SIAM J. SCI. COMPUT.
Vol. 39, No. 5, pp. B840–B859

SOLVING PHASELIFT BY LOW-RANK RIEMANNIAN OPTIMIZATION METHODS FOR COMPLEX SEMIDEFINITE CONSTRAINTS*

WEN HUANG†, K. A. GALLIVAN‡, AND XIANGXIONG ZHANG§

Abstract. A framework, PhaseLift, was recently proposed to solve the phase retrieval problem. In this framework, the problem is solved by optimizing a cost function over the set of complex Hermitian positive semidefinite matrices. This approach to phase retrieval motivates a more general consideration of optimizing cost functions on semidefinite Hermitian matrices where the desired minimizers are known to have low rank. This paper considers an approach based on an alternative cost function defined on a union of appropriate manifolds. It is related to the original cost function in a manner that preserves the ability to find a global minimizer and is significantly more efficient computationally. A rank-based optimality condition for stationary points is given and optimization algorithms based on state-of-the-art Riemannian optimization and dynamically reducing rank are proposed. Empirical evaluations are performed using the PhaseLift problem. The new approach is shown to be an effective method of phase retrieval with computational efficiency increased substantially compared to a state-of-the-art algorithm, the Wirtinger flow algorithm. A preliminary version of this paper can be found in [W. Huang, K. A. Gallivan, and X. Zhang, *Procedia Comput. Sci.*, 80 (2016), pp. 1125–1134].

Key words. Riemannian optimization, low-rank optimization, complex optimization, phase retrieval, PhaseLift

AMS subject classifications. 65K05, 49N30, 49N45

DOI. 10.1137/16M1072838

1. Introduction.

Recovering a signal given the modulus of its transform, e.g., Fourier or wavelet transform, is an important task in the phase retrieval problem. It is a key problem for many important applications, e.g., X-ray crystallography imaging [Har93], diffraction imaging [BDP+07], optics [Wal63], and microscopy [MISE08].

The continuous form of the problem with the Fourier transform recovers $x(t) : \mathbb{R}^s \to \mathbb{C}$ from $|\tilde{x}(u)|$, where $\tilde{x}(u) : \mathbb{R}^s \to \mathbb{C}$ is defined by

$$\tilde{x}(u) = \int_{\mathbb{R}^s} x(t) \exp\left(-2\pi u \cdot t\sqrt{-1}\right) dt,$$

and $\cdot$ denotes the Euclidean inner product. This paper considers the discrete form of the problem where an indexed set of complex numbers $\mathbf{x} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_s}$ is to be recovered from the modulus of its discrete Fourier transform (DFT) $|\tilde{\mathbf{x}}(g_1, g_2, \ldots, g_s)|$,

*Submitted to the journal's Computational Methods in Science and Engineering section April 27, 2016; accepted for publication (in revised form) April 26, 2017; published electronically September 21, 2017.

http://www.siam.org/journals/sisc/39-5/M107283.html

Funding: This work was supported by grant FNRS PDR T.0173.13.

†Corresponding author. Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005 (huwst08@gmail.com).

‡Department of Mathematics, Florida State University, Tallahassee, FL 32306-4510 (kgallivan@fsu.edu).

§Department of Mathematics, Purdue University, West Lafayette, IN 47907-2067 (zhan1966@purdue.edu).

where $(g_1, g_2, \ldots, g_s) \in \Omega := G_1 \times G_2 \times \ldots G_s$ and Ω is a grid of an s-dimensional space. The DFT $\tilde{\mathbf{x}}$ is given by

$$\tilde{\mathbf{x}}(g_1, g_2, \ldots, g_s) = \frac{1}{\sqrt{n}} \sum_{i_1, i_2, \ldots, i_s} \mathbf{x}_{i_1 i_2 \ldots i_s} \exp\left(-2\pi \left(\frac{(i_1-1)g_1}{n_1} + \cdots + \frac{(i_s-1)g_s}{n_s}\right)\sqrt{-1}\right), \tag{1.1}$$

where $n = n_1 n_2 \cdots n_s$, i_j is an integer satisfying $1 \le i_j \le n_j$ for $j = 1, \ldots s$, $\mathbf{x}_{i_1 i_2 \cdots i_s}$ denotes the corresponding entry of $\mathbf{x}$, and $\tilde{\mathbf{x}}(g_1, g_2, \ldots, g_s)$ denotes the corresponding entry of $\tilde{\mathbf{x}}$.

It is well-known that the solution of the phase retrieval is not unique. For example, when Ω is the uniform grid, i.e., $G_i = \{0, 1, \ldots, n_i - 1\}, i = 1, 2, \ldots, s$, if $\mathbf{x}$ is a solution, then

1. $\mathbf{y} = c\mathbf{x}$ is a solution, where $c \in \mathbb{C}$ and $|c| = 1$;
2. $\mathbf{y} \in \mathbb{C}^{n_1 \times n_2 \times \cdots n_s}$ such that $\mathbf{y}_{i_1 i_2 \cdots i_s} = \mathbf{x}_{j_1 j_2 \cdots j_s}$ is a solution, where $j_k = i_k + a_k \mod n_k, k = 1, \ldots, s$, and $a_1, a_2, \ldots a_s$ are integers;
3. $\mathbf{y} \in \mathbb{C}^{n_1 \times n_2 \times \cdots n_s}$ such that $\mathbf{y}_{i_1, i_2, \ldots i_s} = \bar{\mathbf{x}}_{j_1 j_2 \ldots j_s}$ is a solution, where $j_k = -i_k \mod n_k, k = 1, \ldots, s$, and $\bar{\mathbf{x}}$ is the conjugate of $\mathbf{x}$.

These equivalent solutions are called trivial associates of $\mathbf{x}$ and infinitely many additional solutions may exist [San85].

Oversampling in the Fourier domain is a standard method to obtain a unique solution and it has been shown to almost always give a unique solution for multiple dimensional problems for real-valued and nonnegative signals [BS79, Hay82, San85]. Many algorithms based on alternating projection [GS72] have been developed to solve phase retrieval problem using the oversampling framework [Fie78, Fie82, Els03, Bla04, Mar07, CMWL07]. While these algorithms are efficient and effective in some problem settings, they may not perform well in other settings. For details on the capabilities and difficulties of these algorithms see [CESV13] and the references therein.

In recent years other frameworks, using multiple structured illuminations, or the mathematically equivalent construct of masks, combined with convex programming, have been proposed to recover the phase exactly, e.g., PhaseLift [CESV13] and PhaseCut [WDAM13]. It was later proved that a feasibility problem of two convex sets can be solved for PhaseLift in [CL13, DH14]. For the PhaseLift framework, three major results are of interest here. First, using a small number (related to s) of noiseless measurements of the modulus defined by certain carefully designed illuminations, the phase can be recovered exactly [CESV13]. Second, when these carefully designed measurements are not used, the phase can be recovered exactly with high probability using $O(n \log n)$ noiseless measurements of the modulus [CSV13]. This result is further improved in [CL13] that exact recovery is still possible using $O(n)$ noiseless measurements. Finally, the stability of recovering the phase using noisy measurements is shown in [CSV13, CL13].

For the PhaseCut framework, it is known that if the phase can be recovered using PhaseLift, then it can also be recovered by a modified version of PhaseCut and that the PhaseCut is at least as stable as the weak formulation of PhaseLift for noisy measurements [WDAM13]. The weak formulation is formally defined in [WDAM13, section 4.1]; however, the idea of a weak formulation is also given earlier in the proof of [CESV13, Theorem 2.1]. Empirically, PhaseCut is observed to be more stable in the situation of sparse sampling of the modulus.

The problems in both PhaseLift and PhaseCut concern optimizing convex cost functions defined on a convex set of complex matrices, i.e.,

$$\min_{X \in \mathcal{D}_n} H(X), \tag{1.2}$$

where $H : \mathcal{D}_n \to \mathbb{R} : X \mapsto H(X)$, and $\mathcal{D}_n$ denotes the set of all n-by-n complex Hermitian positive semidefinite matrices. PhaseCut further requires that the diagonal entries of X are 1. However, the dimension of (1.2) is usually too large to be solved by standard convex programming techniques. For example, in order to recover an image of 100 by 100 pixels, i.e., $s = 2$ and $n_1 = n_2 = 100$, solving an optimization problem with an argument that is a 100^2 by 100^2 matrix is required. The complexity of solving PhaseLift and PhaseCut using standard semidefinite programming solvers, e.g., SDPT3 [TTT99], is discussed in [WDAM13, section 4.6].

Since the desired optimum, X_*, is known to be a rank-one matrix, a low-rank matrix approximation of the argument matrix is used in [CESV13] to save computations for PhaseLift. While this approximation has good empirical performance, no convergence proof is given in [CESV13]. For PhaseCut, a block coordinate descent algorithm is proposed in [WDAM13] and the algorithm is shown to be computationally inexpensive for each iteration. However, the block coordinate descent algorithm converges slowly, i.e., linear convergence [BV04, section 9.4.3], and the overall computational cost can be unacceptably high.

This paper uses the framework of PhaseLift and an alternate cost function $F : \mathbb{C}^{n\times p} \to \mathbb{R} : Y \mapsto F(Y) = H(YY^*)$ defined by matrix factorization is considered. Even though F is not convex, it is shown to be a suitable replacement of the cost function H. Riemannian optimization methods on an appropriate quotient space are used for optimizing F. Using the cost function F with a small dimension p reduces storage and the computational complexity of each iteration. Fast convergence rate is also guaranteed theoretically by known Riemannian optimization results. This new approach is shown to perform empirically much better than the low-rank approximate version of the algorithm used for PhaseLift in [CESV13] and the Wirtinger flow algorithm in [CLS16] from the points of view of efficiency and effectiveness. Finally, note that the analysis and algorithm presented is not specific to the cost function used for phase retrieval in PhaseLift but for a general cost function defined on $\mathcal{D}_n$ and therefore the approach has potential for optimization in other applications where the global optimum is known to have low rank.

A forerunner to this paper appeared in [HGZ16b]. This paper differs from the conference paper in the following main aspects: (i) proofs of theoretical results and derivations of required ingredients for Riemannian optimization are given in this paper; (ii) a different Riemannian metric is used for the fixed-rank Hermitian positive semidefinite matrices and this metric yields a cheaper Riemannian gradient and isometric vector transport;[1] and (iii) the initial iterate in this paper is chosen by exploiting the approach in [CLS16, Algorithm 1], whereas [HGZ16b] uses only a random initial iterate. The proposed initial iterate has an edge over the previous one in the sense that it significantly reduces computational time, and (iv) this paper compares the proposed algorithm with the state-of-the-art algorithm, the Wirtinger flow algorithm [CLS16], which is not done in [HGZ16b].

