Q_m is the size of the desired signal $\bar{x}$ in the phase retrieval problem (2.1.1). Since Q_m must be stored in random-access memory. Thus the choice of m is a trade-off between computational efficiency and data storage constraints. We find that the default parameter $m = 40$ strikes this balance properly.

7.3. Results for the adaptive inner iteration method

(1) This section demonstrates the efficiency of the adaptive inner iteration method (Algorithm 15) for solving the EMEP (6.2.3). We begin by demonstrating that Algorithm 15 is more efficient than the default IRAM parameter settings for the EMEP (6.2.3). Next, we show that the Arnoldi decomposition size $m = 40$ strikes a proper balance between increasing computational efficiency and minimizing data storage. Finally, we show that Algorithm 15 is nearly optimal as a method for choosing the ideal number of requested eigenvalues j_k corresponding to the minimum number of matrix-vector products necessary for each EMEP (6.2.3) iteration. In particular, an increase in the number of requested eigenvalues j_k in Algorithm 15 is shown to correspond with increased clustering of the largest algebraic eigenvalues of A_k, thus allowing the shifted QR iteration (Algorithm 6) to restart the Arnoldi decomposition properly.

The experiments in this section involve six EMEPs (6.2.3) which were created using the image first discussed in Figure 2.2, resized to 64×64 pixels. These EMEPs (6.2.3) are based on PLGD models with Gaussian noise (5.2.3) with noise ratios $\epsilon_{\text{rel}} = 0.05, 0.15, 0.30$ and oversampling rates $L = 5, 10$. To examine the behavior of Algorithm 15, each EMEP was solved for all possible parameter combinations of Arnoldi decomposition size $m = 20, 40, 60, 80, 100$ and number of requested eigenvalues $j = 2, 3, \ldots, \min(30, m-5)$. For all experiments, computational cost is measured in terms of the number of matrix-vector products required to solve the EMEP (6.2.3).

(2) Table 7.1 depicts the computational cost for solving these six EMEPs with the default IRAM parameters and Algorithm 15 with various Arnoldi decomposition sizes m.

n = 4,096			Default	Adaptive inner iteration method (Algorithm 15)				
L	ϵ_{rel}	EMEP its	$j = 2, m = 20$	$m = 20$	$m = 40$	$m = 60$	$m = 80$	$m = 100$
5	0.05	300	406,308	358,195	198,169	189,295	192,042	201,270
5	0.15	300	1,099,045	806,412	258,574	225,736	214,083	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	67,507	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 7.1. Total number of matrix-vector products for various EMEPs (6.2.3) with initial image from Figure 2.2. Parameter j is the number of requested eigenvalues in the IRAM (Algorithm 7) and m is the Arnoldi decomposition size (6.3.4).

Table 7.1 demonstrates that Algorithm 15 reduces computational costs from those of the default IRAM parameters for all experiments considered. Yet this cost reduction varies significantly depending on the choice of Arnoldi decomposition (6.3.4) size m. Thus we seek to select a default setting for the parameter m which is sufficiently large to yield the benefits of Algorithm 15. To select the value for m, we examine the two experiments from Table 7.1 with $\epsilon_{\text{rel}} = 0.15, 0.30$ and $L = 5$ which have the greatest default computational cost, along with the greatest total decrease in cost when using Algorithm 15 with a sufficiently large parameter m. Figure 7.1 singles out these two experiments, depicting the number of matrix-vector products for each EMEP (6.2.3) iteration.

