PHASE RECOVERY, MAXCUT AND COMPLEX SEMIDEFINITE PROGRAMMING

IRÈNE WALDSPURGER, ALEXANDRE D'ASPREMONT, AND STÉPHANE MALLAT

ABSTRACT. Phase retrieval seeks to recover a signal $x \in \mathbb{C}^p$ from the amplitude $|Ax|$ of linear measurements $Ax \in \mathbb{C}^n$. We cast the phase retrieval problem as a non-convex quadratic program over a complex phase vector and formulate a tractable relaxation (called PhaseCut) similar to the classical *MaxCut* semidefinite program. We solve this problem using a provably convergent block coordinate descent algorithm whose structure is similar to that of the original greedy algorithm in Gerchberg and Saxton [1972], where each iteration is a matrix vector product. Numerical results show the performance of this approach over three different phase retrieval problems, in comparison with greedy phase retrieval algorithms and matrix completion formulations.

1. INTRODUCTION

The phase recovery problem, i.e. the problem of reconstructing a complex phase vector given only the magnitude of linear measurements, appears in a wide range of engineering and physical applications. It is needed for example in X-ray and crystallography imaging [Harrison, 1993], diffraction imaging [Bunk et al., 2007] or microscopy [Miao et al., 2008]. In these applications, the detectors cannot measure the phase of the incoming wave and only record its amplitude. Recovering the complex phase of wavelet transforms from their amplitude also has applications in audio signal processing [Griffin and Lim, 1984].

In all these problems, complex measurements of a signal $x \in \mathbb{C}^p$ are obtained from a linear injective operator A, but we can only measure the magnitude vector $|Ax|$. Depending on the properties of A, the phase of Ax may or may not be uniquely characterized by the magnitude vector $|Ax|$, up to an additive constant, and it may or may not be stable. For example, if A is a one-dimensional Fourier transform, then the recovery is not unique but it becomes unique almost everywhere for an oversampled two-dimensional Fourier transform, although it is not stable. Uniqueness is also obtained with an oversampled wavelet transform operator A, and the recovery of x from $|Ax|$ is then continuous [Waldspurger and Mallat, 2012]. If A consists in random Gaussian measurements then uniqueness can be proved, together with stability results [Candes et al., 2011b; Candes and Li, 2012].

Recovering the phase of Ax from $|Ax|$ is a nonconvex optimization problem. Until recently, this problem was solved using various greedy algorithms [Gerchberg and Saxton, 1972; Fienup, 1982; Griffin and Lim, 1984], which alternate projections on the range of A and on the nonconvex set of vectors y such that $|y| = |Ax|$. However, these algorithms often stall in local minima. A convex relaxation called PhaseLift was introduced in [Chai et al., 2011] and [Candes et al., 2011b] by observing that $|Ax|^2$ is a linear function of $X = xx^*$ which is a rank one Hermitian matrix. The recovery of x is thus expressed as a rank minimization problem over positive semidefinite Hermitian matrices X satisfying some linear conditions. This last problem is approximated by a semidefinite program. It has been shown that this program recovers x with high probability when A has gaussian independant entries [Candes et al., 2011b,a]. Numerically, the same result seems to hold for several classes of linear operators A.

Our main contribution here is to formulate phase recovery as a quadratic optimization problem over the unit complex torus. We then write a convex relaxation to phase recovery very similar to the *MaxCut* semidefinite program (we call this relaxation PhaseCut in what follows). While the resulting SDP is typically

Date: July 24, 2013.

2010 *Mathematics Subject Classification.* 94A12, 90C22, 90C27.

Key words and phrases. Phase Recovery, MaxCut, Semidefinite Programming, Convex Relaxation, Scattering.

larger than the PhaseLift relaxation, its simple structure (the constraint matrices are singletons) allows us to solve it very efficiently. In particular, this allows us to use a provably convergent block coordinate descent algorithm whose iteration complexity is similar to that of the original greedy algorithm in Gerchberg and Saxton [1972] (each iteration is a matrix vector product, which can be computed efficiently). We also show that tightness of PhaseLift implies tightness of a modified version of PhaseCut. Furthermore, under the condition that A is injective and b has no zero coordinate, we derive an equivalence result between PhaseLift and a modified version of PhaseCut, in the noiseless setting. This result implies that both algorithms are simultaneously tight (an earlier version of this paper showed PhaseLift tightness implies PhaseCut tightness and the reverse direction was then proved in [Voroninski, 2012] under mild additional assumptions). In a noisy setting, one can show that PhaseCut is at least as stable as a variant of PhaseLift, while PhaseCut empirically appears to be more stable in some cases, e.g. when b is sparse.

Seeing the *MaxCut* relaxation emerge in a phase recovery problem is not entirely surprising: it appears, for example, in an angular synchronization problem where one seeks to reconstruct a sequence of angles θ_i (up to a global phase), given information on pairwise differences $\theta_i - \theta_j$ mod. 2π, for $(i,j) \in S$ [see Singer, 2011], the key difference between this last problem and the phase recovery problem in (1) is that the sign information is lost in the input to (1). Complex *MaxCut*-like relaxations of decoding problems also appear in maximum-likelihood channel detection [Luo et al., 2003; Kisialiou and Luo, 2010; So, 2010]. From a combinatorial optimization perspective, showing the equivalence between phase recovery and *MaxCut* allows us to expose a new class of nontrivial problem instances where the semidefinite relaxation for a *MaxCut*-like problem is tight, together with explicit conditions for tightness directly imported from the matrix completion formulation of these problems (these conditions are of course also hard to check, but hold with high probability for some classes of random experiments).

The paper is organized as follows. Section 2 explains how to factorize away the magnitude information to form a nonconvex quadratic program on the phase vector $u \in \mathbb{C}^n$ satisfying $|u_i| = 1$ for $i = 1, \ldots, n$, and a greedy algorithm is derived in Section 2.3. We then derive a tractable relaxation of the phase recovery problem, written as a semidefinite program similar to the classical *MaxCut* relaxation in [Goemans and Williamson, 1995], and detail several algorithms for solving this problem in Section 3. Section 4 proves that a variant of PhaseCut and PhaseLift are equivalent in the noiseless case and thus simultaneously tight. We also prove that PhaseCut is as stable as a weak version of PhaseLift and discuss the relative complexity of both algorithms. Finally, Section 5 performs a numerical comparison between the greedy, PhaseLift and PhaseCut phase recovery algorithms for three phase recovery problems, in the noisy and noiseless case. In the noisy case, these results suggest that if b is sparse, then PhaseCut may be more stable than PhaseLift.

Notations. We write $\mathbf{S}_p$ (resp. $\mathbf{H}_p$) the cone of symmetric (resp. Hermitian) matrices of dimension p ; $\mathbf{S}_p^+$ (resp. $\mathbf{H}_p^+$) denotes the set of positive symmetric (resp. Hermitian) matrices. We write $\|\cdot\|_p$ the Schatten p-norm of a matrix, that is the p-norm of the vector of its eigenvalues (in particular, $\|\cdot\|_\infty$ is the spectral norm). We write $A^\dagger$ the (Moore-Penrose) pseudoinverse of a matrix A and $\|A\|_{\ell_1}$ the sum of the modulus of the coefficients of A. For $x \in \mathbb{R}^p$, we write $\mathbf{diag}(x)$ the matrix with diagonal x. When $X \in \mathbf{H}_p$ however, $\mathbf{diag}(X)$ is the vector containing the diagonal elements of X. For $X \in \mathbf{H}_p$, X^* is the Hermitian transpose of X, with $X^* = (\bar{X})^T$. Finally, we write b^2 the vector with components b_i^2, $i = 1, \ldots, n$.