The idea of using low-rank factorization to solve positive semidefinite constrained problems is, of course, not new, but all the research results of which the authors are aware are for real positive semidefinite matrix constraints. Burer and Monteiro [BM03] first investigated this approach for semidefinite programming in which the cost

[1] The Riemannian gradient in [HGZ16b] requires solving a Sylvester equation, whereas the Riemannian gradient in this paper has a cheaper closed form. The isometric vector transports in this paper has complexity $O(np^2) + O(p^3)$ which is cheaper than the one of $O(np^2) + O(p^6)$ in [HGZ16b].

function is linear. Journée et al. [JBAS10] use low-rank factorization for a more general problem in the sense that the cost function H is not necessary linear,

$$\min_{X\in\mathbb{S}_n^+} H(X), \quad \text{such that } \operatorname{tr}(A_iX) = b_i, i = 1,\dots m,$$

where $\mathbb{S}_n^+$ denotes the set of all real n-by-n symmetric positive semidefinite matrices, $A_i \in \mathbb{R}^{n\times n}$, $A_i = A_i^T$, and $A_iA_j = 0$ for any $i \neq j$. The conditions that $A_i = A_i^T$ and $A_iA_j = 0$ for any $i \neq j$ imply the number of equality constraints m is at most n as pointed out in [JBAS10]. The complex problem (1.2) does not belong to this category of problem since m is much larger than n when the complex problem is written as a real problem; see details in [HGZ16a, Appendix A].

The paper is organized as follows. Section 2 presents the notation used. The derivation of the optimization problem framework in PhaseLift is given in section 3. The alternate cost function and optimality conditions are derived in section 4. Riemannian optimization methods and the required geometric objects are presented in section 5. In section 6, the effectiveness of the methods is demonstrated with several numerical experiments, and, finally, conclusions are given in section 7.

2. Notation. For any $\mathbf{z} \in \mathbb{C}^{n_1\times n_2\times\cdots n_s}$, $\operatorname{vec}(\mathbf{z}) \in \mathbb{C}^n$, where $n = n_1n_2\cdots n_s$, denotes the vector form of $\mathbf{z}$, i.e., $(\operatorname{vec}(\mathbf{z}))_k = \mathbf{z}_{i_1i_2\cdots i_s}$, where $k = i_1+\sum_{j=1}^{s-1} n_1n_2\cdots n_j (i_{j+1}-1)$. $\operatorname{Re}(\cdot)$ denotes the real part of the argument and superscript $*$ denotes the conjugate transpose operator. Given a vector v with length h, $\operatorname{Diag}(v)$ denotes an h-by-h diagonal matrix the diagonal entries of which are v.

$0_{s\times k}$ denotes an $s\times k$ zero matrix; $I_{s\times k}$ denotes a diagonal matrix with diagonal entries 1; and 0_s denotes a vector with length s with entries all 0. $\operatorname{diag}(M)$ denotes a vector of the diagonal entries of $M \in \mathbb{C}^{s\times k}$ and $\operatorname{tr}(M)$ denotes the trace of M. If $s \geq k$, $M_\perp$ denotes an $s\times(s-k)$ matrix such that $M_\perp^*M_\perp = I_{(s-k)\times(s-k)}$ and $M_\perp^*M = 0_{(s-k)\times k}$. $M(:, 1:k)$ denotes a matrix that is formed by the first k columns of matrix M. $\operatorname{span}(M)$ denotes the column space of M.

Given a manifold $\mathcal{M}$, $\mathrm{T}_x\mathcal{M}$ denotes the tangent space of $\mathcal{M}$ at $x \in \mathcal{M}$. $\mathcal{D}_k$ denotes set $\{X \in \mathbb{C}^{n\times n} | X = X^*, X \succeq 0, \operatorname{rank}(X) \leq k\}, 1 \leq k \leq n$, where the statement $X \succeq 0$ means that matrix X is positive semidefinite or definite. $\operatorname{St}(k, s)$ denotes the complex compact Stiefel manifold $\{A \in \mathbb{C}^{s\times k} | A^*A = I_{k\times k}\}$ with $s \geq k$. $\mathrm{S}_+^{\mathbb{C}}(k, s)$ denotes the set of all Hermitian positive semidefinite $s\times s$ matrices of fixed rank k. When elements of $\mathrm{S}_+^{\mathbb{C}}(k, s)$ are restricted to be real, it is denoted by $\mathrm{S}_+^{\mathbb{R}}(k, s)$. $\mathbb{C}_*^{s\times k}$ denotes the complex noncompact Stiefel manifold, i.e., the set of all $s\times k$ full column rank complex matrices. $\mathcal{O}_s$ denotes the group of s-by-s unitary matrices.

Given a function $f(x)$ on $\mathcal{M}$ or $\mathbb{C}^{s\times k}$, $\operatorname{grad} f(x)$ denotes the gradient of f at x.

3. The PhaseLift approach to phase retrieval. The phase retrieval problem recovers $\mathbf{x}$ from its quadratic measurements of the form $\mathbb{A}(\mathbf{x}) = \{|\langle \mathbf{a}_k, \mathbf{x}\rangle|^2 : k = 1, 2, \dots, m\}$, where $\mathbf{a}_k \in \mathbb{C}^{n_1\times n_2\times\cdots n_s}, k = 1, 2, \dots, m$, are given. It is well-known that the quadratic measurements can be lifted up to be linear measurements about the rank-one matrix $X = xx^*$, where $x = \operatorname{vec}(\mathbf{x}) \in \mathbb{C}^n$. Specifically, the measurements are $|\langle \mathbf{a}_k, \mathbf{x}\rangle|^2 = \operatorname{tr}(a_ka_k^*xx^*) := \operatorname{tr}(A_kX)$, where $a_k = \operatorname{vec}(\mathbf{a}_k) \in \mathbb{C}^n$. Define $\mathcal{A}$ to be the linear operator mapping X into $b := \begin{bmatrix}\operatorname{tr}(A_1X) & \operatorname{tr}(A_2X) & \cdots & \operatorname{tr}(A_mX)\end{bmatrix}^T$. The goal of the phase retrieval problem is to

$$\text{find } X \quad \text{such that } \mathcal{A}(X) = b, X \succeq 0 \text{ and } \operatorname{rank}(X) = 1. \tag{3.1}$$

The alternative problem suggested in [CESV13] considers an optimization problem that does not force the rank of matrix to be one but adds a nuclear norm penalty term to favor low-rank solutions,

$$\min_{X \in \mathcal{D}_n} \|b - \mathcal{A}(X)\|_2^2 + \kappa \operatorname{tr}(X), \tag{3.2}$$

where κ is a positive constant.

Measurements with noise, $b \in \mathbb{R}^m$, are assumed to have the form $b = \mathcal{A}(X) + \varepsilon$, where $\varepsilon \in \mathbb{R}^m$ is noise sampled from a distribution $p(:;\mu)$. The task suggested in [CESV13] is

$$\begin{aligned} &\min_X -\log(p(b;\mu)) + \kappa \operatorname{tr}(X) \\ &\text{such that } \mu = \mathcal{A}(X) \text{ and } X \in \mathcal{D}_n \end{aligned} \tag{3.3}$$

or equivalently

$$\min_{X \in \mathcal{D}_n} -\log(p(b;\mathcal{A}(X))) + \kappa \operatorname{tr}(X), \tag{3.4}$$

where κ is a positive constant. Problems (3.3) and (3.4) are preferred over problem (3.1), since they are convex programming problems when the log-likelihood function is concave.

4. Theoretical results. This section presents theoretical results that motivate the design of algorithms for optimizing a cost function H defined on $\mathcal{D}_n$. The analysis does not rely on the convexity of the cost function H.

4.1. Equivalent cost function. The cost functions generically denoted H all satisfy

$$H : \mathcal{D}_n \to \mathbb{R} : X \mapsto H(X). \tag{4.1}$$

It is well-known that for any $X \in \mathcal{D}_n$, there exists $Y_n \in \mathbb{C}^{n\times n}$ such that $Y_n Y_n^* = X$. Furthermore, if X is rank p, then there exists $Y_p \in \mathbb{C}^{n\times p}$ such that $Y_p Y_p^* = X$. Throughout this paper, the subscript p is used to emphasize the column size of Y. Therefore, a surjective mapping between $\mathbb{C}^{n\times p}$ and $\mathcal{D}_p$ is given by $\alpha_p : \mathbb{C}^{n\times p} \to \mathcal{D}_p : Y_p \mapsto Y_p Y_p^*$. It is clear that α_p is not an injection. Specifically, given $X \in \mathcal{D}_p$, if Y_p satisfies $\alpha_p(Y_p) = Y_p Y_p^* = X$, then $Y_p O_p$ also satisfies $\alpha_p(Y_p O_p) = X$ for any $O_p \in \mathcal{O}_p$. Thus, if the desired solution of H is known to be at most rank p, then an alternate cost function to H can be used:

$$F_p : \mathbb{C}^{n\times p} \to \mathbb{R} : Y_p \mapsto H(\alpha_p(Y_p)) = H(Y_p Y_p^*).$$

The subscripts of F and α indicate the column size of the argument. The domain of F_p has lower dimension than that of H which may yield computational efficiency. Therefore, instead of problem of (1.2), the problem $\min_{Y_p \in \mathbb{C}^{n\times p}} F_p(Y_p)$ is considered.

4.2. Optimality conditions. In this section, the characterizations of stationary points of F and H over $\mathcal{D}_n$ are used to derive the relationship between optimizing F and optimizing H over $\mathcal{D}_n$. Since H is defined on a constrained set, a stationary point of H does not simply satisfy $\operatorname{grad} H(X) = 0$. One can define the stationary points of H as follows by [HGZ16a, Lemma A.1].

DEFINITION 4.1. *A stationary point of* (4.1) *is a matrix* $X \in \mathbb{D}_n$ *such that* $\operatorname{grad} H(X)X = 0$ *and* $\operatorname{grad} H(X) \succeq 0$.

The gradient is easily computed and is given in Lemma 4.2 in terms of H.