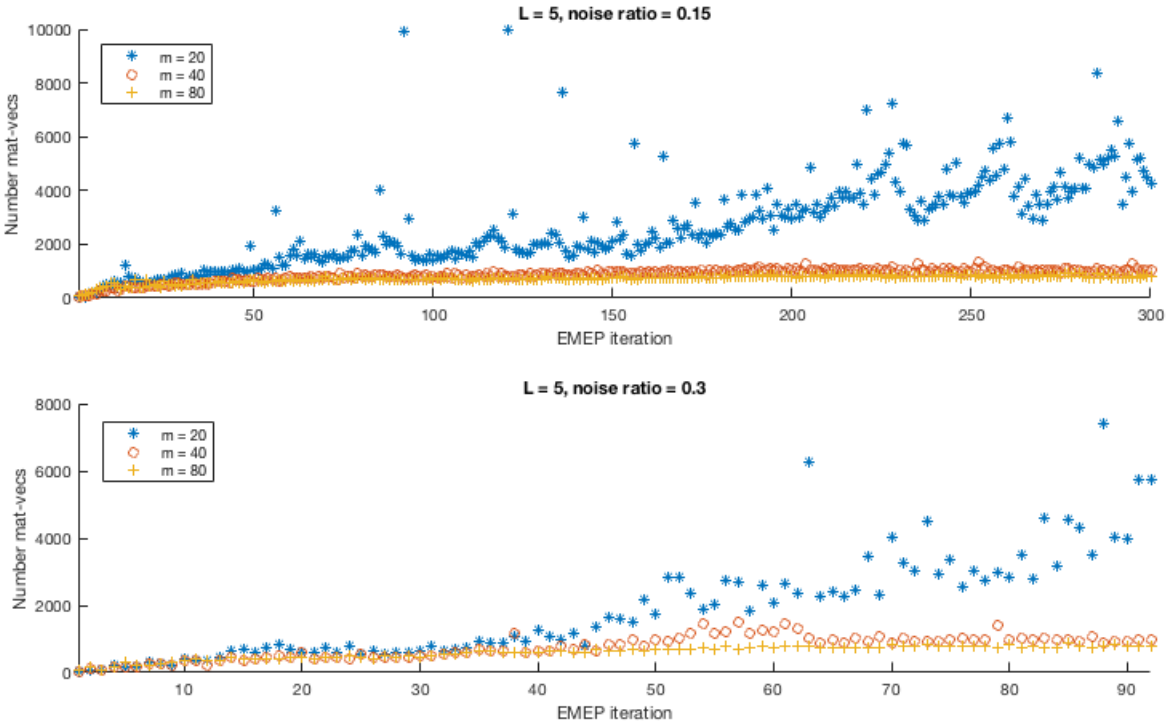

FIGURE 7.1. Number of matrix-vector products for each EMEP (6.2.3) iteration from two experiments in Figure 7.1 with various Arnoldi decomposition size $m = 20, 40, 80$.

Figure 7.1 demonstrates that the Arnoldi decomposition size of $m = 20$ is not sufficiently large to allow Algorithm 15 to decrease computational costs. The dramatic computational cost spikes for $m = 20$ in Figure 7.1 resemble those first seen in Figure 6.1 when solving the EMEP (6.2.3) with the default IRAM parameters $m = 20, j = 2$. Yet when the Arnoldi decomposition size is increased to $m = 40$, these cost spikes effectively disappear. The change in computational cost between $m = 40$ and $m = 80$ is minimal for each EMEP iterate. Thus the default parameter of $m = 40$ for Algorithm 15 strikes the proper balance between efficiency and data storage.

(3) Next, we demonstrate that Algorithm 15 with parameter $m = 40$ is nearly optimal in the sense of choosing the number of requested eigenvalues j_k which minimizes the computational cost for each EMEP (6.2.3) iterate k. Table 7.2 restates the computational cost for solving the six EMEPs from Table 7.1 with an additional column depicting the minimal

possible computational cost if each value j_k was chosen such that $2 \leq j_k \leq m-1$ and j_k corresponds to the minimal number of matrix-vector products for the IRAM (Algorithm 7) to handle matrix iterate A_k with parameters (m, j_k).

			Default	Optimal $2 \leq j \leq m-1$		Algorithm 15	
L	ϵ_{rel}	EMEP its	$j = 2, m = 20$	$m = 40$		$m = 40$	
5	0.05	300	406,308	179,807	56%	198,169	51%
5	0.15	300	1,099,045	242,003	78%	258,574	76%
5	0.30	92	444,697	58,780	87%	69,510	84%
10	0.05	153	80,453	61,948	23%	68,709	15%
10	0.15	108	88,317	51,311	42%	57,231	35%
10	0.30	54	72,486	23,217	68%	25,809	64%

TABLE 7.2. Total number of matrix-vector products and percent decrease from the default IRAM parameters for various EMEPs (6.2.3) with initial image from Figure 2.2. Parameter j is the number of requested eigenvalues in the IRAM (Algorithm 7) and m is the Arnoldi decomposition size (6.3.4).

Table 7.2 demonstrates that Algorithm 15 decreases the computational cost of each EMEP (6.2.3) from the default IRAM parameters by a percentage comparable to that of the optimal choice for parameter j. Notably, Algorithm 15 is particularly effective at decreasing computational costs when the costs for the default IRAM parameters are much greater than the costs for the optimal parameters.

(4) To understand why Algorithm 15 is particularly effective for EMEPs (6.2.3) with high default computational costs, we will examine the behavior of Algorithm 15 and structure of EMEP for the experiments from Table 7.2 with $L = 5$ and $\epsilon_{\text{rel}} = 0.15, 0.30$. Figure 7.2 depicts the number of requested eigenvalues j_k for each EMEP (6.2.3) iterate k in these two experiments.

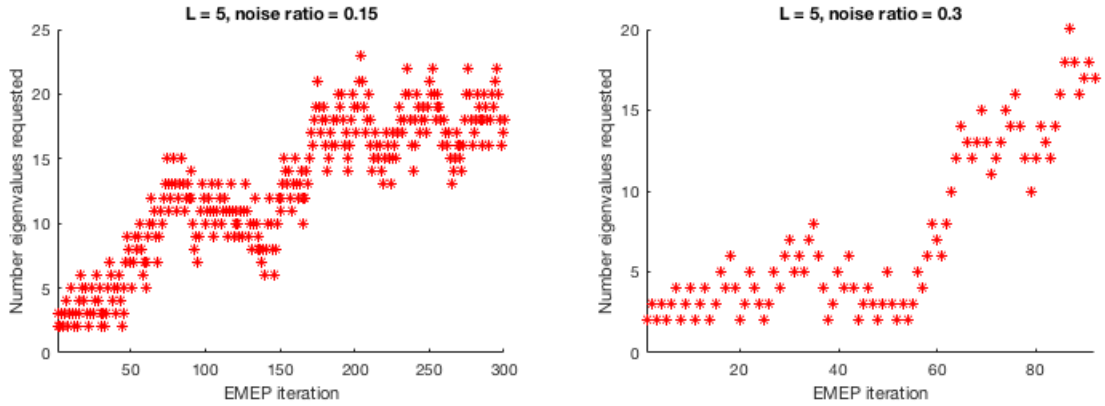

FIGURE 7.2. Number of requested eigenvalues j chosen by Algorithm 15 for two experiments from Table 7.2 with $\epsilon_{\text{rel}} = 0.15, 0.30$ and $L = 5$.