2. Phase recovery

The phase recovery problem seeks to retrieve a signal $x \in \mathbb{C}^p$ from the amplitude $b = |Ax|$ of n linear measurements, solving

$$\begin{array}{ll} \text{find} & x \\ \text{such that} & |Ax| = b, \end{array} \tag{1}$$

in the variable $x \in \mathbb{C}^p$, where $A \in \mathbb{C}^{n \times p}$ and $b \in \mathbb{R}^n$.

2.1. **Greedy Optimization in the Signal.** Approximate solutions x of the recovery problem in (1) are usually computed from $b = |Ax|$ using algorithms inspired from the alternating projection method in [Gerchberg and Saxton, 1972]. These algorithms compute iterates y^k in the set $\mathbf{F}$ of vectors $y \in \mathbb{C}^n$ such that $|y| = b = |Ax|$, which are getting progressively closer to the image of A. The Gerchberg-Saxton algorithm projects the current iterate y^k on the image of A using the orthogonal projector $AA^\dagger$ and adjusts to b_i the amplitude of each coordinate. We describe this method explicitly below.

Algorithm 1 Gerchberg-Saxton.

Input: An initial $y^1 \in \mathbf{F}$, i.e. such that $|y^1| = b$.

1: **for** $k = 1, \ldots, N-1$ **do**

2: Set

$$y_i^{k+1} = b_i \frac{(AA^\dagger y^k)_i}{|(AA^\dagger y^k)_i|}, \quad i = 1, \ldots, n. \qquad \text{(Gerchberg-Saxton)}$$

3: **end for**

Output: $y_N \in \mathbf{F}$.

Because $\mathbf{F}$ is not convex however, this alternating projection method usually converges to a stationary point y^∞ which does not belong to the intersection of $\mathbf{F}$ with the image of A, and hence $|AA^\dagger y^\infty| \neq b$. Several modifications proposed in [Fienup, 1982] improve the convergence rate but do not eliminate the existence of multiple stationary points. To guarantee convergence to a unique solution, which hopefully belongs to the intersection of $\mathbf{F}$ and the image of A, this non-convex optimization problem has recently been relaxed as a semidefinite program [Chai et al., 2011; Candes et al., 2011b], where phase recovery is formulated as a matrix completion problem (described in Section 4). Although the computational complexity of this relaxation is much higher than that of the Gerchberg-Saxton algorithm, it is able to recover x from $|Ax|$ (up to a multiplicative constant) in a number of cases [Chai et al., 2011; Candes et al., 2011b].

2.2. **Splitting Phase and Amplitude Variables.** As opposed to these strategies, we solve the phase recovery problem by explicitly separating the amplitude and phase variables, and by only optimizing the values of the phase variables. In the noiseless case, we can write $Ax = \mathbf{diag}(b)u$ where $u \in \mathbb{C}^n$ is a phase vector, satisfying $|u_i| = 1$ for $i = 1, \ldots, n$. Given $b = |Ax|$, the phase recovery problem can thus be written as

$$\min_{\substack{u \in \mathbb{C}^n, |u_i|=1, \\ x \in \mathbb{C}^p}} \|Ax - \mathbf{diag}(b)u\|_2^2,$$

where we optimize over both variables $u \in \mathbb{C}^n$ and $x \in \mathbb{C}^p$. In this format, the inner minimization problem in x is a standard least squares and can be solved explicitly by setting

$$x = A^\dagger \, \mathbf{diag}(b)u,$$

which means that problem (1) is equivalent to the reduced problem

$$\min_{\substack{|u_i|=1 \\ u \in \mathbb{C}^n}} \|AA^\dagger \, \mathbf{diag}(b)u - \mathbf{diag}(b)u\|_2^2.$$

The objective of this last problem can be rewritten as follows

$$\begin{aligned} \|AA^\dagger \, \mathbf{diag}(b)u - \mathbf{diag}(b)u\|_2^2 &= \|(AA^\dagger - \mathbf{I}) \, \mathbf{diag}(b)u\|_2^2 \\ &= u^* \, \mathbf{diag}(b^T) \tilde{M} \, \mathbf{diag}(b)u. \end{aligned}$$

where $\tilde{M} = (AA^\dagger - \mathbf{I})^*(AA^\dagger - \mathbf{I}) = \mathbf{I} - AA^\dagger$. Finally, the phase recovery problem (1) becomes

$$\begin{array}{ll} \text{minimize} & u^* M u \\ \text{subject to} & |u_i| = 1, \quad i = 1, \ldots n, \end{array} \qquad (2)$$

in the variable $u \in \mathbb{C}^n$, where the Hermitian matrix

$$M = \mathbf{diag}(b)(\mathbf{I} - AA^\dagger)\,\mathbf{diag}(b)$$

is positive semidefinite. The intuition behind this last formulation is simple, $(\mathbf{I} - AA^\dagger)$ is the orthogonal projector on the orthogonal complement of the image of A (the kernel of A^*), so this last problem simply minimizes in the phase vector u the norm of the component of $\mathbf{diag}(b)u$ which is not in the image of A.

2.3. **Greedy Optimization in Phase.** Having transformed the phase recovery problem (1) in the quadratic minimization problem (2), suppose that we are given an initial vector $u \in \mathbb{C}^n$, and focus on optimizing over a single component u_i for $i = 1, \ldots, n$. The problem is equivalent to solving

$$\begin{array}{ll} \text{minimize} & \bar{u}_i M_{ii} u_i + 2\,\mathrm{Re}\left(\sum_{j\neq i} \bar{u}_j M_{ji} u_i\right) \\ \text{subject to} & |u_i| = 1, \quad i = 1, \ldots n, \end{array}$$

in the variable $u_i \in \mathbb{C}$ where all the other phase coefficients u_j remain constant. Because $|u_i| = 1$ this then amounts to solving

$$\min_{|u_i|=1} \mathrm{Re}\left(u_i \sum_{j\neq i} M_{ji}\bar{u}_j\right)$$

which means

$$u_i = \frac{-\sum_{j\neq i} M_{ji}\bar{u}_j}{\left|\sum_{j\neq i} M_{ji}\bar{u}_j\right|} \tag{3}$$

for each $i = 1, \ldots, n$, when u is the optimum solution to problem (2). We can use this fact to derive Algorithm 2, a greedy algorithm for optimizing the phase problem.

Algorithm 2 Greedy algorithm in phase.

Input: An initial $u \in \mathbb{C}^n$ such that $|u_i| = 1$, $i = 1, \ldots, n$. An integer $N > 1$.