LEMMA 4.2. *The gradient of F_p at Y_p is given by*

$$\operatorname{grad} F_p(Y_p) = 2 \operatorname{grad} H(Y_p Y_p^*) Y_p. \tag{4.2}$$

Proof. On one hand, it satisfies that for all $\eta_p \in \mathbb{C}^{n\times p}$

$$\mathrm{D}\, F_p(Y_p)[\eta_p] = g^E(\operatorname{grad} F_p(Y_p), \eta_p).$$

On the other hand, we have

$$\begin{aligned}\mathrm{D}\, F_p(Y_p)[\eta_p] &= \mathrm{D}\, H(Y_p Y_p^*)[Y_p \eta_p^* + \eta_p Y_p^*] = g^E(\operatorname{grad} H(Y_p Y_p^*), Y_p \eta_p^* + \eta_p Y_p^*) \\ &= g^E((\operatorname{grad} H(Y_p Y_p^*) + \operatorname{grad} H(Y_p Y_p^*)^*) Y_p, \eta_p),\end{aligned}$$

which implies $\operatorname{grad} F_p(Y_p) = (\operatorname{grad} H(Y_p Y_p^*) + \operatorname{grad} H(Y_p Y_p^*)^*) Y_p$. Since H is defined on Hermitian matrices, $\operatorname{grad} H$ can be written as a Hermitian matrix. It follows that $\operatorname{grad} F_p(Y_p) = 2 \operatorname{grad} H(Y_p Y_p^*) Y_p$, which is (4.2). ∎

Theorem 4.3 and [JBAS10, Theorem 7] show similar results under different frameworks. Both results suggest considering the cost function F_p if the desired minimizer of H is known to have rank smaller than p, as is the case with PhaseLift for phase retrieval. This is formalized in Theorem 4.3 and has critical algorithmic, efficiency, and optimality implications when H has suitable structure such as convexity as in the case of PhaseLift. These implications for PhaseLIft are discussed in section 6.1.

THEOREM 4.3. *Suppose $Y_p = K_s Q^*$ is a rank deficient minimizer of F_p, where $K_s \in \mathcal{C}_*^{n\times s}$ with $s < p$ and $Q \in \mathrm{St}(s,p)$. Then $\operatorname{grad} H(Y_p Y_p^*)$ is a positive semidefinite matrix and, therefore, $X = Y_p Y_p^*$ is a stationary point of H. If furthermore H is convex, then X is a global minimizer of* (4.1).

Proof. We first show that $(K_s)_\perp^* \operatorname{grad} H(X)(K_s)_\perp$ is a positive semidefinite matrix. This is proved by contradiction. If $(K_s)_\perp^* \operatorname{grad} H(X)(K_s)_\perp$ is not a positive semidefinite matrix, then it has at least one negative eigenvalue. If μ and v denote a negative eigenvalue and the corresponding eigenvector, then the semidefinite positive matrix $\eta = -(K_s)_\perp (v\mu v^*)(K_s)_\perp^*$ satisfies $g^E(\eta, \operatorname{grad} H(X)) < 0$. Thus, a smooth curve $\gamma(t) = X + t\eta$ satisfies that $\dot\gamma(0) = \eta$, $\gamma(t) \in \mathcal{D}_p$ for all $t \in [0, \delta)$ and $\gamma(0) = X$, where δ is a positive constant. The derivative $\frac{d}{dt} H(\gamma(t))|_{t=0}$ by definition is $g^E(\eta, \operatorname{grad} H(X))$ and, therefore, $\frac{d}{dt} H(\gamma(t))|_{t=0} < 0$.

Decomposing $\gamma(t)$ yields

$$\gamma(t) = \begin{pmatrix} K_s & \tilde v \end{pmatrix} \begin{pmatrix} I_{s\times s} & \\ & -t\mu \end{pmatrix} \begin{pmatrix} K_s & \tilde v \end{pmatrix}^*,$$

where $\tilde v = (K_s)_\perp v$. Define $r(t) = \begin{pmatrix} K_s & \tilde v \end{pmatrix} \operatorname{Diag}(I_s, \sqrt{-t\mu})$. Therefore, $\gamma(t) = r(t) r^*(t)$. The derivative of $r(t^2)$ is $\xi(t) = \begin{pmatrix} 0_{n\times s} & \sqrt{-\mu}\tilde v \end{pmatrix}$. It follows that

$$\frac{d^2}{dt^2} H(\gamma(t^2))|_{t=0} = \frac{d^2}{dt^2} F_n(r(t^2))|_{t=0} = g^E(\xi(0), \operatorname{Hess} F_p(\tilde Y_p)[\xi(0)]) \geq 0. \tag{4.3}$$

Let $a(t) = H(\gamma(t))$ and so $\dot a(0) < 0$. It follows that

$$\frac{d^2}{dt^2} H(\gamma(t^2))|_{t=0} = \frac{d^2}{dt^2} a(t^2) = (4t^2 \ddot a(t^2) + 2\dot a(t^2))|_{t=0} = 2\dot a(0) < 0,$$

which conflicts with (4.3). Therefore, $(K_s)_\perp^* \operatorname{grad} H(Y_pY_p^*)(K_s)_\perp$ is a positive semidefinite matrix which is a contradiction with the initial assumption of the proof.

Let Q_s denote an orthonormal basis of $\operatorname{span}(K_s)$. $\operatorname{grad} H(X)$ can be written as

$$\operatorname{grad} H(X) = \begin{pmatrix} Q_s & (K_s)_\perp \end{pmatrix} \begin{pmatrix} S & A^* \\ A & R \end{pmatrix} \begin{pmatrix} Q_s & (K_s)_\perp \end{pmatrix}^*,$$

where $S \in \mathbb{C}^{s\times s}$, $S^* = S$, $A \in \mathbb{C}^{(n-s)\times p}$, and $R = (K_s)_\perp^* \operatorname{grad} H(X)(K_s)_\perp$. Since Y_p is a stationary point of F, we have $\operatorname{grad} H(X)K_sQ^* = \operatorname{grad} F(Y_p) = 0_{n\times s}$. It follows that $S = 0_{s\times s}$ and $A = 0_{(n-s)\times s}$. Therefore, $R \succeq 0$ implies $\operatorname{grad} H(X) \succeq 0$, which means X is a stationary point of H by Definition 4.1. ∎

For the optimization problems in the PhaseLift framework for phase retrieval, Theorem 4.3 is important due to the following reasons. First, the cost function H in PhaseLift is convex over a convex domain $\mathcal{D}_n$. Therefore, finding a stationary point of H by using the cost function F is sufficient to find a global minimizer of H. Second, the rank of the desired minimizer of H in PhaseLift is one. It follows that by using a low-rank factorization-based cost function F_p with small $p > 1$ it is possible to find the desired unique rank-one minimizer of H by optimizing F_p with small $p > 1$. (This approach also has lower storage and computational complexity compared to optimizing H.) The theorem guarantees that any minimizer, Y_p, of F_p with rank less than p must have rank 1 and $Y_pY_p^*$ must be the global minimizer of H.

5. A Riemannian approach. Riemannian optimization is an active research area and recently many Riemannian optimization methods have been systemically analyzed and efficient libraries designed, e.g., the Riemannian trust-region Newton method (RTR-Newton) [Bak08], the Riemannian Broyden family method including the BFGS method and its limited-memory version (RBroyden family, RBFGS, LRBFGS) [RW12, Hua13, HGA15], the Riemannian trust-region symmetric rank-one update method and its limited-memory version (RTR-SR1, LRTR-SR1) [Hua13, HAG15], the Riemannian Newton method (RNewton), and the Riemannian nonlinear conjugate gradient method (RCG) [AMS08, SI15, Sat15].

Journée et al. [JBAS10] have proposed a method that combines a Riemannian optimization method on a fixed-rank manifold with a procedure for increasing rank for their semidefinite constrained problem setting. Specifically, given an iterate with rank r, a Riemannian optimization method is applied for a cost function on a manifold with rank r. If the limit point is not rank deficient, then either a descent direction to a higher rank space can be found or a desired stationary point is obtained. For the former case, a descent algorithm is applied to find a next descent iterate which is used to be the initial point for a Riemannian optimization method on a manifold with larger rank. For the latter case, the convergence rate can be obtained and depends on the Riemannian optimization algorithm. If the limit point is rank deficient, then convergence analyses are complicated and may need to consider a union of manifolds [ZHG+15]. This case was ignored in [JBAS10] since situations of a limit point being rank deficient were not encountered in their experiments.

If the rank of the desired minimizer is known, such as in the problems in PhaseLift, then [BM03] suggests choosing the rank of initial point to be that rank. However, this, in fact, is not the appropriate response due to complexity considerations as the theory and algorithms derived in this section indicate and the numerical experiments in section 6 demonstrate. To see this, let r_* denote the desired rank of the global minimizer. Note that there may exist a stationary point Y_{r_*} of F_{r_*} for which $Y_{r_*}Y_{r_*}^*$ is not a stationary point of H. It follows that forcing iterates to be rank r_* is not

appropriate and starting from a higher rank or moving to a higher rank to move to the minimizer of H is necessary.

There is also a potential problem of using a rank increasing procedure. If Y_p is a stationary point of F_p, then $(Y_p, 0_{n\times(k-p)})$ is also a stationary point of F_k. A procedure that increases rank starting from Y_p may find a point Y_k which is close to $(Y_p, 0_{n\times(k-p)})$. It follows that using Y_k to be an initial point of the iteration on the rank-k manifold may not work efficiently since Y_k may be too close to a stationary point. Therefore, an algorithm based on Riemannian optimization methods on a fixed-rank manifold and a procedure to decrease rank without using any rank increase technique is proposed in this section.

5.1. Riemannian optimization on fixed-rank manifold. In order to make use of Riemannian optimization theory and algorithms on a fixed-rank manifold, the Riemannian gradient of the cost function, the tangent space of an element in the manifold, the retraction operation on the manifold, and an appropriate vector transport are needed. The definitions of Riemannian gradient, tangent space, retraction, and vector transport are standard and can be found, e.g., in [Boo86, AMS08].

Derivations for Riemannian objects of $\mathrm{S}_+^{\mathbb{R}}(p,n)$ have been given in [AIDV09]. This section includes derivations of Riemannian objects for the complex case, i.e., $\mathrm{S}_+^{\mathbb{C}}(p,n)$. Since the mapping α_p is not an injection, all the minimizers of F_p are degenerate, which causes difficulties in some algorithms, e.g., Riemannian and Euclidean Newton methods. In order to overcome this difficulty, a function defined on a quotient manifold with fixed rank is considered. To this end, define the mapping β_p to be the mapping α_p restricted on $\mathbb{C}_*^{n\times p}$, i.e., $\beta_p : \mathbb{C}_*^{n\times p} \to \mathrm{S}_+^{\mathbb{C}}(p,n) : Y \mapsto \alpha_p(Y) = YY^*$, and function G_p to be the function F_p restricted on $\mathbb{C}_*^{n\times p}$, i.e., $G_p : \mathbb{C}_*^{n\times p} \to \mathbb{R} : Y \mapsto F_p(Y) = H(\beta_p(Y))$. Like α_p, the mapping β_p is a surjection but not an injection and there are multiple matrices in $\mathbb{C}_*^{n\times p}$ mapping to a single point in $\mathrm{S}_+^{\mathbb{C}}(p,n)$. Nevertheless, given an $X \in \mathrm{S}_+^{\mathbb{C}}(p,n)$, $\beta_p^{-1}(X)$ is a manifold, while $\alpha_p^{-1}(X)$ is not a manifold. Therefore, using the mapping β_p, a quotient manifold can be used to remove the degeneracy by defining the equivalence class $\beta_p^{-1}(YY^*) = [Y] = \{YO | O \in \mathcal{O}_p\}$ and the set

$$\mathbb{C}_*^{n\times p}/\mathcal{O}_p = \{[Y] | Y \in \mathbb{C}_*^{n\times p}\}.$$

This set can be shown to be a quotient manifold over $\mathbb{R}$. To clarify the notation, $\pi(Y)$ is used to denote $[Y]$ viewed as an element in $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$ and $\pi^{-1}(\pi(Y))$ is used to denote $[Y]$ viewed as a subset of $\mathbb{C}_*^{n\times p}$. The function $m_p : \pi(Y) \mapsto YY^*$ is a diffeomorphism between $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$ and $\mathrm{S}_+^{\mathbb{C}}(p,n)$.