For both experiments in Figure 7.2, the number of requested eigenvalues j_k chosen by Algorithm 15 changes greatly from early to later EMEP iterates. In particular, the experiment with $L = 5$ and $\epsilon_{\text{rel}} = 0.30$ shows a rapid change in j_k around the 60-th iterate (where several iterates had 2-step shift value (7.2.1) $\delta_2 = 1$ and 4-step shift value (7.2.3) $\delta_4 = 1$, causing Algorithm 15 to increase j_k by 2).

To understand why the number of requested eigenvalues j_k chosen by Algorithm 15 changed greatly for the two experiments in Figure 7.2, we will examine the behavior of the IRAM (Algorithm 7) and the structure of the EMEP (6.2.3) for the iterates which show greatest change in j_k. Figures 7.3 depicts the number of matrix-vector products for the IRAM (top plot) and the difference of the largest algebraic eigenvalues (bottom plot) for a range of EMEP iterates and number of requested eigenvalues from the experiment with $L = 5$ and $\epsilon_{\text{rel}} = 0.15$. Figure 7.4 depicts the same results for the experiment with $L = 5$ and $\epsilon_{\text{rel}} = 0.30$. In both figures, the black dots indicate the number of requested eigenvalues j_k chosen by Algorithm 15 for each EMEP iterate k.