1: **for** $k = 1, \ldots, N$ **do**
2: **for** $i = 1, \ldots n$ **do**
3: Set

$$u_i = \frac{-\sum_{j\neq i} M_{ji}\bar{u}_j}{\left|\sum_{j\neq i} M_{ji}\bar{u}_j\right|}$$

4: **end for**
5: **end for**

Output: $u \in \mathbb{C}^n$ such that $|u_i| = 1$, $i = 1, \ldots, n$.

This greedy algorithm converges to a stationary point u^∞, but it is generally not a global solution of problem (2), and hence $|AA^\dagger\,\mathbf{diag}(u^\infty)b| \neq b$. It has often nearly the same stationary points as the Gerchberg-Saxton algorithm. One can indeed verify that if u^∞ is a stationary point then $y^\infty = \mathbf{diag}(u^\infty)b$ is a stationary point of the Gerchberg-Saxton algorithm. Conversely if b has no zero coordinate and y^∞ is a stable stationary point of the Gerchberg-Saxton algorithm then $u_i^\infty = y_i^\infty/|y_i^\infty|$ defines a stationary point of the greedy algorithm in phase.

If Ax can be computed with a fast algorithm using $O(n \log n)$ operations, which is the case for Fourier or wavelets transform operators for example, then each Gerchberg-Saxton iteration is computed with $O(n \log n)$ operations. The greedy phase algorithm above does not take advantage of this fast algorithm and requires $O(n^2)$ operations to update all coordinates u_i for each iteration k. However, we will see in Section 3.6 that a small modification of the algorithm allows for $O(n \log n)$ iteration complexity.

2.4. **Complex *MaxCut*.** Following the classical relaxation argument in [Shor, 1987; Lovász and Schrijver, 1991; Goemans and Williamson, 1995; Nesterov, 1998], we first write $U = uu^* \in \mathbf{H}_n$. Problem (2), written

$$\begin{array}{rll} QP(M) \triangleq & \text{min.} & u^*Mu \\ & \text{subject to} & |u_i| = 1, \quad i = 1, \dots n, \end{array}$$

in the variable $u \in \mathbb{C}^n$, is equivalent to

$$\begin{array}{ll} \text{min.} & \mathbf{Tr}(UM) \\ \text{subject to} & \mathbf{diag}(U) = 1 \\ & U \succeq 0, \, \mathbf{Rank}(U) = 1, \end{array}$$

in the variable $U \in \mathbf{H}_n$. After dropping the (nonconvex) rank constraint, we obtain the following convex relaxation

$$\begin{array}{rll} SDP(M) \triangleq & \text{min.} & \mathbf{Tr}(UM) \\ & \text{subject to} & \mathbf{diag}(U) = 1, \, U \succeq 0, \end{array} \tag{PhaseCut}$$

which is a semidefinite program (SDP) in the matrix $U \in \mathbf{H}_n$ and can be solved efficiently. When the solution of problem PhaseCut has rank one, the relaxation is tight and the vector u such that $U = uu^*$ is an optimal solution of the phase recovery problem (2). If the solution has rank larger than one, a normalized leading eigenvector v of U is used as an approximate solution, and $\mathbf{diag}(U - vv^T)$ gives a measure of the uncertainty around the coefficients of v.

In practice, semidefinite programming solvers are rarely designed to directly handle problems written over Hermitian matrices and start by reformulating complex programs in $\mathbf{H}_n$ as real semidefinite programs over $\mathbf{S}_{2n}$ based on the simple facts that follow. For $Z, Y \in \mathbf{H}_n$, we define $\mathcal{T}(Z) \in \mathbf{S}_{2n}$ as in [Goemans and Williamson, 2004]

$$\mathcal{T}(Z) = \begin{pmatrix} \mathrm{Re}(Z) & -\mathrm{Im}(Z) \\ \mathrm{Im}(Z) & \mathrm{Re}(Z) \end{pmatrix} \tag{4}$$

so that $\mathbf{Tr}(\mathcal{T}(Z)\mathcal{T}(Y)) = 2\,\mathbf{Tr}(ZY)$. By construction, $Z \in \mathbf{H}_n$ iff $\mathcal{T}(Z) \in \mathbf{S}_{2n}$. One can also check that $z = x + iy$ is an eigenvector of Z with eigenvalue λ if and only if

$$\begin{pmatrix} x \\ y \end{pmatrix} \text{ and } \begin{pmatrix} -y \\ x \end{pmatrix}$$

are eigenvectors of $\mathcal{T}(Z)$, both with eigenvalue λ (depending on the normalization of z, one corresponds to $(\mathrm{Re}(z), \mathrm{Im}(z))$, the other one to $(\mathrm{Re}(i\,z), \mathrm{Im}(i\,z))$. This means in particular that $Z \succeq 0$ if and only if $\mathcal{T}(Z) \succeq 0$.

We can use these facts to formulate an equivalent semidefinite program over real symmetric matrices, written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(\mathcal{T}(M)X) \\ \text{subject to} & X_{i,i} + X_{n+i,n+i} = 2 \\ & X_{i,j} = X_{n+i,n+j}, \, X_{n+i,j} = -X_{i,n+j}, \quad i,j = 1, \dots, n, \\ & X \succeq 0, \end{array}$$

in the variable X in $\mathbf{S}_{2n}$. This last problem is equivalent to PhaseCut. In fact, because of symmetries in $\mathcal{T}(M)$, the equality constraints enforcing symmetry can be dropped, and this problem is equivalent to a *MaxCut* like problem in dimension $2n$, which reads

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(\mathcal{T}(M)X) \\ \text{subject to} & \mathbf{diag}(X) = 1, X \succeq 0, \end{array} \tag{5}$$

in the variable X in $\mathbf{S}_{2n}$. As we will see below, formulating a relaxation to the phase recovery problem as a complex *MaxCut*-like semidefinite program has direct computational benefits.

3. ALGORITHMS

In the previous section, we have approximated the phase recovery problem (2) by a convex relaxation, written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(UM) \\ \text{subject to} & \mathbf{diag}(U) = 1,\ U \succeq 0, \end{array}$$

which is a semidefinite program in the matrix $U \in \mathbf{H}_n$. The dual, written

$$\max_{w \in \mathbb{R}^n} \ n\lambda_{\min}(M + \mathbf{diag}(w)) - 1^T w, \tag{6}$$

is a minimum eigenvalue maximization problem in the variable $w \in \mathbb{R}^n$. Both primal and dual can be solved efficiently. When exact phase recovery is possible, the optimum value of the primal problem PhaseCut is zero and we must have $\lambda_{\min}(M) = 0$, which means that $w = 0$ is an optimal solution of the dual.

3.1. **Interior Point Methods.** For small scale problems, with $n \sim 10^2$, generic interior point solvers such as SDPT3 [Toh et al., 1999] solve problem (5) with a complexity typically growing as $O\left(n^{4.5}\log(1/\epsilon)\right)$ where $\epsilon > 0$ is the target precision [Ben-Tal and Nemirovski, 2001, §4.6.3]. Exploiting the fact that the $2n$ equality constraints on the diagonal in (5) are singletons, Helmberg et al. [1996] derive an interior point method for solving the *MaxCut* problem, with complexity growing as $O\left(n^{3.5}\log(1/\epsilon)\right)$ where the most expensive operation at each iteration is the inversion of a positive definite matrix, which costs $O(n^3)$ flops.