An element of a quotient manifold is an equivalence class which is often cumbersome computationally . Fortunately, choosing a representative for an equivalence class and definitions of related mathematical objects have been developed in many papers in the literature of computation on manifolds, e.g., [AMS08]. The vertical space at $Y \in \pi^{-1}(\pi(Y))$, which is the tangent space of $\pi^{-1}(\pi(Y))$ at Y, is

$$\mathcal{V}_Y = \{Y\Omega | \Omega^* = -\Omega, \Omega \in \mathbb{C}^{p\times p}\}.$$

The horizontal space at Y, $\mathcal{H}_Y$, is defined to be a subspace of $\mathrm{T}_Y\, \mathbb{C}_*^{n\times p} = \mathbb{C}^{n\times p}$ that is orthogonal to $\mathcal{V}_Y$, i.e., satisfying $\mathcal{H}_A \oplus \mathcal{V}_A = \mathrm{T}_A\, \mathrm{GL}(n,\mathbb{C})$. Therefore, a Riemannian metric of $\mathbb{C}_*^{n\times p}$ is required to define the meaning of orthogonal. The metric used is

$$\hat{g}_Y(\eta_Y, \xi_Y) = \mathrm{Re}(\mathrm{tr}((Y^*Y)\eta_Y^*\xi_Y)) \tag{5.1}$$

for all $\eta_Y, \xi_Y \in \mathrm{T}_Y \mathbb{C}_*^{n\times p}$ and $Y \in \mathbb{C}_*^{n\times p}$. The metric (5.1) yields a cheap vector transport by parallelization which is discussed later. The horizontal space is therefore

$$\mathcal{H}_Y = \{YS + Y_\perp K | S^* = S, S \in \mathbb{C}^{p\times p}, K \in \mathbb{C}^{(n-p)\times p}\}.$$

The horizontal space $\mathcal{H}_Y$ is a representation of the tangent space $\mathrm{T}_{\pi(Y)} \mathbb{C}_*^{n\times p}/\mathcal{O}_p$. It is known that for any $\eta_{\pi(Y)} \in \mathrm{T}_{\pi(Y)} \mathbb{C}_*^{n\times p}/\mathcal{O}_p$, there exists a unique vector in $\mathcal{H}_Y$, called the horizontal lift of $\eta_{\pi(Y)}$ and denoted by $\eta_{\uparrow_Y}$, satisfying $\mathrm{D}\,\pi(Y)[\eta_{\uparrow_Y}] = \eta_{\pi(Y)}$; see e.g., [AMS08]. Lemma 5.1 gives a relationship among horizontal lifts of a tangent vector $\eta_{\pi(Y)}$ when different representations in $\pi^{-1}(\pi(Y))$ are chosen. The result follows from [Hua13, Theorem 9.3.1].

LEMMA 5.1. *A horizontal vector field $\hat{\eta}$ of $\mathbb{C}_*^{n\times p}$ is the horizontal lift of a vector field η on $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$ if and only if, for each $Y \in \mathbb{C}_*^{n\times p}$, we have $\hat{\eta}_{YO} = \hat{\eta}_Y O$ for all $O \in \mathcal{O}_p$.*

The orthogonal projections on to the horizontal space or the vertical space are also easily characterized.

LEMMA 5.2. *The orthogonal projection to vertical space $\mathcal{V}_Y$ is $P_Y^v(\eta) = Y\Omega$, where $\Omega = ((Y^*Y)^{-1}Y^*\eta - \eta^*Y(Y^*Y)^{-1})/2$ is a skew Hermitian matrix. The orthogonal projection to Horizontal space $\mathcal{H}_Y$ is $P_Y^h(\eta) = \eta - Y\Omega$.*

Proof. By definition of $\mathcal{H}_Y$ and $\mathcal{V}_Y$, $P_Y^h(\eta)$ satisfies that $(Y^*Y)^{-1}Y^*P_Y^h(\eta) = P_Y^h(\eta)^*Y(Y^*Y)^{-1}$ and can be expressed as $\eta - Y\Omega$. It follows that $\Omega = ((Y^*Y)^{-1}Y^*\eta - \eta^*Y(Y^*Y)^{-1})/2$, which gives the desired results. ∎

Finally, the desired cost function that removes the equivalence can be defined as

$$f_p : \mathbb{C}_*^{n\times p}/\mathcal{O}_p \to \mathbb{R} : \pi(Y) \mapsto f_p(\pi(Y)) = G_p(Y) = F_p(Y). \tag{5.2}$$

The function f_p in (5.2) has the important property that $\pi(Y)$ is a nondegenerate minimizer of f over $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$ if and only if YY^* is a nondegenerate minimizer of H over $\mathrm{S}_+^{\mathbb{C}}(p, n)$.

The gradient is given in Lemma 5.3.

LEMMA 5.3. *The horizontal lift of the gradient of* (5.2) *at Y is*

$$(\operatorname{grad} f(\pi(Y)))_{\uparrow_Y} = P_Y^h(\operatorname{grad} F(Y)(Y^*Y)^{-1}).$$

Proof. The directional derivative of f along any $\eta_{\pi(Y)} \in \mathrm{T}_{\pi(Y)} \mathbb{C}_*^{n\times p}/\mathcal{O}_p$ is

$$\begin{aligned}
&\mathrm{D}\, f(\pi(Y))[\eta_{\pi(Y)}] = \mathrm{D}\, f(\pi(Y))[\mathrm{D}\,\pi(Y)[\eta_{\uparrow_Y}]] = \mathrm{D}\, F(Y)[\eta_{\uparrow_Y}] \\
&= \operatorname{Re}(\operatorname{tr}(\operatorname{grad} F(Y)^*\eta_{\uparrow_Y})) = \hat{g}_Y(\operatorname{grad} F(Y)(Y^*Y)^{-1}, \eta_{\uparrow_Y}) \\
&= \hat{g}_Y(P_Y^h(\operatorname{grad} F(Y)(Y^*Y)^{-1}), \eta_{\uparrow_Y}).
\end{aligned}$$

Additionally using the definition of gradient [AMS08, (3.31)], i.e., $\mathrm{D}\, f(\pi(Y))[\eta_{\pi(Y)}] = g_{\pi(Y)}(\operatorname{grad} f(\pi(Y)), \eta_{\pi(Y)})$, and the equation $g_{\pi(Y)}(\operatorname{grad} f(\pi(Y)), \eta_{\pi(Y)}) = \hat{g}_Y((\operatorname{grad} f(\pi(Y)))_{\uparrow_Y}, \eta_{\uparrow_Y})$, yields the result. ∎

Retraction is used in updating iterates in a Riemannian algorithm. Vector transport is used to compare tangent vectors in different tangent spaces. Specifically, a retraction R is a smooth mapping from the tangent bundle $\mathrm{T}\,\mathcal{M}$, which is the set of all tangent spaces, onto $\mathcal{M}$ such that (i) $R(0_x) = x$ for all $x \in \mathcal{M}$ (where 0_x denotes the origin of $\mathrm{T}_x\,\mathcal{M}$) and (ii) $\frac{d}{dt}R(t\xi_x)|_{t=0} = \xi_x$ for all $\xi_x \in \mathrm{T}_x\,\mathcal{M}$. The restriction of R to

$\mathrm{T}_x\,\mathcal{M}$ is denoted by R_x. A vector transport $\mathcal{T} : \mathrm{T}\,\mathcal{M} \oplus \mathrm{T}\,\mathcal{M} \to \mathrm{T}\,\mathcal{M}, (\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}\xi_x$ with associated retraction R is a smooth mapping such that, for all (x, η_x) in the domain of R and all $\xi_x \in \mathrm{T}_x\,\mathcal{M}$, it holds that (i) $\mathcal{T}_{\eta_x}\xi_x \in \mathrm{T}_{R(\eta_x)}\,\mathcal{M}$, (ii) $\mathcal{T}_{0_x}\xi_x = \xi_x$, and (iii) $\mathcal{T}_{\eta_x}$ is a linear map. The retraction used in the Riemannian optimization methods is

$$R_{\pi(Y)}(\eta_{\pi(Y)}) = \pi(Y + \eta_{\uparrow_Y}), \tag{5.3}$$

and the vector transport used is the vector transport by parallelization [HAG16b]:

$$\mathcal{T}_{\eta_x}\xi_x = B_y B_x^{\dagger},$$

where B is a smooth tangent basis field defined on an open set $\mathcal{V}$ of $\mathcal{M}$ and $B_x^{\dagger}$ denotes the pseudoinverse of B_x. A smooth orthonormal tangent basis of $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$ can be defined as follows: given $\pi(Z) \in \mathbb{C}_*^{n\times p}/\mathcal{O}_p$, the horizontal lifts of columns in $B_{\pi(Z)}$ at Z is

$$\begin{aligned}
&\left\{ZL^{-*}e_i e_i^T L^{-1}, i = 1, \ldots, p\right\}\\
&\bigcup\left\{\frac{1}{\sqrt{2}}ZL^{-*}(e_i e_j^T - e_j e_i^T)L^{-1}, i = 1, \ldots, p, j = i+1, \ldots, p\right\}\\
&\bigcup\left\{\frac{1}{\sqrt{2}}ZL^{-*}(e_i e_j^T + e_j e_i^T)\sqrt{-1}L^{-1} i = 1, \ldots, p, j = i+1, \ldots, p\right\}\\
&\bigcup\left\{Z_\perp \tilde{e}_i e_j^T L^{-1}, i = 1, \ldots n-p, j = 1, \ldots p\right\}\\
&\bigcup\left\{Z_\perp \tilde{e}_i e_j^T \sqrt{-1}L^{-1}, i = 1, \ldots n-p, j = 1, \ldots p\right\},
\end{aligned}$$

where $(e_1, \ldots, e_p)$ is the canonical basis of $\mathbb{R}^p$, $(\tilde{e}_1, \ldots, \tilde{e}_{(n-p)})$ is the canonical basis of $\mathbb{R}^{n-r}$, and $Z^*Z = LL^*$ is the Cholesky decomposition.