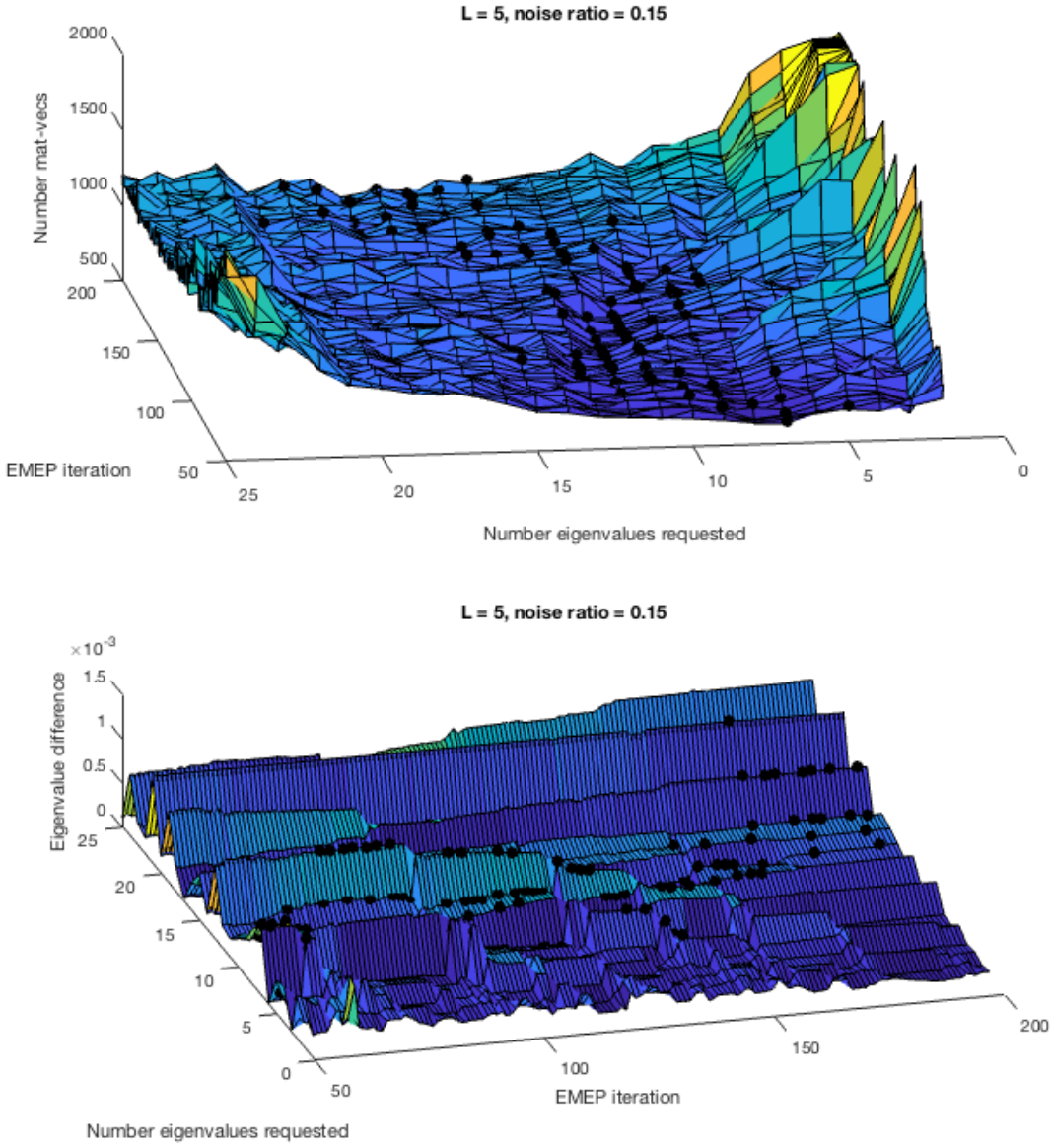

FIGURE 7.3. Behavior of Algorithm 15 for the experiment from Table 7.2 with $L = 5$, $\epsilon_{\text{rel}} = 0.15$. Top: Number of matrix-vector products (capped at 2,000 for easier viewing) for each EMEP (6.2.3) iterate k and number of requested eigenvalues j. Bottom: eigenvalue differences $\lambda_j - \lambda_{j+1}$ for each EMEP (6.2.3) iterate k and number of requested eigenvalues j. Black dots in both plots indicate the value j_k chosen by Algorithm 15.

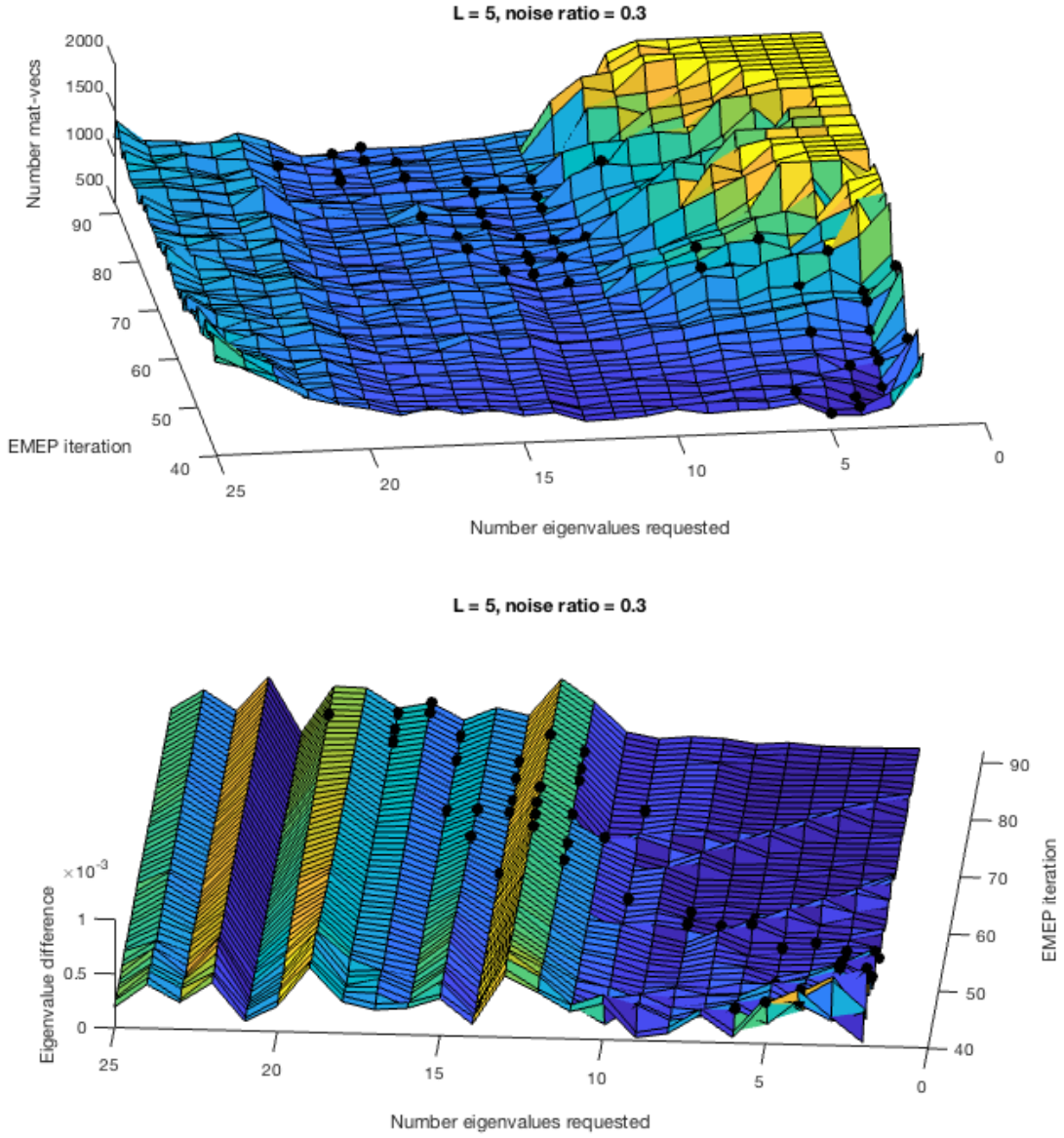

FIGURE 7.4. Behavior of Algorithm 15 for the experiment from Table 7.2 with $L = 5$, $\epsilon_{\text{rel}} = 0.30$. Top: Number of matrix-vector products (capped at 2,000 for easier viewing) for each EMEP (6.2.3) iterate k and number of requested eigenvalues j. Bottom: eigenvalue differences $\lambda_j - \lambda_{j+1}$ for each EMEP (6.2.3) iterate k and number of requested eigenvalues j. Black dots in both plots indicate the value j_k chosen by Algorithm 15.