3.2. **First-Order Methods.** When n becomes large, the cost of running even one iteration of an interior point solver rapidly becomes prohibitive. However, we can exploit the fact that the dual of problem (5) can be written (after switching signs) as a maximum eigenvalue minimization problem. Smooth first-order minimization algorithms detailed in [Nesterov, 2007] then produce an ϵ-solution after

$$O\left(\frac{n^3\sqrt{\log n}}{\epsilon}\right)$$

floating point operations. Each iteration requires forming a matrix exponential, which costs $O(n^3)$ flops. This is not strictly smaller than the iteration complexity of specialized interior point algorithms, but matrix structure often allows significant speedup in this step. Finally, the simplest subgradient methods produce an ϵ-solution in

$$O\left(\frac{n^2\log n}{\epsilon^2}\right)$$

floating point operations. Each iteration requires computing a leading eigenvector which has complexity roughly $O(n^2\log n)$.

3.3. **Block Coordinate Descent.** We can also solve the semidefinite program in PhaseCut using a block coordinate descent algorithm. While no explicit complexity bounds are available for this method in our case, the algorithm is particularly simple and has a very low cost per iteration (it only requires computing a matrix vector product). We write i^c the index set $\{1, \ldots, i-1, i+1, \ldots, n\}$ and describe the method as Algorithm 3.

Block coordinate descent is widely used to solve statistical problems where the objective is separable (LASSO is a typical example) and was shown to efficiently solve semidefinite programs arising in covariance estimation [d'Aspremont et al., 2006]. These results were extended by [Wen et al., 2012] to a broader class of semidefinite programs, including *MaxCut*. We briefly recall its simple construction below, applied to a barrier version of the *MaxCut* relaxation PhaseCut, written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(UM) - \mu\log\det(U) \\ \text{subject to} & \mathbf{diag}(U) = 1 \end{array} \tag{7}$$

which is a semidefinite program in the matrix $U \in \mathbf{H}_n$, where $\mu > 0$ is the barrier parameter. As in interior point algorithms, the barrier enforces positive semidefiniteness and the value of $\mu > 0$ precisely controls the

distance between the optimal solution to (7) and the optimal set of PhaseCut. We refer the reader to [Boyd and Vandenberghe, 2004] for further details. The key to applying coordinate descent methods to problems penalized by the $\log\det(\cdot)$ barrier is the following block-determinant formula

$$\det(U) = \det(B)\,\det(y - x^T B^{-1} x), \quad \text{when} \quad U = \begin{pmatrix} B & x \\ x^T & y \end{pmatrix}, \quad U \succ 0. \tag{8}$$

This means that, all other parameters being fixed, minimizing the function $\det(X)$ in the row and column block of variables x, is equivalent to minimizing the quadratic form $y - x^T Z^{-1} x$, arguably a much simpler problem. Solving the semidefinite program (7) row/column by row/column thus amounts to solving the simple problem (9) described in the following lemma.

Lemma 3.1. *Suppose $\sigma > 0$, $c \in \mathbb{R}^{n-1}$, and $B \in \mathbf{S}_{n-1}$ are such that $b \neq 0$ and $B \succ 0$, then the optimal solution of the block problem*

$$\min_x \; c^T x - \sigma \log(1 - x^T B^{-1} x) \tag{9}$$

is given by

$$x = \frac{\sqrt{\sigma^2 + \gamma} - \sigma}{\gamma} Bc$$

where $\gamma = c^T B c$.

Proof. As in [Wen et al., 2012], a direct consequence of the first order optimality conditions for (9). ■

Here, we see problem (7) as an unconstrained minimization problem over the off-diagonal coefficients of U, and (8) shows that each block iteration amounts to solving a minimization subproblem of the form (9). Lemma 3.1 then shows that this is equivalent to computing a matrix vector product. Linear convergence of the algorithm is guaranteed by the result in [Boyd and Vandenberghe, 2004, §9.4.3] and the fact that the function $\log\det$ is strongly convex over compact subsets of the positive semidefinite cone. So the complexity of the method is bounded by $O\left(\log \frac{1}{\epsilon}\right)$ but the constant in this bound depends on n here, and the dependence cannot be quantified explicitly.

Algorithm 3 Block Coordinate Descent Algorithm for **PhaseCut**.

Input: An initial $X^0 = \mathbf{I}_n$ and $\nu > 0$ (typically small). An integer $N > 1$.

1: **for** $k = 1, \ldots, N$ **do**

2: Pick $i \in [1, n]$.

3: Compute

$$x = X^k_{i^c, i^c} M_{i^c, i} \quad \text{and} \quad \gamma = x^* M_{i^c, i}$$

4: If $\gamma > 0$, set

$$X^{k+1}_{i^c, i} = X^{k+1*}_{i, i^c} = -\sqrt{\frac{1-\nu}{\gamma}}\, x$$

else

$$X^{k+1}_{i^c, i} = X^{k+1*}_{i, i^c} = 0.$$

5: **end for**

Output: A matrix $X \succeq 0$ with $\mathbf{diag}(X) = 1$.

3.4. **Initialization & Randomization.** Suppose the Hermitian matrix U solves the semidefinite relaxation PhaseCut. As in [Goemans and Williamson, 2004; Ben-Tal et al., 2003; Zhang and Huang, 2006; So et al., 2007], we generate complex Gaussian vectors $x \in \mathbb{C}^n$ with $x \sim \mathcal{N}_{\mathbb{C}}(0, U)$, and for each sample x, we form $z \in \mathbb{C}^n$ such that

$$z_i = \frac{x_i}{|x_i|}, \quad i = 1, \ldots, n.$$

All the sample points z generated using this procedure satisfy $|z_i| = 1$, hence are feasible points for problem (2). This means in particular that $QP(M) \leq \mathbf{E}[z^*Mz]$. In fact, this expectation can be computed almost explicitly, using

$$\mathbf{E}[zz^*] = F(U), \quad \text{with} \quad F(w) = \tfrac{1}{2}e^{i\arg(w)} \int_0^\pi \cos(\theta)\arcsin(|w|\cos(\theta))d\theta$$

where $F(U)$ is the matrix with coefficients $F(U_{ij})$, $i, j = 1, \ldots, n$. We then get

$$SDP(M) \leq QP(M) \leq \mathbf{Tr}(MF(U)) \tag{10}$$

In practice, to extract good candidate solutions from the solution U to the SDP relaxation in PhaseCut, we sample a few points from $\mathcal{N}_{\mathbb{C}}(0, U)$, normalize their coordinates and simply pick the point which minimizes z^*Mz.

This sampling procedure also suggests a simple spectral technique for computing rough solutions to problem PhaseCut: compute an eigenvector of M corresponding to its lowest eigenvalue and simply normalize its coordinates (this corresponds to the simple bound on *MaxCut* by [Delorme and Poljak, 1993]). The information contained in U can also be used to solve a robust formulation [Ben-Tal et al., 2009] of problem (1) given a Gaussian model $u \sim \mathcal{N}_{\mathbb{C}}(0, U)$.