In summary, this section provides the objects used in Riemannian optimization methods, i.e., the horizontal space, the projection to a horizontal space, the Riemannian metric, the retraction, the vector transport, and the Riemannian gradient.

5.2. Dynamic rank reduction. Since the domain of f_p, $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$, is not closed, i.e., a sequence $\{W^{(i)}\}$ representing $\{\pi(W^{(i)})\}$ generated by an algorithm may have a limit point $\hat{W}$ with rank less than p, a simple well-known strategy for dynamically reducing rank is adapted and used. Since it is impossible in practice to check whether a limit point of iterates $\{W^{(i)}\}$ is a lower-rank matrix or just close to one of lower rank, the idea suggested below makes more sense when the desired rank of the minimizer is known and the current iterate $W^{(i)}$ has a higher rank than the desired rank. This is the case with PhaseLift for phase retrieval.

The thin singular value decomposition of the ith iterate is $W^{(i)} = U\Sigma V^*$ and $\Sigma = \mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)$, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. Let $\tilde{\sigma}$ be $\|\mathrm{Diag}(\sigma_1, \ldots, \sigma_p)\|_F/\sqrt{p}$. If there exists $q < p$ such that $\sigma_q/\tilde{\sigma} > \delta$ and $\sigma_{q+1}/\tilde{\sigma} \leq \delta$ for a given threshold δ, then $\hat{W} = U(:, 1:q)\,\mathrm{Diag}(\sigma_1, \ldots \sigma_q)V(:, 1:q)^*$ is chosen to be the initial point for optimizing cost function f_q over $\mathbb{C}_*^{n\times q}/\mathcal{O}_q$. The details of reducing rank are given in Algorithm 1. Note that the step of decreasing the rank may produce an iterate that increases the cost function value. This facilitates global optimization by allowing nondescent steps.

Combining a Riemannian optimization method with the procedure of reducing rank gives Algorithm 2.

Recently, a rigorous definition of a rank adaptation strategy, which not only reduces rank but also increases rank if necessary, for optimization with rank inequality

Algorithm 1. Reduce rank.

Require: $Y \in \mathbb{C}^{n\times p}$; threshold δ;
Ensure: $W \in \mathbb{C}^{n\times q}$;
Take thin singular value decomposition for Y, i.e., $Y = U\,\mathrm{Diag}(\sigma_1,\dots,\sigma_p)V^*$, where $U \in \mathbb{C}^{n\times p}$, $V \in \mathbb{C}^{p\times p}$ and $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$;
Set $\tilde{\sigma} = \|\,\mathrm{Diag}(\sigma_1,\dots,\sigma_p)\|_F/\sqrt{p}$;
if $\sigma_p/\tilde{\sigma} > \delta$ **then**
    $q \leftarrow p$, $W \leftarrow Y$ and return;
else
    Find q such that $\sigma_q/\tilde{\sigma} > \delta$ and $\sigma_{q+1}/\tilde{\sigma} \leq \delta$;
    Let $W = U(:, 1:q)\,\mathrm{Diag}(\sigma_1,\dots\sigma_q)V(:, 1:q)^*$ and return;
end if

Algorithm 2. Rank reduce algorithm.

Require: $p > 0$; $Y_p^{(0)} \in \mathbb{C}^{n\times p}$ a representation of initial point $\pi(Y_p^{(0)})$ for f; stopping criterion threshold ϵ; rank reducing threshold δ; a Riemannian optimization method;
Ensure: W
for $k = 0, 1, 2, \dots$ **do**
    Apply Riemannian method for cost function f over $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$ with initial point $\pi(Y_p^{(k)})$ until ith iterate $W^{(i)}$ satisfying $g(\mathrm{grad}\, f, \mathrm{grad}\, f) < \epsilon^2$ or the requirement of reducing rank with threshold δ;
    if $g(\mathrm{grad}\, f, \mathrm{grad}\, f) < \epsilon^2$ **then**
        Set $W \leftarrow W^{(i)}$ and return;
    else[iterate in the Riemannian optimization method meets the requirements of reducing rank]
        Apply Algorithm 1 with threshold δ and obtain an output $\hat{W} \in \mathbb{C}^{n\times q}$;
        $p \leftarrow q$ and set $Y_p^{(k+1)} = \hat{W}$;
    end if
end for

constraints based on the notion of rank-related Riemannian retractions has been developed in [ZHG+15]. It is pointed out here that one can exploit the idea in [ZHG+15] and similarly define a rank adaptation algorithm to optimize F.

6. Experiments. In this section, numerical simulations for noiseless problems and those with Gaussian noise are used to illustrate the performance of the proposed method. The required Riemannian objects are derived in section 6.1 and the experimental environment and parameters are defined in section 6.3. The detailed implementations are given in section 6.4. Algorithm 2 with LRBFGS is compared for a range of parameters in section 6.5. In section 6.6, the Riemannian approach is compared to the Wirtinger flow algorithm in [CLS16] that represents the current state-of-the-art algorithm. Finally, the performance is evaluated for various sizes of natural images.

6.1. Cost function, gradient, and complexity for PhaseLift. The known random masks or illumination fields defined on the discrete signal domain are denoted $\mathbf{w}_r \in \mathbb{C}^{n_1\times n_2\times\cdots n_s}$, $r = 1,\dots l$. It follows that $\{\langle \mathbf{a}_k, \mathbf{x}\rangle, k = 1,\dots m\}$ is

$$\begin{pmatrix} (\mathcal{F}_{n_s} \otimes \mathcal{F}_{n_{s-1}} \otimes \cdots \mathcal{F}_{n_1}) \operatorname{Diag}(w_1)x \\ \vdots \\ (\mathcal{F}_{n_s} \otimes \mathcal{F}_{n_{s-1}} \otimes \cdots \mathcal{F}_{n_1}) \operatorname{Diag}(w_l)x \end{pmatrix},$$

where $\otimes$ denotes the Kronecker product and $\mathcal{F}_{n_i} \in \mathbb{C}^{t_i \times n_i}, i = 1, \ldots, s$, denotes the one-dimensional DFT. Let Z_i denote $(\mathcal{F}_{n_s} \otimes \mathcal{F}_{n_{s-1}} \otimes \cdots \mathcal{F}_{n_1}) \operatorname{Diag}(w_i)$ and Z denote $(Z_1^T \ Z_2^T \ \cdots \ Z_l^T)^T$. We have $\mathbb{A}(\mathbf{x}) = \operatorname{diag}(Zxx^*Z^*)$, which implies that $\mathcal{A}(X) = \operatorname{diag}(ZXZ^*)$.

When the entries in the noise ϵ are drawn from the normal distribution with mean 0 and variance τ, the cost functions of (3.4) and (3.2) are essentially identical, i.e., for (3.2), $H_1(X) = \|b - \operatorname{diag}(ZXZ^*)\|_2^2 + \kappa \operatorname{tr}(X)$, and for (3.4), $H_2(X) = \frac{1}{\tau^2}\|b - \operatorname{diag}(ZXZ^*)\|_2^2 + \kappa \operatorname{tr}(X)$. Without loss of generality, only the cost function $H(X) = \|b - \operatorname{diag}(ZXZ^*))\|_2^2/\|b\|_2^2 + \kappa \operatorname{tr}(X)$ is considered. It can be shown that the Euclidean gradient of H is $\operatorname{grad} H(X) = \frac{2}{\|b\|_2^2} Z^* \operatorname{Diag}(\operatorname{diag}(ZXZ^*) - b)Z + \kappa I_{n\times n}$. The gradients of functions F_p and f_p can be constructed by using Lemmas 4.2 and 5.3.

Stationary points, including local minimizers, of F_p with rank p can be discarded if found and the algorithm restarted appropriately possibly with an increased p. If an X with numerical rank $1 < r < p$ is encountered when iterating using F_p and X is not a stationary point, then the rank reduction strategy increases efficiency by removing the unnecessary directions from X and continuing the iteration on F_r.

Even though stationary points with rank 1 that are not local minimizers of F_1 may exist, their presence tends to simply slow the algorithm rather than stop the iteration at the saddle point; see the experiments in section 6.5. As expected, therefore, running with $p > 1$ avoids this issue completely. There is no theorem guaranteeing that the iterates generated by optimizing F_p with adapting p but remaining greater than 1 always converge to an approximation of the rank-one minimizer of H in PhaseLift, but such convergence occurred in all of the experiments below. The use of a carefully chosen initial iterate as discussed below is partly responsible, though the convergence is also observed with random initial iterate.

6.2. Initial iterate. The initial iterate $Y_p^{(0)}$ is computed by Algorithm 3, which generalizes [CLS16, Algorithm 1] such that the value $p > 1$ is allowed. It is shown in [CLS16] that the initialization can be satisfactory with high probability if the number of measure m is large enough, i.e., $m \geq cn\log(n)$ for some sufficiently large constant c. Note that the initial iterate is chosen such that its singular values are identical. This choice of initial point minimizes the influence of magnitudes of singular

Algorithm 3. Initialization.

Require: $Y \in \mathbb{C}^{n\times p}$ drawn from the standard normal distribution;

Ensure: Initial iterate $Y_p^{(0)}$;

1: $Y \leftarrow \operatorname{qr}(Y)$, where $\operatorname{qr}(M)$ is a Q-factor of the QR-decomposition of M;
2: **for** i = 1, . . . , N **do**
3: $\quad Y \leftarrow \operatorname{qr}(Z^* \operatorname{Diag}(b) ZY)$;
4: **end for**
5: $\lambda = \sqrt{\frac{n\sum_{i=1}^m b_i}{\sum_{i=1}^l \operatorname{vec}(\mathbf{w}_i)^* \operatorname{vec}(\mathbf{w}_i)}}$;
6: $Y_p^{(0)} \leftarrow \lambda Y$;

values of the initial point. In other words, if a bias of magnitudes of singular values is shown during iteration, one knows that the bias is generated by the algorithm and the surface of the cost function not the initial iterate.

6.3. Data, parameters, and notation. A complex number $a + b\sqrt{-1}$ is said to be drawn from a distribution in this paper if both a and b are drawn from the distribution independently. The entries of the true solution x_* and Gaussian masks $w_i, i = 1, \dots l$, are drawn from the standard normal distribution. The entries of x_* are further normalized by $\|x_*\|_2$. For the noiseless problem, the measurement b is set to be $\mathrm{diag}(Zx_*x_*^*Z^*)$, and for the Gaussian noise problem, the measurement b is set to be $\mathrm{diag}(Zx_*x_*^*Z^*) + \varepsilon$, where the entries of $\varepsilon \in \mathbb{R}^m$ are drawn from the normal distribution with mean 0 and variance τ that is specified later for each experiment. The rank of the first iterate is denoted by p_0.