Figures 7.3 and 7.4 demonstrate why it is necessary to vary the number of requested eigenvalues j_k in the EMEP (6.2.3). In the top plot of both figures, the number of matrix-vector products forms a "valley" in which Algorithm 15 attempts to find the value j_k corresponding to the minimum for each EMEP iterate k. Likewise, the bottom plot in both figures depicts "hills" of eigenvalue differences $\lambda_j - \lambda_{j+1}$ toward which Algorithm 15 appears to shift.

In Figure 7.3, the top plot depicts a "valley" around $j_k = 10$ for EMEP iterates $50 \leq k \leq 125$ which then shifts to the region $15 \leq j_k \leq 20$. Likewise, the bottom plot in Figure 7.3 depicts a "hill" of eigenvalue differences around $j_k = 10$ for EMEP iterates $50 \leq k \leq 125$ which settles into a flat space of decreasing eigenvalue differences $\lambda_j - \lambda_{j+1}$. Around iterate $k = 125$, Algorithm 15 increases j_k, avoiding the eigenvalue differences "valley" in the bottom plot of Figure 7.3 and approaching the nearby hill around $15 \leq j_k \leq 20$.

In a similar manner, the top plot in Figure 7.4 demonstrates a "hill" around $2 \leq j_k \leq 10$ which appears around EMEP iterate $k = 60$, causing Algorithm 15 to shift quickly toward the "valley" around $j_k = 12$ for $k \geq 60$. Likewise, the bottom plot in Figure 7.4 shows that the smaller "hills" of eigenvalue differences for $2 \leq j_k \leq 10$ flatten out around EMEP iterate $k = 60$. Algorithm 15 then shifts to the nearest "hill" which begins at $\lambda_{12} - \lambda_{13}$.

The tendency of Algorithm 15 to approach and settle near "hills" of eigenvalue differences is a consequence of both the structure of the EMEP (6.2.3) and the behavior of the IRAM (Algorithm 7). Section 5.3 demonstrated that PLGD models with Gaussian noise (5.2.3) tend to have optimal dual matrices $\mathcal{A}^* y_\star$ with largest algebraic eigenvalues of multiplicity greater than one. Thus later EMEP (6.2.3) iterates are likely to have several largest algebraic eigenvalues with similar values. To promote the convergence of the IRAM for these later EMEP iterates, we must choose the number of requested eigenvalues j such that the eigenvalue λ_j will have modest separation from λ_{j+1}, thus allowing the shifted QR iteration (Algorithm 6) to shift away the unwanted part of the spectrum.

As we see Figures 7.3 and 7.4, Algorithm 15 successfully handles the EMEP (6.2.3) efficiently by adaptively selecting the number of requested eigenvalues in the IRAM, allowing the shifted QR (Algorithm 6) to promote convergence.

CHAPTER 8

Revised eigenvalue method for the PLGD EMEP

In this chapter we present a revised method for handling the EMEP (6.2.3), the computational bottleneck for first-order PLGD methods like Algorithm 3.

APPENDIX A

Proof of PhaseLift-l_1 gauge duality

In this appendix we show that the following pair of PhaseLift-l_1 models discussed in Section 2.4 are a primal and gauge dual pair

$$\begin{array}{lllll} & \min & ||\mathcal{A}(X)-b||_1 & & \min\limits_{y} \quad ||y||_\infty \\ (\text{PLP-}l_1) & \text{s.t.} & X \succeq 0 & (\text{PLGD-}l_1) & \text{s.t.} \quad -\mathcal{A}^* y \succeq 0 \\ & & & & \quad\ \langle b, y\rangle \geq 1. \end{array} \tag{A.0.1}$$

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

$$\begin{array}{ll} \min\limits_{X,z} & \kappa(X,z) := ||z||_1 + \delta_{\succeq 0}(X) \\ \text{s.t.} & \overline{\mathcal{A}}(X,z) := \mathcal{A}(X) + z = b, \end{array} \tag{A.0.2}$$

which is now in the form of (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

$$\kappa^\circ(V,w) = \max\{||w||_\infty,\ \delta_{\succeq 0}(-V)\} = ||w||_\infty + \delta_{\succeq 0}(-V). \tag{A.0.3}$$

Additionally, the adjoint of $\overline{\mathcal{A}}$ is $\overline{\mathcal{A}}^* y = (\mathcal{A}^* y, y)$, giving

$$\kappa^\circ(\overline{\mathcal{A}}^* y) = ||y||_\infty + \delta_{\succeq 0}(-\mathcal{A}^* y). \tag{A.0.4}$$

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.