3.5. **Approximation Bounds.** The semidefinite program in PhaseCut is a *MaxCut*-type graph partitioning relaxation whose performance has been studied extensively. Note however that most approximation results for *MaxCut* study *maximization* problems over positive semidefinite or nonnegative matrices, while we are *minimizing* in PhaseCut so, as pointed out in [Kisialiou and Luo, 2010; So and Ye, 2010] for example, we do not inherit the constant approximation ratios that hold in the classical *MaxCut* setting.

3.6. **Exploiting Structure.** In some instances, we have additional structural information on the solution of problems (1) and (2), which usually reduces the complexity of approximating PhaseCut and improves the quality of the approximate solutions. We briefly highlight a few examples below.

3.6.1. *Symmetries.* In some cases, e.g. signal processing examples where the signal is symmetric, the optimal solution u has a known symmetry pattern. For example, we might have $u(k_- - i) = u(k_+ + i)$ for some k_-, k_+ and indices $i \in [0, k_- - 1]$. This means that the solution u to problem (1) can be written $u = Pv$, where $v \in \mathbb{C}^q$ with $q < n$, and we can solve (1) by focusing on the smaller problem

$$\begin{array}{ll} \text{minimize} & v^*P^*MPv \\ \text{subject to} & |(Pv)_i| = 1, \quad i = 1, \ldots n, \end{array}$$

in the variable $v \in \mathbb{C}^q$. We reconstruct a solution u to (1) from a solution v to the above problem as $u = Pv$. This produces significant computational savings.

3.6.2. *Alignment.* In other instances, we might have prior knowledge that the phases of certain samples are aligned, i.e. that there is an index set I such that $u_i = u_j, \quad$ for all $i, j \in I$, this reduces to the symmetric case discussed above when the phase is arbitrary. W.l.o.g., we can also fix the phase to be one, with $u_i = 1$ for $i \in I$, and solve a constrained version of the relaxation PhaseCut

$$\begin{array}{ll} \text{min.} & \mathbf{Tr}(UM) \\ \text{subject to} & U_{ij} = 1, \quad i, j \in I, \\ & \mathbf{diag}(U) = 1,\ U \succeq 0, \end{array}$$

which is a semidefinite program in $U \in \mathbf{H}_n$.

3.6.3. *Fast Fourier transform.* If the product Mx can be computed with a fast algorithm in $O(n \log n)$ operations, which is the case for Fourier or wavelet transform operators, we significantly speed up the iterations of Algorithm 3 to update all coefficients at once. Each iteration of the modified Algorithm 3 then has cost $O(n \log n)$ instead of $O(n^2)$.

3.6.4. *Real valued signal.* In some cases, we know that the solution vector x in (1) is real valued. Problem (1) can be reformulated to explicitly constrain the solution to be real, by writing it

$$\min_{\substack{u\in\mathbb{C}^n,\,|u_i|=1,\\ x\in\mathbb{R}^p}} \|Ax - \mathbf{diag}(b)u\|_2^2$$

or again, using the operator $\mathcal{T}(\cdot)$ defined in (4)

$$\begin{array}{ll} \text{minimize} & \left\| \mathcal{T}(A) \begin{pmatrix} x \\ 0 \end{pmatrix} - \mathbf{diag} \begin{pmatrix} b \\ b \end{pmatrix} \begin{pmatrix} \mathrm{Re}(u) \\ \mathrm{Im}(u) \end{pmatrix} \right\|_2^2 \\ \text{subject to} & u \in \mathbb{C}^n,\ |u_i| = 1 \\ & x \in \mathbb{R}^p. \end{array}$$

The optimal solution of the inner minimization problem in x is given by $x = A_2^\dagger B_2 v$, where

$$A_2 = \begin{pmatrix} \mathrm{Re}(A) \\ \mathrm{Im}(A) \end{pmatrix}, \quad B_2 = \mathbf{diag} \begin{pmatrix} b \\ b \end{pmatrix}, \quad \text{and} \quad v = \begin{pmatrix} \mathrm{Re}(u) \\ \mathrm{Im}(u) \end{pmatrix}$$

hence the problem is finally rewritten

$$\begin{array}{ll} \text{minimize} & \|(A_2 A_2^\dagger B_2 - B_2)v\|_2^2 \\ \text{subject to} & v_i^2 + v_{n+i}^2 = 1, \quad i = 1, \ldots, n, \end{array}$$

in the variable $v \in \mathbb{R}^{2n}$. This can be relaxed as above by the following problem

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(V M_2) \\ \text{subject to} & V_{ii} + V_{n+i,n+i} = 1, \quad i = 1, \ldots, n, \\ & V \succeq 0, \end{array}$$

which is a semidefinite program in the variable $V \in \mathbf{S}_{2n}$, where $M_2 = (A_2 A_2^\dagger B_2 - B_2)^T (A_2 A_2^\dagger B_2 - B_2) = B_2^T (\mathbf{I} - A_2 A_2^\dagger) B_2$.

4. Matrix completion & exact recovery conditions

In [Chai et al., 2011; Candes et al., 2011b], phase recovery (1) is cast as a matrix completion problem. We briefly review this approach and compare it with the semidefinite program in PhaseCut. Given a signal vector $b \in \mathbb{R}^n$ and a sampling matrix $A \in \mathbb{C}^{n\times p}$, we look for a vector $x \in \mathbb{C}^p$ satisfying

$$|a_i^* x| = b_i, \quad i = 1, \ldots, n,$$

where the vector a_i^* is the i^{th} row of A and $x \in \mathbb{C}^p$ is the signal we are trying to reconstruct. The phase recovery problem is then written as

$$\begin{array}{ll} \text{minimize} & \mathbf{Rank}(X) \\ \text{subject to} & \mathbf{Tr}(a_i a_i^* X) = b_i^2, \quad i = 1, \ldots, n \\ & X \succeq 0 \end{array}$$

in the variable $X \in \mathbf{H}_p$, where $X = xx^*$ when exact recovery occurs. This last problem can be relaxed as

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(X) \\ \text{subject to} & \mathbf{Tr}(a_i a_i^* X) = b_i^2, \quad i = 1, \ldots, n \\ & X \succeq 0 \end{array} \qquad \text{(PhaseLift)}$$

which is a semidefinite program (called PhaseLift by Candes et al. [2011b]) in the variable $X \in \mathbf{H}_p$. Recent results in [Candes et al., 2011a; Candes and Li, 2012] give explicit (if somewhat stringent) conditions on A and x under which the relaxation is tight (i.e. the optimal X in PhaseLift is unique, has rank one, with leading eigenvector x).

4.1. **Weak Formulation.** We also introduce a weak version of PhaseLift, which is more directly related to PhaseCut and is easier to interpret geometrically. It was noted in [Candes et al., 2011a] that, when $\mathbf{I} \in \text{span}\{a_i a_i^*\}_{i=1}^n$, the condition $\mathbf{Tr}(a_i a_i^* X) = b_i^2, i = 1, ..., n$ determines $\mathbf{Tr}(X)$, so in this case the trace minimization objective is redundant and PhaseLift is equivalent to

$$\begin{array}{ll} \text{find} & X \\ \text{subject to} & \mathbf{Tr}(a_i a_i^* X) = b_i^2, \quad i = 1, \dots, n \\ & X \succeq 0. \end{array} \tag{Weak PhaseLift}$$

When $\mathbf{I} \notin \text{span}\{a_i a_i^*\}_{i=1}^n$ on the other hand, Weak PhaseLift and PhaseLift are not equivalent: solutions of PhaseLift solve Weak PhaseLift too but the converse is not true. Interior point solvers typically pick a solution at the analytic center of the feasible set of Weak PhaseLift which in general can be significantly different from the minimum trace solution.