The limited-memory version of the Riemannian BFGS method (LRBFGS) is chosen to be the representative Riemannian method in step 2 of Algorithm 2. If the norm of the gradient over the norm of the initial gradient is smaller than 10^{-6}, then the step size of LRBFGS is fixed to be 1. The minimum number of iterations at each rank is 10. The parameter κ is chosen to be $1/\sqrt{n}$ if $p > 1$ and 0 if $p = 0$ for Algorithm 2.

The codes are written in C++ using the library ROPTLIB [HAGH16] through its MATLAB interface. All experiments are performed in MATLAB R2016b on a 64-bit Windows system with a 3.4-GHz CPU (Intel Core i7-6700). The DFT is performed using the library FFTW [FJ05] with one thread. The code is available at http://www.math.fsu.edu/~whuang2/papers/SPLRROMCSC.htm.

6.4. Implementation of a limited-memory BFGS method on the fixed-rank manifold $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$. We exploit the version of LRBFGS developed in [HGA15, Algorithm 2] and modify the version to use an alternate update defined in [HAG16a] which allows the line search using the Wolfe conditions to be replaced by the Armijo line search. The LRBFGS method is stated in Algorithm 4. It is shown in [HAG16b] that a tangent vector can be represented by a vector with size the dimension of the manifold. The vector transport by parallelization using this representation is essentially identity, which is the cheapest one can expect. The necessary subalgorithms for efficiently computing the representation of vector transports are given in Algorithms 5 to 9.

The complexities of the algorithms are measured by the number of fast Fourier transforms (FFT) (or inverse FFT) and the number of matrix multiplications (MM) between an m-by-p matrix and a p-by-p matrix. The complexities are given on the right-hand side of the algorithms except the operations with lower-order complexities. Note that $p \ll n$ holds in practice. The operations with complexity of $O(p^3)$ are not reported either. Since MM depends on p, we define complexity unit NN, which is independent of p, such that MM = NNp^2. If the step size is accepted at the first try in the line search algorithm, then the complexity of one iteration in Algorithm 4 is $2lp$ FFT + $7p^2$ NN without consideration of the cost in the stopping criterion. If $p > 1$, then the computations of singular values for checking the stopping criterion take additional $2p^2$ NN.

6.5. Initial point size and rank reducing threshold. In this section, noiseless problems are used. The initiate iterate of LRBFGS is obtained by Algorithm 3 with $N = 10$. The stopping criterion of Algorithm 2 requires the norm of gradient to be less than 10^{-10}.

Algorithm 4. LRBFGS for problems on $\mathbb{C}_*^{n\times p}/\mathcal{O}_p$.

Require: Initial iterate $x_0 \in \mathcal{M}$; an integer $\chi > 0$; line search constant $\delta \in (0,1)$.

1: $k = 0$, $\gamma_0 = 1$, $\ell = 0$; compute $F_p(x_k)$ and $\operatorname{grad} F_p(x_k)$; ▷ # $2lp$ FFT;
2: Compute the intrinsic representation gf_k^d of $\operatorname{grad} F(x_k)(x_k^* x_k)^{-1}$ by Algorithm 5; ▷ # $\frac{9}{2}p^2$ NN;[2]
3: Obtain $\eta_k \in \mathbb{C}^d$, intrinsic representation of a vector $\eta^w \in \mathrm{T}_{x_k}\mathcal{M}$, by the following algorithm, step 4 to step 14:
4: $q \leftarrow \mathrm{gf}_k^d$;
5: **for** $i = k-1, k-2, \ldots, k-\ell$ **do**
6: $\quad \xi_i \leftarrow \rho_i \operatorname{Re}(q^* s_i)$;
7: $\quad q \leftarrow q - \xi_i y_i$;
8: **end for**
9: $r \leftarrow \gamma_k q$;
10: **for** $i = k-\ell, k-\ell+1, \ldots, k-1$ **do**
11: $\quad \omega \leftarrow \rho_i \operatorname{Re}(r^* y_i)$;
12: $\quad r \leftarrow r + s_i(\xi_i - \omega)$;
13: **end for**
14: set $\eta_k = -r$;
15: Compute $\eta_k^w = \mathrm{D2E}_{x_k}(\eta_k)$ by Algorithm 6; ▷ # $\frac{5}{2}p^2$ NN
16: find the largest $\alpha_k \in \{1, \varrho, \varrho^2, \ldots\}$ satisfying

$$F(x_k + \alpha_k \eta_k^w) \le F(x_k) + \delta \alpha_k \eta_k^T \mathrm{gf}_k^d,$$

17: Compute $F(x_{k+1})$; ▷ # lp FFT
18: Set $x_{k+1} = x_k + \alpha_k \eta_k^w$;
19: Apply Algorithm 7 with $Z = x_{k+1}$ and obtain unit vectors $V = \{v_1, v_2, \ldots, v_p\}$ and complex numbers $S = \{s_1, s_2, \ldots, s_p\}$ and $W = \{w_1, w_2, \ldots, w_p\}$, and the lower triangle matrix L. ▷ # p^2 NN
20: Compute $\operatorname{grad} F(x_{k+1})$; ▷ # lp FFT
21: Compute the intrinsic representation gf_{k+1}^d of $\operatorname{grad} F(x_{k+1}) L^{-1} L^{-*}$; ▷ # $\frac{7}{2}p^2$ NN
22: Define $s_k = \alpha_k \eta_k$ and $y = \mathrm{gf}_{k+1}^d - \mathrm{gf}_k^d$;
23: Compute $a = \operatorname{Re}(y_k^* s_k)$ and $b = \|s_k\|_2^2$;
24: **if** $\frac{a}{b} \ge 10^{-4} \|\mathrm{gf}_k^d\|_2$ **then**
25: $\quad$ Compute $c = \|y_k^{(k+1)}\|_2^2$ and define $\rho_k = 1/a$ and $\gamma_{k+1} = a/c$;
26: $\quad$ Add s_k, y_k and ρ_k into storage and if $\ell \ge \chi$, then discard vector pair $\{s_{k-\ell}, y_{k-\ell}\}$ and scalar $\rho_{k-\ell}$ from storage, else $\ell \leftarrow \ell + 1$;
27: **else**
28: $\quad$ Set $\gamma_{k+1} \leftarrow \gamma_k$, $\{\rho_k, \ldots, \rho_{k-\ell+1}\} \leftarrow \{\rho_{k-1}, \ldots, \rho_{k-\ell}\}$, $\{s_k, \ldots, s_{k-\ell+1}\} \leftarrow \{s_{k-1}, \ldots, s_{k-\ell}\}$ and $\{y_k, \ldots, y_{k-\ell+1}\} \leftarrow \{y_{k-1}, \ldots, y_{k-\ell}\}$;
29: **end if**
30: $k = k+1$, goto Step 3;

Table 1 presents the experimental results of Algorithm 2 with $n_1 = n_2 = 128$ several values of l, p_0, and δ. When $l = 6$, the average computational time and the standard derivation of $p_0 = 1$ are much larger relatively than the starting ranks

[2]Note that $Alg5(Y, U) = Alg5(Y, P_Y^h(U))$, where $Y \in \mathbb{C}_*^{n\times p}$ and $Alg5 : (\mathbb{C}_*^{n\times p}, \mathbb{C}^{n\times p}) \to \mathbb{C}^{np-p(p+1)/2}$ is the function defined by Algorithm 5

Algorithm 5. Compute the intrinsic representation of $U \in \mathcal{H}_Y$.

Require: $Y \in \mathbb{C}_*^{n\times p}$, $U \in \mathcal{H}_X$, a function $\alpha_Y : \mathbb{C}^{n\times p} \to \mathbb{C}^{n\times p} : A \mapsto \begin{bmatrix} YL^{-*} & Y_\perp \end{bmatrix}^T A$ (see Algorithm 8), where $Y^*Y = LL^*$ is the Cholesky decomposition.

1: $\begin{bmatrix} \tilde{\Omega} \\ \tilde{K} \end{bmatrix} = \alpha_Y(U)$, where $\tilde{\Omega} \in \mathbb{C}^{p\times p}$ and $\tilde{K} \in \mathbb{C}^{(n-p)\times p}$; ▷ # $2p^2$ NN

2: Set $\Omega = (\tilde{\Omega}L + L^*\tilde{\Omega}^*)/2$, $K = \tilde{K}L$ and $k = 1$; ▷ # $\frac{1}{2}p^2$ NN

3: **for** $j = 2, \ldots, p$, $i = 1, \ldots j-1$ **do**

4: $v_X(k) = \Omega_{ij}$, where Ω_{ij} is the ith row jth column entry of Ω;

5: $k \leftarrow k + 1$;

6: **end for**

7: **for** $i = 1, \ldots, (n-p)$, $j = 1, \ldots, p$ **do**

8: $v_X(k) = K_{ij}$ and $k \leftarrow k + 1$;

9: **end for**

10: return vector $v_X \in \mathbb{C}^{np-p(p+1)/2}$;

Algorithm 6. Compute a vector in $\mathcal{H}_Y$ from its intrinsic representation.

Require: $Y \in \mathbb{C}_*^{n\times p}$, $v_Y \in \mathbb{C}^{np-p(p+1)/2}$, a function $\beta_Y : \mathbb{C}^{n\times p} \to \mathbb{C}^{n\times p} : A \mapsto \begin{bmatrix} YL^{-*} & Y_\perp \end{bmatrix} A$ (see Algorithm 9), where $Y^*Y = LL^*$ is the Cholesky decomposition.

1: $k = 1$;

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

3: $\Omega_{ij} = v_Y(k)$ and $\Omega_{ji} = -v_Y(k)$;

4: $k \leftarrow k + 1$;

5: **end for**

6: **for** $i = 1, \ldots, (n-p)$, $j = 1, \ldots, p$ **do**

7: $K_{ij} = v_Y(k)$ and $k \leftarrow k + 1$;

8: **end for**

9: Set $\tilde{\Omega} = \Omega L^{-1}$ and $\tilde{K} = KL^{-1}$; ▷ # $\frac{1}{2}p^2$ NN;

10: return $\beta_Y \begin{bmatrix} \tilde{\Omega} \\ \tilde{K} \end{bmatrix}$; ▷ # $2p^2$ NN;

Algorithm 7. Compute unit vectors in Householder matrices $(v_1, v_2, \ldots, v_p)$, complex numbers $(s_1, s_2, \ldots, s_p)$, $(w_1, w_2, \ldots, w_p)$, and a lower triangle matrix L satisfying $LL^* = Z^*Z$.