APPENDIX B

Further justification of residuals established in Chapter 5

(1) The previous section demonstrated that the primal difference (5.2.7e) and dual variable difference (5.2.7i) effectively identify the point at which Algorithm 3 stagnates for PLGD models with Gaussian noise (5.2.3). We close this chapter with a justification for ignoring the remaining residuals (5.2.7). As we will see, each of these residuals is 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 Algorithm 3 include the feasibility requirement (4.2.13)

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

which is equivalent to the primal relative error (5.2.7d) using the relation $\epsilon_{\text{rel}} = \epsilon/||b||$

$$\frac{||\mathcal{A}(xx^*) - b||}{||b||} \leq \epsilon_{\text{rel}} + \text{tol}_{\text{feas}} \left(\frac{1 + ||b||}{||b||} \right). \tag{B.0.1}$$

As indicated in Table 5.4, Algorithm 3 typically requires only a few iterations before a primal feasible signal x is found. Yet in certain cases this 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.

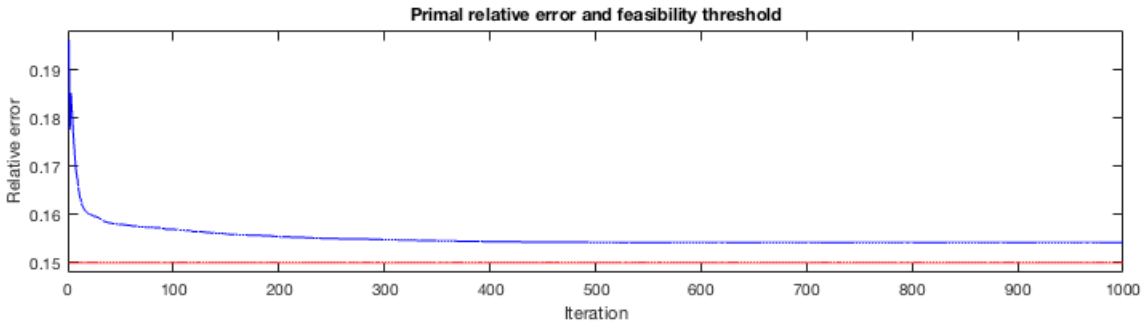

FIGURE B.1. Primal relative error (5.2.7d) (blue) and noise ratio threshold (red) for noise ratio $\epsilon_{\text{rel}} = 0.15$ and oversampling rate $L = 5$. Iterate 1,000 remains infeasible with a primal relative error of 0.1541.

The experiment in Figure B.1 demonstrates a model where both the exact tolerance $\epsilon_{\text{rel}} = 0.15$ and the relaxed tolerance $\epsilon_{\text{rel}} + \text{tol}_{\text{feas}} \left(\frac{1+||b||}{||b||} \right) = 0.1502$ as suggested in (4.2.13) are unattainable by Algorithm 3. Nevertheless, this algorithm makes significant progress in recovering an approximate signal. Notably, wflow 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 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, Algorithm 3 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 Algorithm 3.

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 Algorithm 3 stagnates, $\hat{p}/p$ approaches 1. Additionally, note that typical optimal signals have fairly large norms (i.e., $||\bar{x}\bar{x}^*||_F = \mathcal{O}(10^3)$ 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

$$||A - \hat{A}||_F = ||\mathcal{A}^*(y - \hat{y})||_F \leq ||\mathcal{A}^*||\ ||y - \hat{y}||,$$

where we have the induced norm

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

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 Algorithm 3 would satisfy this tolerance.

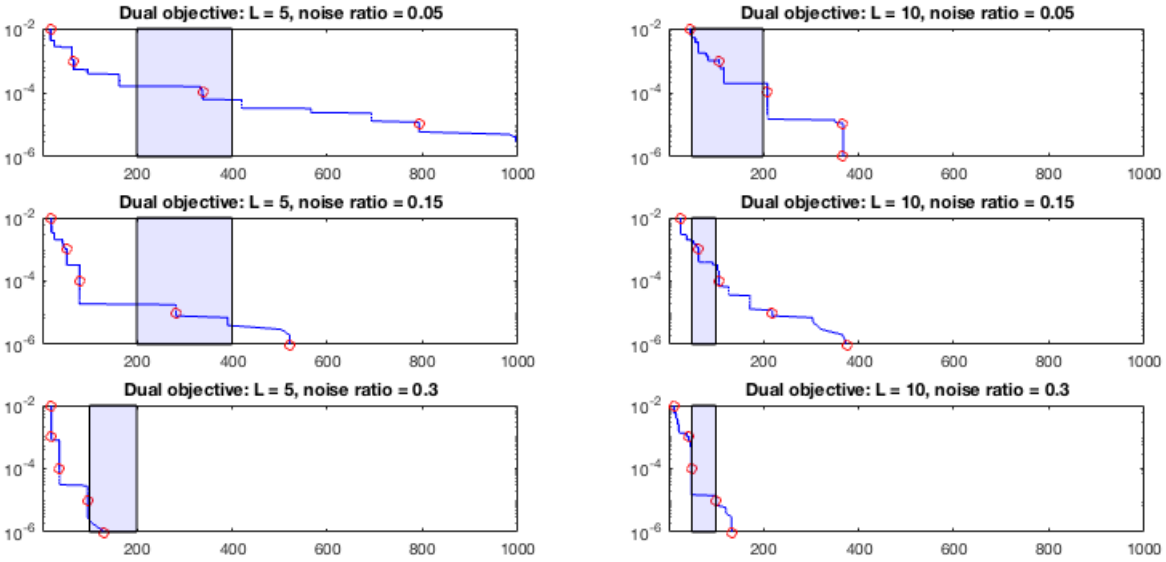

FIGURE B.2. Plots of dual objective difference values (5.2.7h) against the iterate at which Algorithm 3 first satisfies this tolerance for the models discussed in Figure 5.3. Red circles are placed at tolerances 10^{-n}. The blue rectangles indicate the proposed intervals of termination from Table 5.5.