However, in practice, the removal of trace minimization does not really seem to alter the performances of the algorithm. We will illustrate this affirmation with numerical experiments in §5.4 and a formal proof is given in [Demanet and Hand, 2012] who showed that, in the case of Gaussian random measurements, the relaxation of Weak PhaseLift was tight with high probability under the same conditions as PhaseLift.

4.2. **Phase Recovery as a Projection.** We will see in what follows that phase recovery can interpreted as a projection problem. These results will prove useful later to study stability. The PhaseCut reconstruction problem defined in PhaseCut is written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(UM) \\ \text{subject to} & \mathbf{diag}(U) = 1,\ U \succeq 0, \end{array}$$

with $M = \mathbf{diag}(b)(\mathbf{I} - AA^\dagger)\,\mathbf{diag}(b)$. In what follows, we assume $b_i \neq 0$, $i = 1, ..., n$, which means that, after scaling U, solving PhaseCut is equivalent to solving

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(V(\mathbf{I} - AA^\dagger)) \\ \text{subject to} & \mathbf{diag}(V) = b^2, V \succeq 0. \end{array} \tag{11}$$

In the following lemma, we show that this last semidefinite program can be understood as a projection problem on a section of the semidefinite cone using the trace (or nuclear) norm. We define

$$\mathcal{F} = \{V \in \mathbf{H}_n : x^* V x = 0, \forall x \in \mathcal{R}(A)^\perp\}$$

which is also $\mathcal{F} = \{V \in \mathbf{H}_n : (\mathbf{I} - AA^\dagger)V(\mathbf{I} - AA^\dagger) = 0\}$, and we now formulate the objective of problem (11) as a distance.

Lemma 4.1. *For all $V \in \mathbf{H}_n$ such that $V \succeq 0$,*

$$\mathbf{Tr}(V(\mathbf{I} - AA^\dagger)) = d_1(V, \mathcal{F}) \tag{12}$$

where d_1 is the distance associated to the trace norm.

Proof. Let $\mathcal{B}_1$ (resp. $\mathcal{B}_2$) be an orthonormal basis of range A (resp. $(\text{range } A)^\perp$). Let T be the transformation matrix from canonical basis to orthonormal basis $\mathcal{B}_1 \cup \mathcal{B}_2$. Then

$$\mathcal{F} = \{V \in \mathbf{H}_n \text{ s.t. } T^{-1}VT = \begin{pmatrix} S_1 & S_2 \\ S_2^* & 0 \end{pmatrix}, S_1 \in \mathbf{H}_p, S_2 \in \mathcal{M}_{p,n-p}\}$$

As the transformation $X \to T^{-1}XT$ preserves the nuclear norm, for every matrix $V \succeq 0$, if we write

$$T^{-1}VT = \begin{pmatrix} V_1 & V_2 \\ V_2^* & V_3 \end{pmatrix}$$

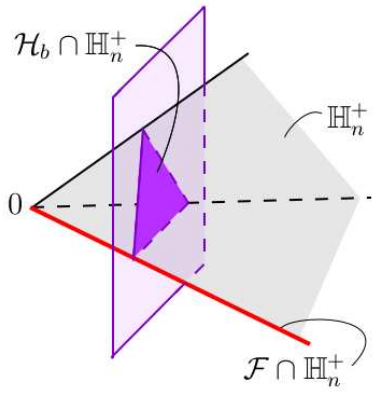

FIGURE 1. Schematic representation of the sets involved in equations (13) and (14) : the cone of positive hermitian matrices $\mathbb{H}_n^+$ (in light grey), its intersection with the affine subspace $\mathcal{H}_b$, and $\mathcal{F} \cap \mathbb{H}_n^+$, which is a face of $\mathbb{H}_n^+$.

then the orthogonal projection of V onto $\mathcal{F}$ is

$$W = T \begin{pmatrix} V_1 & V_2 \\ V_2^* & 0 \end{pmatrix} T^{-1},$$

so $d_1(V, \mathcal{F}) = \|V - W\|_1 = \| \begin{pmatrix} 0 & 0 \\ 0 & V_3 \end{pmatrix} \|_1$. As $V \succeq 0$, $\begin{pmatrix} V_1 & V_2 \\ V_2^* & V_3 \end{pmatrix} \succeq 0$ hence $\begin{pmatrix} 0 & 0 \\ 0 & V_3 \end{pmatrix} \succeq 0$, so $d_1(V, \mathcal{F}) = \mathbf{Tr} \begin{pmatrix} 0 & 0 \\ 0 & V_3 \end{pmatrix}$. Because $AA^\dagger$ is the orthogonal projection onto $\mathcal{R}(A)$, we have $T^{-1}(\mathbf{I} - AA^\dagger)T = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{I} \end{pmatrix}$ hence

$$d_1(V, \mathcal{F}) = \mathbf{Tr} \begin{pmatrix} 0 & 0 \\ 0 & V_3 \end{pmatrix} = \mathbf{Tr}((T^{-1}VT)(T^{-1}(\mathbf{I} - AA^\dagger)T)) = \mathbf{Tr}(V(\mathbf{I} - AA^\dagger))$$

which is the desired result. ■

This means that PhaseCut can be written as a projection problem, i.e.

$$\begin{array}{ll} \text{minimize} & d_1(V, \mathcal{F}) \\ \text{subject to} & V \in \mathbf{H}_n^+ \cap \mathcal{H}_b \end{array} \tag{13}$$

in the variable $V \in \mathbf{H}_n$, where $\mathcal{H}_b = \{V \in \mathbf{H}_n \text{ s.t. } V_{i,i} = b_i^2, i = 1, ..., n\}$. Moreover, with a_i the i-th row of A, we have for all $X \in \mathbf{H}_p^+$, $\mathbf{Tr}(a_i a_i^* X) = a_i^* X a_i = \mathbf{diag}(AXA^*)_i, \quad i = 1, \ldots, n$, so if we call $V = AXA^* \in \mathcal{F}$, when A is injective, $X = A^\dagger V A^{\dagger *}$ and Weak PhaseLift is equivalent to

$$\begin{array}{ll} \text{find} & V \in \mathbf{H}_n^+ \cap \mathcal{F} \\ \text{subject to} & \mathbf{diag}(V) = b^2. \end{array}$$

First order algorithms for Weak PhaseLift will typically solve

$$\begin{array}{ll} \text{minimize} & d(\mathbf{diag}(V), b^2) \\ \text{subject to} & V \in \mathbf{H}_n^+ \cap \mathcal{F} \end{array}$$

for some distance d over $\mathbb{R}^n$. If d is the l^s-norm, for any $s \geq 1$, $d(\mathbf{diag}(V), b^2) = d_s(V, \mathcal{H}_b)$, where d_s is the distance generated by the Schatten s-norm, the algorithm becomes

$$\begin{array}{ll} \text{minimize} & d_s(V, \mathcal{H}_b) \\ \text{subject to} & V \in \mathbf{H}_n^+ \cap \mathcal{F} \end{array} \tag{14}$$

which is another projection problem in V.