Require: $Z = \begin{bmatrix} z_1 & z_2 & \cdots & z_p \end{bmatrix} \in \mathbb{C}^{n\times p}$;

1: **for** $i = 1, \ldots, p$ **do** ▷ # p^2 NN

2: Let a denote $-e^{\arg \tilde{z}_{i1}\sqrt{-1}}\|\tilde{z}_i\|_2$ and define $v_i = (\tilde{z}_i - ae_1)/\|\tilde{z}_i - ae_1\|_2$, $s_i = -e^{\arg \tilde{z}_{i1}\sqrt{-1}}$, and $w_i = \tilde{z}_i^* v_i / v_i^* \tilde{z}_i$, where $\tilde{z}_i$ is the vector formed by last $n-i+1$ entries of z_i, $\tilde{z}_{i1}$ is the first entry of $\tilde{z}_i$ and e_1 denotes the first canonical basis of $\mathbb{R}^{n-i+1}$;

3: $Z = \begin{bmatrix} z_1 & z_2 & \cdots & z_p \end{bmatrix} \leftarrow Q_i Z$, where $Q_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & I_{n-i+1} - (1+w_i)v_i v_i^* \end{bmatrix}$;

4: **end for**

5: $L \leftarrow Z(1:p, 1:p)^*$; ▷ Z is an upper triangle matrix.

6: return $(v_1, v_2, \ldots, v_p)$, $(s_1, s_2, \ldots, s_p)$, $(w_1, w_2, \ldots, w_p)$, and L;

Algorithm 8. Compute $\alpha_X(A)$.

Require: $A \in \mathbb{C}^{n\times p}$, and $V_X = (v_1, v_2, \ldots, v_p)$, $S_X = (s_1, s_2, \ldots, s_p)$, and $W_X = (w_1, w_2, \ldots, w_p)$ generated by Algorithm 7 with input X;

1: **for** $i = 1, \ldots, p$ **do** ▷ # $2p^2$ NN

2: $A \leftarrow Q_i A$, where $Q_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & I_{n-i+1} - (1+w_i)v_i v_i^* \end{bmatrix}$;

3: **end for**

4: return $\mathrm{diag}(s_1^*, s_2^*, \ldots, s_p^*, I_{n-p})A$;

Algorithm 9. Compute $\beta_X(A)$.

Require: $A \in \mathbb{C}^{n\times p}$, and $V_X = (v_1, v_2, \ldots, v_p)$, $S_X = (s_1, s_2, \ldots, s_p)$, and $W_X = (w_1, w_2, \ldots, w_p)$ generated by Algorithm 7 with input X;

1: $A \leftarrow \mathrm{diag}(s_1, s_2, \ldots, s_p, I_{n-p})A$

2: **for** $i = p, (p-1) \ldots, 1$ **do** ▷ # $2p^2$ NN

3: $A \leftarrow Q_i A$, where $Q_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & I_{n-i+1} - (1+w_i)v_i v_i^* \end{bmatrix}$.

4: **end for**

5: Return A;

TABLE 1

The mean and the standard derivation of computational time of 100 *runs of Algorithm* 2 *using LRBFGS with variant* l, p_0, *and* δ *and output format is* (*mean*)/(*the standard derivation*). *Since* δ *does not take effect for* $p_0 = 1$, *the row corresponding to* $p_0 = 1$ *has only one result. The subscript* k *indicates a scale of* 10^k.

		δ:0.95	δ:0.9	δ:0.85	δ:0.8	δ:0.75
l:6	p_0:1			$3.13/3.54$		
	p_0:2	$1.60/5.54_{-1}$	$1.49/3.56_{-1}$	$1.39/2.72_{-1}$	$1.37/2.31_{-1}$	$1.38/2.14_{-1}$
	p_0:4	$1.79/3.96_{-1}$	$1.73/2.22_{-1}$	$1.72/1.87_{-1}$	$1.73/1.99_{-1}$	$1.75/1.79_{-1}$
l:20	p_0:1			$9.35_{-1}/8.48_{-2}$		
	p_0:2	$1.30/7.20_{-2}$	$1.30/8.45_{-2}$	$1.31/6.99_{-2}$	$1.29/8.21_{-2}$	$1.29/7.26_{-2}$
	p_0:4	$1.99/1.65_{-1}$	$2.00/1.71_{-1}$	$1.99/1.55_{-1}$	$1.99/1.46_{-1}$	$2.02/1.66_{-1}$

$p_0 = 2, 4$. Note that Algorithm 3 with a small value of l, which is the case in these tests, often does not give a satisfactory initial iterate. Therefore, if the initial point is close to the global rank-one minimizer, then Algorithm 2 with $p_0 = 1$ is fast; otherwise Algorithm 2 with $p_0 = 1$ is usually very slow. This explains the big standard derivation of computational time for $p = 1$. Using $p_0 > 1$ improves the efficiency of the algorithm. It allows the algorithm to search on a larger dimensional space and find a more reasonable initial point for Algorithm 2 when p finally reduces to 1. However, since optimizing over a higher dimensional space requires more work on each iteration, Algorithm 2 with $p_0 = 4$ is not as fast as $p_0 = 2$. On the other hand, when $l = 20$, Algorithm 3 is able to provide a satisfactory initial condition. Therefore, it is not necessary to search over a larger dimensional space and Algorithm 2 with $p_0 = 1$ is efficient and reliable.

Table 2 reports empirical probabilities of success with the starting rank $p_0 = 1, 2$ and $n_1 = n_2 = 32$. The threshold δ is set to 0.9 for $p_0 = 2$. Multiple values of l are used. Using starting rank $p_0 = 2$ increases the probability of success when the number of measurements are not sufficient.

TABLE 2
Empirical probability of success based on 100 *random trials for different signal/measurement.*

l	2	3	4	5	6	7	8
$p_0 : 1$	0	0.01	0.69	0.97	0.98	1.00	1.00
$p_0 : 2$	0	0.48	0.96	0.99	1.00	1.00	1.00

TABLE 3
The comparison results of an average of 10 *random runs between the Wirtinger flow algorithm and Algorithm* 2. *WF and R denote the Wirtinger flow algorithm and the LRBFGS method. err denotes the relative error* $\min_{|a|=1} \|ax - x_*\|/\|x_*\|$

		$l = 6, \epsilon = 10^{-5}$, $p_0 = 2$				$l = 20, \epsilon = 10^{-10}$, $p_0 = 1$			
		$n_1 = n_2 = 16$		$n_1 = n_2 = 32$		$n_1 = n_2 = 128$		$n_1 = n_2 = 256$	
τ		WF	R	WF	R	WF	R	WF	R
0	t	9.12_{-2}	2.81_{-2}	2.28_{-1}	4.91_{-2}	3.84	9.23_{-1}	2.25_{1}	6.11
	FFT	4864	1129	7037	1305	12496	2560	11092	2662
	NN	0	739	0	833	0	372	0	385
	err	2.67_{-4}	1.68_{-4}	6.30_{-4}	5.81_{-4}	5.42_{-9}	9.27_{-9}	1.11_{-8}	3.21_{-8}
10^{-6}	t	8.13_{-2}	2.22_{-2}	1.97_{-1}	4.96_{-2}	3.88	9.33_{-1}	2.23_{1}	6.13
	FFT	4864	1129	7037	1327	12492	2554	11092	2688
	NN	0	735	0	839	0	372	0	388
	err	2.68_{-4}	1.65_{-4}	6.36_{-4}	5.15_{-4}	3.51_{-4}	3.51_{-4}	10.00_{-4}	10.00_{-4}
10^{-4}	t	7.97_{-2}	2.18_{-2}	2.00_{-1}	4.91_{-2}	3.79	8.84_{-1}	2.19_{1}	6.14
	FFT	4868	1127	7033	1312	12492	2526	11104	2746
	NN	0	732	0	827	0	370	0	393
	err	3.40_{-3}	3.39_{-3}	9.87_{-3}	9.90_{-3}	3.51_{-2}	3.51_{-2}	10.00_{-2}	10.00_{-2}

We conclude from these experiments that the rank reducing algorithm, Algorithm 2, with $p_0 > 1$ is useful in the sense of both efficiency and effectiveness when a satisfactory initial iterate is unknown.

6.6. Comparisons with a standard low-rank method. A state-of-the-art algorithm, the Wirtinger flow algorithm, is proposed in [CLS16]. This algorithm works by defining

$$Y^{(k+1)} = Y^{(k)} - \frac{\mu_{k+1}}{\|Y^{(0)}\|^2} \nabla F(Y^{(k)}),$$

where $Y^{(0)}$ is the first iterate given by Algorithm 3 and $\{\mu_k\}$ is a sequence of predefined step sizes. We use the same heuristic formula as in [CLS16]: $\mu_k = \min(1 - e^{-k/k_0}, \mu_{\max})\mu_0$, where k_0, $\mu_{\max}$, and μ_0 are given constants. The coefficient in the formula of μ_k needs to be carefully chosen, and we use the best values by tuning them during our tests. The stopping criteria of the Wirtinger flow algorithm and Algorithm 2 require the norm of the gradient to be less than ϵ, which is specified in Table 3. The initial iterate of the Wirtinger flow algorithm is given by Algorithm 3 with $N = 50$, which is the same as the setting in [CLS16].

Table 3 shows that Algorithm 2 is significantly faster than the Wirtinger flow algorithm for both noiseless and noise problems in the sense of both computational time and machine-independent criteria, the number of FFT and NN. Note that the complexity of 1 FFT is larger than 1 NN.

Candes et al. [CESV13, CSV13] use a MATLAB library, TFOCS [BCG11], that contains a variety of accelerated first-order methods given in [Nes04] and, in particular, the method based on FISTA [BT09] is used to optimize the cost functions (3.2) in PhaseLift. Since the domain $\mathcal{D}_n$ has dimension $\frac{1}{2}n(n+1)$, which is usually too large to be solved, a low-rank version of FISTA (LR-FISTA) is used instead in [CESV13].

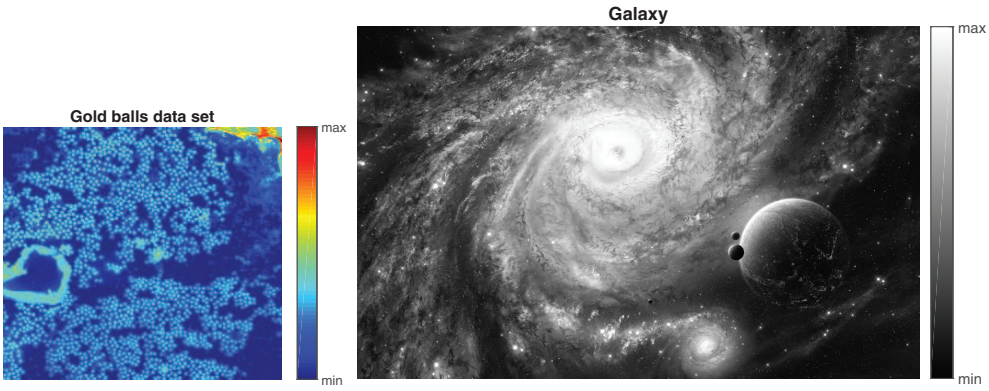

FIG. 1. *Left: A gold balls data set image of* 256 *by* 256 *pixels. The values of pixels are complex numbers. The number of iterations is* 85. *The computational time is* 9.3 *seconds, including the computations for an initial iterate. The relative error* $\min_{|a|=1} \|ax - x_*\|/\|x_*\|$ *is* 1.60×10^{-15}. *The numbers of FFT and NN are* 4020 *and* 682, *respectively. Right: A gray galaxy image of* 1800 *by* 2880 *pixels. The values of pixels are real numbers. The number of iterations is* 136. *The computational time is* 1473 *seconds including the computations for an initial iterate. The relative error is* 1.96×10^{-15}. *The numbers of FFT and NN are* 6000 *and* 1090, *respectively.*

However, LR-FISTA is not a competitive algorithm; see details in [HGZ16a] for comparisons between Algorithm 2 and LR-FISTA.