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 Algorithm 3 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 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 Algorithm 3.

Bibliography

[ABD+17] A. Y. Aravkin, J. V. Burke, D. Drusvyatskiy, M. P. Friedlander, and K. MacPhee, *Foundations of gauge and perspective duality*, arXiv preprint arXiv:1702.08649 (2017).

[BB86] R. N. Bracewell and R. N. Bracewell, *The fourier transform and its applications*, third ed., vol. 31999, McGraw-Hill New York, 1986.

[BB88] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA Journal of Numerical Analysis **8** (1988), no. 1, 141–148.

[BCG11] 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), no. 3, 165.

[BDP+07] 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), no. 4, 306–314.

[Ber16] D. P. Bertsekas, *Nonlinear programming*, third ed., Athena Scientific, 2016.

[BR16] S. Bahmani and J. Romberg, *Phase retrieval meets statistical learning theory: A flexible convex relaxation*, arXiv preprint arXiv:1610.04210 (2016).

[BTN01] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, vol. 2, Siam, 2001.

[BV04] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[CDS01] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM review **43** (2001), no. 1, 129–159.

[CESV13] E. J. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski, *Phase retrieval via matrix completion*, SIAM J. Imaging Sciences **6** (2013), no. 1, 199–225.

[CL14] 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), no. 5, 1017–1026.

[CLM+16] 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), no. 5, 2221–2251.

[CLS15] E. J. Candès, X. Li, and M. Soltanolkotabi, *Phase retrieval via wirtinger flow: Theory and algorithms*, IEEE Trans. Information Theory **61** (2015), no. 4, 1985–2007.

[CMP10] A. Chai, M. Moscoso, and G. Papanicolaou, *Array imaging using intensity-only measurements*, Inverse Problems **27** (2010), no. 1, 015005.

[CRT06] 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), no. 8, 1207–1223.

[CSV13] 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), no. 8, 1241–1274.

[DF04] D. S. Dummit and R. M. Foote, *Abstract algebra*, vol. 3, Wiley Hoboken, 2004.

[DMM+11] H. Duadi, O. Margalit, V. Mico, J. A. Rodrigo, T. Alieva, J. Garcia, and Z. Zalevsky, *Digital holography and phase retrieval*, Holography, Research and Technologies, InTech, 2011.

[ELB17] V. Elser, T.-Y. Lan, and T. Bendory, *Benchmark problems for phase retrieval*, arXiv preprint arXiv:1706.00399 (2017).

[EM12] Y. C. Eldar and S. Mendelson, *Phase retrieval: Stability and recovery guarantees*, CoRR **abs/1211.0872** (2012), 1211.0872.

[FD87] C. Fienup and J. Dainty, *Phase retrieval and image reconstruction for astronomy*, Image Recovery: Theory and Application **231** (1987), 275.

[Fie82] J. R. Fienup, *Phase retrieval algorithms: a comparison*, Appl. Opt. **21** (1982), no. 15, 2758–2769.

[FM16] M. P. Friedlander and I. Macedo, *Low-rank spectral optimization via gauge duality*, SIAM J. Scientific Computing **38** (2016), no. 3.

[FMP14] M. P. Friedlander, I. Macedo, and T. K. Pong, *Gauge optimization and duality*, SIAM Journal on Optimization **24** (2014), no. 4, 1999–2022.

[FR64] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, The computer journal **7** (1964), no. 2, 149–154.

[Fre87] R. M. Freund, *Dual gauge programs, with applications to quadratic programming and the minimum-norm problem*, Math. Program. **38** (1987), no. 1, 47–67.

[Gri91] R. D. Grigorieff, *A note on von neumann's trace inequalitv*, (1991).

[GS72] R. W. Gerchberg and W. O. Saxton, *A practical algorithm for the determination of phase from image and diffraction plane pictures*, Optik **35** (1972), 237–250.

[GS18a] T. Goldstein and C. Studer, *Phasemax: Convex phase retrieval via basis pursuit*, IEEE Transactions on Information Theory (2018).

[GS18b] M. Guizar-Sicairos, *Phase retrieval for x-ray coherent lensless imaging*, 2018.

[GVL12] G. H. Golub and C. F. Van Loan, *Matrix computations*, 4 ed., JHU Press, 2012.

[GY02] 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), no. 1, 312–334.

[Har93] R. W. Harrison, *Phase problem in crystallography*, JOSA a **10** (1993), no. 5, 1046–1055.

[HS52] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, vol. 49, NBS Washington, DC, 1952.