Thus, PhaseCut and Weak PhaseLift are comparable, in the sense that both algorithms aim at finding a point of $\mathbb{H}_n^+ \cap \mathcal{F} \cap \mathcal{H}_b$ but PhaseCut does so by picking a point of $\mathbb{H}_n^+ \cap \mathcal{H}_b$ and moving towards $\mathcal{F}$ while Weak PhaseLift moves a point of $\mathbb{H}_n^+ \cap \mathcal{F}$ towards $\mathcal{H}_b$. We can push the parallel between both relaxations much further. We will show in what follows that, in a very general case, PhaseLift and a modified version

of PhaseCut are simultaneously tight. We will also be able to compare the stability of Weak PhaseLift and PhaseCut when measurements become noisy.

4.3. **Tightness of the Semidefinite Relaxation.** We will now formulate a refinement of the semidefinite relaxation in PhaseCut and prove that this refinement is equivalent to the relaxation in PhaseLift under mild technical assumptions. Suppose u is the optimal phase vector, we know that the optimal solution to (1) can then be written $x = A^\dagger \mathbf{diag}(b)u$, which corresponds to the matrix $X = A^\dagger \mathbf{diag}(b)uu^* \mathbf{diag}(b)A^{\dagger *}$ in PhaseLift, hence

$$\mathbf{Tr}(X) = \mathbf{Tr}(\mathbf{diag}(b)A^{\dagger *}A^\dagger \mathbf{diag}(b)uu^*).$$

Writing $B = \mathbf{diag}(b)A^{\dagger *}A^\dagger \mathbf{diag}(b)$, when problem (1) is solvable, we look for the "minimum trace" solution among all the optimal points of relaxation PhaseCut by solving

$$\begin{array}{rlll} SDP2(M) \triangleq & \text{min.} & \mathbf{Tr}(BU) & \\ & \text{subject to} & \mathbf{Tr}(MU) = 0 & \text{(PhaseCutMod)} \\ & & \mathbf{diag}(U) = 1,\, U \succeq 0, & \end{array}$$

which is a semidefinite program in $U \in \mathbf{H}_n$. When problem (1) is solvable, then every optimal solution of the semidefinite relaxation PhaseCut is a feasible point of relaxation PhaseCutMod. In practice, the semidefinite program $SDP(M + \gamma B)$, written

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}((M + \gamma B)U) \\ \text{subject to} & \mathbf{diag}(U) = 1,\, U \succeq 0, \end{array}$$

obtained by replacing M by $M + \gamma B$ in problem PhaseCut, will produce a solution to PhaseCutMod whenever $\gamma > 0$ is sufficiently small (this is essentially the exact penalty method detailed in [Bertsekas, 1998, §4.3] for example). This means that all algorithms (greedy or SDP) designed to solve the original PhaseCut problem can be recycled to solve PhaseCutMod with negligible effect on complexity. We now show that the PhaseCutMod and PhaseLift relaxations are simultaneously tight when A is injective. An earlier version of this paper showed PhaseLift tightness implies PhaseCut tightness and the argument was reversed in [Voroninski, 2012] under mild additional assumptions.

Proposition 4.2. *Assume that $b_i \neq 0$ for $i = 1, \ldots, n$, that A is injective and that there is a solution x to (1). The function*

$$\begin{array}{rcl} \Phi : \mathbf{H}_p & \to & \mathbf{H}_n \\ X & \mapsto & \Phi(X) = \mathbf{diag}(b)^{-1} AXA^* \mathbf{diag}(b)^{-1} \end{array}$$

is a bijection between the feasible points of PhaseCutMod and those of PhaseLift.

Proof. Note that Φ is injective whenever $b > 0$ and A has full rank. We have to show that U is a feasible point of PhaseCutMod if and only if it can be written under the form $\Phi(X)$, where X is feasible for PhaseLift. We first show that

$$\mathbf{Tr}(MU) = 0, \quad U \succeq 0, \tag{15}$$

is equivalent to

$$U = \Phi(X) \tag{16}$$

for some $X \succeq 0$. Observe that $\mathbf{Tr}(UM) = 0$ means $UM = 0$ because $U, M \succeq 0$, hence $\mathbf{Tr}(MU) = 0$ in (15) is equivalent to

$$AA^\dagger \mathbf{diag}(b)U \mathbf{diag}(b) = \mathbf{diag}(b)U \mathbf{diag}(b)$$

because $b > 0$ and $M = \mathbf{diag}(b)(\mathbf{I} - AA^\dagger) \mathbf{diag}(b)$. If we set $X = A^\dagger \mathbf{diag}(b)U \mathbf{diag}(b)A^{\dagger *}$, this last equality implies both

$$AX = AA^\dagger \mathbf{diag}(b)U \mathbf{diag}(b)A^{\dagger *} = \mathbf{diag}(b)U \mathbf{diag}(b)A^{\dagger *}$$

and

$$AXA^* = \mathbf{diag}(b)U \mathbf{diag}(b)A^{\dagger *}A^* = \mathbf{diag}(b)U \mathbf{diag}(b)$$

which is $U = \Phi(X)$, and shows (15) implies (16). Conversely, if $U = \Phi(X)$ then $\mathbf{diag}(b)U\,\mathbf{diag}(b) = AXA^*$ and using $AA^\dagger A = A$, we get $AXA^* = AA^\dagger AXA^* = AA^\dagger\,\mathbf{diag}(b)U\,\mathbf{diag}(b)$ which means $MU = 0$, hence (15) is in fact equivalent to (16) since $U \succeq 0$ by construction.

Now, if X is feasible for PhaseLift, we have shown $\mathbf{Tr}(M\Phi(X)) = 0$ and $\phi(X) \succeq 0$, moreover $\mathbf{diag}(\Phi(X))_i = \mathbf{Tr}(a_i a_i^* X)/b_i^2 = 1$, so $U = \Phi(X)$ is a feasible point of PhaseCutMod. Conversely, if U is feasible for PhaseCutMod, we have shown that there exists $X \succeq 0$ such that $U = \Phi(X)$ which means $\mathbf{diag}(b)U\,\mathbf{diag}(b) = AXA^*$. We also have $\mathbf{Tr}(a_i a_i^* X) = b_i^2 U_{ii} = b_i^2$, which means X is feasible for PhaseLift and concludes the proof. ■

We now have the following central corollary showing the equivalence between PhaseCutMod and PhaseLift in the noiseless case.

Corollary 4.3. *If A is injective, $b_i \neq 0$ for all $i = 1, ..., n$ and if the reconstruction problem (1) admits an exact solution, then PhaseCutMod is tight (i.e. has a unique rank one solution) whenever PhaseLift is.*

Proof. When A is injective, $\mathbf{Tr}(X) = \mathbf{Tr}(B\Phi(X))$ and $\mathbf{Rank}(X) = \mathbf{Rank}(\Phi(X))$. ■

This last result shows that in the noiseless case, the relaxations PhaseLift and PhaseCutMod are in fact equivalent. In the same way, we could have shown that Weak PhaseLift and PhaseCut were equivalent. The performances of both algorithms may not match however when the information on b is noisy and perfect recovery is not possible.