6.7. Performance of PhaseLift on natural images. Two images of different sizes, shown in Figure 1, are used to illustrate the performance of Algorithm 2 for noiseless measurements. Twenty masks $l = 20$ and $p_0 = 1$ are used. Algorithm 2 stops when the norm of the gradient over the norm of the initial gradient is smaller than 10^{-15}. Algorithm 2 is able to recover the images in minutes.

7. Conclusion. In this paper, the recently proposed PhaseLift framework for solving the phase retrieval problem motivated the consideration of cost functions H on the set of complex Hermitian positive semidefinite matrices $\mathcal{D}_n$ that include the PhaseLift cost function.

An alternate cost function F related to factorization is used to replace the cost function H, i.e., $F(Y) = H(YY^*)$. The optimality conditions of H are related to the properties of F, and the important optimality condition, Theorem 4.3, shows that if Y_p is a rank deficient minimizer of F_p, then $Y_pY_p^*$ is a stationary point of H. For general problems defined on $\mathcal{D}_n$, if r_*, the rank of the desired minimizer of cost function H, is low, the optimality condition suggests the use of the alternate cost function F with $p > r_*$. If r_* is small, then a small p can be used and optimization on F_p can be more efficient than optimization on H.

Additionally, Algorithm 2 based on optimization on a fixed-rank manifold and dynamically reducing is developed for optimizing the cost function F. For optimization on a fixed-rank manifold, recently developed state-of-the-art Riemannian optimization methods on a quotient space are used.

For the phase retrieval problem, when the number of measurements is not sufficient large, Algorithm 3 does not give a satisfactory initial condition. Algorithm 2 with higher starting rank $p_0 > 1$ improves the efficiency and reliability. Furthermore, Algorithm 2 is much faster than the Wirtinger flow algorithm in the sense of computational time and the number of machine-independent operations regardless of $p_0 = 1$ or 2.

Acknowledgment. We thank Stefano Marchesini at Lawrence Berkeley National Laboratory for providing the gold balls data set and granting permission to use it.

REFERENCES

[AIDV09] P.-A. Absil, M. Ishteva, L. De Lathauwer, and S. Van Huffel, *A geometric Newton method for Oja's vector field*, Neural Comput., 21 (2009), pp. 1415–33, doi:10.1162/neco.2008.04-08-749.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, NJ, 2008.

[Bak08] C. G. Baker, *Riemannian Manifold Trust-Region Methods with Applications to Eigenproblems*. PhD thesis, Department of Computational Science, Florida State University, 2008.

[BCG11] S. Becker, E. J. Cand, and M. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Math. Program. Comput., 3 (2011), pp. 165–218.

[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 Crystallogr. Sect. A, 63 (2007), pp. 306–314, doi:10.1107/S0108767307021903.

[Bla04] R. Blankenbecler, *Three-dimensional image reconstruction.* II. *Hamiltonian method for phase recovery*, Phys. Rev. B, 69 (2004), doi:10.1103/PhysRevB.69.064108.

[BM03] S. Burer and R. D. C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., 95 (2003), pp. 329–357, doi:10.1007/s10107-002-0352-8.

[Boo86] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, 2nd ed., Academic Press, New York, 1986.

[BS79] Y. M. Bruck and L. G. Sodin, *On the ambiguity of the image recontruction problem*, Optics Commun., 30 (1979), pp. 304–308.

[BT09] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202, doi:10.1137/080716542.

[BV04] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004, doi:10.1017/CBO9780511804441.

[CESV13] E. J. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski, *Phase retrieval via matrix completion*, SIAM J. Imaging Sci., 6 (2013), pp. 199–225, arXiv:1109.0573v2.

[CL13] E. J. Candès and X. Li, *Solving quadratic equations via phaselift when there are about as many equations as unknowns*, Found. Comput. Math., 14 (2014), pp. 1017–1026, doi:10.1007/s10208-013-9162-z.

[CLS16] E. J. Candés, X. Li, and M. Soltanolkotabi, *Phase retrieval via Wirtinger flow: Theory and algorithms*, IEEE Trans. Inform. Theory, 64 (2016), pp. 1985–2007.

[CMWL07] C. Chen, J. Miao, C. W. Wang, and T. K. Lee, *Application of optimization technique to noncrystalline x-ray diffraction microscopy: Guided hybrid input-output method*, Phys. Rev. B, 76 (2007), doi:10.1103/PhysRevB.76.064113.

[CSV13] E. J. Candès, T. Strohmer, and V. Voroninski, *PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming*, Commun. Pure Appl. Math., 66 (2013), pp. 1241–1274.

[DH14] L. Demanet and P. Hand, *Stable optimizationless recovery from phaseless linear measurements*, J. Fourier Anal. Appl., 20 (2014), pp. 199–221.

[Els03] V. Elser, *Solution of the crystallographic phase problem by iterated projections*, Acta. Crystallogr. Sect. A, 59 (2003), pp. 201–209.

[Fie78] J. R. Fienup, *Reconstruction of an object from the modulus of its Fourier transform*, Optics Lett., 3 (1978), pp. 27–29.

[Fie82] J. R. Fienup, *Phase retrieval algorithms: A comparison*, Appl. Optics, 21 (1982), pp. 2758–69.

[FJ05] M. Frigo and S. G. Johnson, *The design and implementation of FFTW*3, Proc. IEEE, 93 (2005), pp. 216–231.

[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), pp. 237–246.

[HAG15] W. Huang, P.-A. Absil, and K. A. Gallivan, *A Riemannian symmetric rank-one trust-region method*, Math. Program., 150 (2015), pp. 179–216.

[HAG16a] W. Huang, P.-A. Absil, and K. A. Gallivan, *A Riemannian BFGS Method for Nonconvex Optimization Problems*, Lect. Notes Comput. Sci. Eng., 112, Springer, New York, 2016, pp. 627–634.

[HAG16b] W. Huang, P.-A. Absil, and K. A. Gallivan, *Intrinsic representation of tangent vectors and vector transport on matrix manifolds*, Numeri. Math., 136 (2017), pp. 523–543.

[HAGH16] W. Huang, P.-A. Absil, K. A. Gallivan, and P. Hand, *ROPTLIB: An Object-Oriented C++ Library for Optimization on Riemannian Manifolds*, Technical Report FSU16-14, Florida State University, 2016.

[Har93] R. W. Harrison, *Phase problem in crystallography*, J. Opt. Soc. Amer. A, 10 (1993), pp. 1046–1055, doi:10.1364/JOSAA.10.001046.

[Hay82] M. H. Hayes, *The reconstruction of a multidimensional sequence from the phase or magnitude of its Fourier transform*, IEEE Trans. Acoustics Speech Signal process., 30 (1982), pp. 140–154.

[HGA15] W. Huang, K. A. Gallivan, and P.-A. Absil, *A Broyden class of quasi-Newton methods for Riemannian optimization*, SIAM J. Optim., 25 (2015), pp. 1660–1685.

[HGZ16a] W. Huang, K. A. Gallivan, and X. Zhang, *Solving PhaseLift by Low-Rank Riemannian Optimization Methods for Complex Semidefinite Constraints*, UCL-INMA-2015.01.v2, 2016.

[HGZ16b] W. Huang, K. A. Gallivan, and X. Zhang, *Solving phaselift by low rank Riemannian optimization methods*, Procedia Comput. Sci., 80 (2016), pp. 1125–1134.

[Hua13] W. Huang, *Optimization Algorithms on Riemannian Manifolds with Applications*, PhD thesis, Department of Mathematics, Florida State University, 2013.

[JBAS10] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre, *Low-rank optimization on the cone of positive semidefinite matrices*, SIAM J. Optim., 20 (2010), pp. 2327–2351.

[Mar07] S. Marchesini, *Invited article: a unified evaluation of iterative projection algorithms for phase retrieval*, Rev. Sci. Instruments, 78 (2007), doi:10.1063/1.2403783.

[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. Physical chemistry, 59 (2008), pp. 387–410, doi:10.1146/annurev.physchem.59.032607.093642.

[Nes04] Y. Nesterov, *Introductory Lectures on Convex Programming: A Basic Course*, Vol. I, Springer, New York, 2004.

[RW12] W. Ring and B. Wirth, *Optimization methods on Riemannian manifolds and their application to shape space*, SIAM J. Optim., 22 (2012), pp. 596–627, doi:10.1137/11082885X.

[San85] J. L. C. Sanz, *Mathematical considerations for the problem of Fourier transform phase retrieval from magnitude*, SIAM J. Appl. Math., 45 (1985), pp. 651–664.

[Sat15] H. Sato, *A Dai-Yuan-type Riemannian conjugate gradient method with the weak Wolfe conditions*, Comput. Optim. Appl., 64 (2016), pp. 101–118.

[SI15] H. Sato and T. Iwai, *A new, globally convergent Riemannian conjugate gradient method*, Optimization, 64 (2015), pp. 1011–1031.

[TTT99] K. C. Toh, M. J. Todd, and R. H. Tutuncu, *SDPT3: A MATLAB software package for semidefinite programming*, Optim. Methods Softw., 11 (1999), pp. 545–581.

[Wal63] A. Walther, *The question of phase retrieval in optics*, Optica Acta, 10 (1963), pp. 41–49, doi:10.1080/713817747.

[WDAM13] I. Waldspurger, A. D'Aspremont, and S. Mallat, *Phase recovery, MaxCut and complex semidefinite programming*, Math. Program., 149 (2015), pp. 47–81, doi:10.1007/s10107-013-0738-9.

[ZHG+15] G. Zhou, W. Huang, K. A. Gallivan, P. Van Dooren, and P.-A. Absil, *A Riemannian rank-adaptive method for low-rank optimization*, Neurocomputing, 192 (2016), pp. 72–80.