[JEH15] K. Jaganathan, Y. C. Eldar, and B. Hassibi, *Phase retrieval: An overview of recent developments*, CoRR **abs/1510.07713** (2015), 1–24, 1510.07713.

[JSL17] X. Jiang, H.-C. So, and X. Liu, *Robust phase retrieval via admm with outliers*, arXiv preprint arXiv:1702.06157 (2017).

[Kat17] V. Katkovnik, *Phase retrieval from noisy data based on sparse approximation of object phase and amplitude*, arXiv preprint arXiv:1709.01071 (2017).

[Kny87] A. V. Knyazev, *Convergence rate estimates for iterative methods for a mesh symmetric eigenvalue problem*, Russian Journal of Numerical Analysis and Mathematical Modelling **2** (1987), no. 5, 371–396.

[Li15] R.-C. Li, *Rayleigh quotient based optimization methods for eigenvalue problems*, Matrix Functions and Matrix Equations, World Scientific, 2015, pp. 76–108.

[LS84] A. Levi and H. Stark, *Image restoration by the method of generalized projections with application to restoration from magnitude*, IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '84, San Diego, California, USA, March 19-21, 1984, 1984, pp. 88–91.

[LSY98] 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.

[MCKS99] 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), no. 6742, 342.

[Mil90] R. P. Millane, *Phase retrieval in crystallography and optics*, JOSA A **7** (1990), no. 3, 394–411.

[MISE08] 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*, Annu. Rev. Phys. Chem. **59** (2008), 387–410.

[MWL+12] 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), no. 15, 16650–16661.

[MY05] J. H. Money and Q. Ye, *Algorithm 845: Eigifp: a matlab program for solving large symmetric generalized eigenvalue problems*, ACM Transactions on Mathematical Software (TOMS) **31** (2005), no. 2, 270–279.

[Nat95] B. K. Natarajan, *Sparse approximate solutions to linear systems*, SIAM journal on computing **24** (1995), no. 2, 227–234.

[NW06] J. Nocedal and S. J. Wright, *Numerical optimization 2nd*, Springer, 2006.

[PB+14] N. Parikh, S. Boyd, et al., *Proximal algorithms*, Foundations and Trends® in Optimization **1** (2014), no. 3, 127–239.

[PR69] E. Polak and G. Ribiere, *Note on convergence of conjugate direction meters*, French Review of Computer Science and Operational Research. Red **3** (1969), no. 16, 35–43.

[QY10] P. Quillen and Q. Ye, *A block inverse-free preconditioned krylov subspace method for symmetric generalized eigenvalue problems*, Journal of computational and applied mathematics **233** (2010), no. 5, 1298–1313.

[RFP10] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM review **52** (2010), no. 3, 471–501.

[Roc70] R. T. Rockafellar, *Convex analysis*, Princeton university press, 1970.

[RS80] M. Reed and B. Simon, *Functional analysis, vol. i*, Academic Press, San Diego, 1980.

[RXC+13] 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), no. 2, 312–318.

[Saa11] Y. Saad, *Numerical methods for large eigenvalue problems: revised edition*, vol. 66, Siam, 2011.

[SBE14] Y. Shechtman, A. Beck, and Y. C. Eldar, *Gespar: Efficient phase retrieval of sparse signals*, IEEE transactions on signal processing **62** (2014), no. 4, 928–938.

[SBFM09] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy, *Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm*, Artificial Intelligence and Statistics, 2009, pp. 456–463.

[Sch05] M. Schmidt, *minfunc: unconstrained differentiable multivariate optimization in matlab*, `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`, 2005.

[Sch08] ———, *minconf: projection methods for optimization with simple constraints in matlab*, `http://www.cs.ubc.ca/~schmidtm/Software/minConf.html`, 2008.

[SEC+15] 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 Process. Mag. **32** (2015), no. 3, 87–109.

[SESS11] Y. Shechtman, Y. C. Eldar, A. Szameit, and M. Segev, *Sparsity based sub-wavelength imaging with partially incoherent light via quadratic compressed sensing*, CoRR **abs/1104.4406** (2011), 1–16, 1104.4406.

[Sor92] D. C. Sorensen, *Implicit application of polynomial filters in ak-step arnoldi method*, Siam journal on matrix analysis and applications **13** (1992), no. 1, 357–385.

[Sor97] ———, *Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations*, (1997), 119–165.

[SQW16] J. Sun, Q. Qu, and J. Wright, *A geometric analysis of phase retrieval*, Information Theory (ISIT), 2016 IEEE International Symposium on, IEEE, 2016, pp. 2379–2383.

[TBI97] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50, Siam, 1997.

[TTT99] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, *Sdpt3a matlab software package for semidefinite programming, version 1.3*, Optimization methods and software **11** (1999), no. 1-4, 545–581.

[Wal63] A. Walther, *The question of phase retrieval in optics*, Optica Acta: International Journal of Optics **10** (1963), no. 1, 41–49.

[ZH04] H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM journal on Optimization **14** (2004), no. 4, 1043–1056.

[ZYLX17] H. Zhang, S. You, Z. Lin, and C. Xu, *Fast compressive phase retrieval under bounded noise.*, AAAI, 2017, pp. 2884–2890.