Remark 4.4. *Note that Proposition 4.2 and corollary 4.3 also hold when the initial signal is real and the measurements are complex. In this case, we define the B in PhaseCutMod by $B = B_2 A_2^{\dagger *} A_2^\dagger B_2$ (with the notations of paragraph 3.6.4). We must also replace the definition of Φ by $\Phi(X) = B_2^{-1} A_2 X A_2^* B_2^{-1}$. Furthermore, all steps in the proof of proposition 4.2 are still valid if we replace M by M_2, A by A_2 and $\mathbf{diag}(b)$ by B_2. The only difference is that now $\frac{1}{b_i^2}\mathbf{Tr}(a_i a_i^* X) = \mathbf{diag}(\Phi(X))_i + \mathbf{diag}(\Phi(X))_{n+i}$.*

4.4. **Stability in the Presence of Noise.** We now consider the case where the vector of measurements b is of the form $b = |Ax_0| + b_{\text{noise}}$. We first introduce a definition of C-stability for PhaseCut and Weak PhaseLift. The main result of this section is that, when the Weak PhaseLift solution in (14) is stable at a point, PhaseCut is stable too, with a constant of the same order. The converse does not seem to be true when b is sparse.

Definition 4.5. *Let $x_0 \in \mathbb{C}^n, C > 0$. The algorithm PhaseCut (resp. Weak PhaseLift) is said to be C-stable at x_0 iff for all $b_{\text{noise}} \in \mathbb{R}^n$ close enough to zero, every minimizer V of equation (13) (resp. (14)) with $b = |Ax_0| + b_{\text{noise}}$, satisfies*

$$\|V - (Ax_0)(Ax_0)^*\|_2 \leq C\|Ax_0\|_2\|b_{\text{noise}}\|_2.$$

The following matrix perturbation result motivates this definition, by showing that a C-stable algorithm generates a $O(C\|b_{\text{noise}}\|_2)$-error over the signal it reconstructs.

Proposition 4.6. *Let $C > 0$ be arbitrary. We suppose that $Ax_0 \neq 0$ and $\|V - (Ax_0)(Ax_0)^*\|_2 \leq C\|Ax_0\|_2\|b_{\text{noise}}\|_2 \leq \|Ax_0\|_2^2/2$. Let y be V's main eigenvector, normalized so that $(Ax_0)^* y = \|Ax_0\|_2$. Then*

$$\|y - Ax_0\|_2 = O(C\|b_{\text{noise}}\|_2),$$

and the constant in this last equation does not depend upon A, x_0, C or $\|b\|_2$.

Proof. We use [El Karoui and d'Aspremont, 2009, Eq.10] for

$$u = \frac{Ax_0}{\|Ax_0\|_2} \qquad v = \frac{y}{\|Ax_0\|_2} \qquad E = \frac{V - (Ax_0)(Ax_0)^*}{\|Ax_0\|_2^2}$$

This result is based on [Kato, 1995, Eq. 3.29], which gives a precise asymptotic expansion of $u - v$. For our purposes here, we only need the first-order term. See also Bhatia [1997], Stewart and Sun [1990] or Stewart [2001] among others for a complete discussion. We get $\|v - u\| = O(\|E\|_2)$ because if $M = uu^*$, then $\|R\|_\infty = 1$ in [El Karoui and d'Aspremont, 2009, Eq.10]. This implies

$$\|y - Ax_0\|_2 = \|Ax_0\|_2\|u - v\| = O\left(\frac{\|V - (Ax_0)(Ax_0)^*\|_2}{\|Ax_0\|_2}\right) = O(C\|b_{\text{noise}}\|)$$

which is the desired result. ■

Note that normalizing y differently, we would obtain $\|y - Ax_0\|_2 \leq 4C\|b_{\text{noise}}\|_2$. We now show the main result of this section, according to which PhaseCut is "almost as stable as" Weak PhaseLift. In practice of course, the exact values of the stability constants has no importance, what matters is that they are of the same order.

Theorem 4.7. *Let $A \in \mathbb{C}^{n\times m}$, for all $x_0 \in \mathbb{C}^n, C > 0$, if Weak PhaseLift is C-stable in x_0, then PhaseCut is $(2C + 2\sqrt{2} + 1)$-stable in x_0.*

Proof. Let $x_0 \in \mathbb{C}^n, C > 0$ be such that Weak PhaseLift is C-stable in x_0. Ax_0 is a non-zero vector (because, with our definition, neither Weak PhaseLift nor PhaseCut may be stable in x_0 if $Ax_0 = 0$ and $A \neq 0$). We set $D = 2C + 2\sqrt{2} + 1$ and suppose by contradiction that PhaseCut is not D-stable in x_0. Let $\epsilon > 0$ be arbitrary. Let $b_{\text{n,PC}} \in \mathbb{R}^n$ be such that $\|b_{\text{n,PC}}\|_2 \leq \max(\|Ax_0\|_2, \epsilon/2)$ and such that, for $b = |Ax_0| + b_{\text{n,PC}}$, the minimizer V_{PC} of (13) verifies

$$\|V_{PC} - (Ax_0)(Ax_0)^*\|_2 > D\|Ax_0\|_2\|b_{\text{n,PC}}\|_2$$

Such a V_{PC} must exist or PhaseCut would be D-stable in x_0. We call $V_{PC}^{/\!/}$ the restriction of V_{PC} to range(A) (that is, the matrix such that $x^*(V_{PC}^{/\!/})y = x^*(V_{PC})y$ if $x, y \in$ range(A) and $x^*(V_{PC}^{/\!/})y = 0$ if $x \in$ range$(A)^\perp$ or $y \in$ range$(A)^\perp$) and $V_{PC}^\perp$ the restriction of V_{PC} to range$(A)^\perp$. Let us set $b_{\text{n,PL}} = \sqrt{V_{PC\,ii}^{/\!/}} - |Ax_0|_{ii}$ for $i = 1, ..., n$. As $V_{PC}^{/\!/} \in \mathbf{H}_n^+ \cap \mathcal{F}$, $V_{PC}^{/\!/}$ minimizes (14) for $b = |Ax_0| + b_{\text{n,PL}}$ (because $V_{PC}^{/\!/} \in \mathcal{H}_b$). Lemmas A.1 and A.2 (proven in the appendix) imply that $\|V_{PC}^{/\!/} - (Ax_0)(Ax_0)^*\|_2 > C\|Ax_0\|_2\|b_{\text{n,PL}}\|_2$ and $\|b_{\text{n,PL}}\|_2 \leq \epsilon$. As ϵ is arbitrary, Weak PhaseLift is not C-stable in x_0, which contradicts our hypotheses. Consequently, PhaseCut is $(2C + 2\sqrt{2} + 1)$-stable in x_0. ■

Theorem 4.7 is still true if we replace $2C + 2\sqrt{2} + 1$ by any $D > 2C + \sqrt{2}$. We only have to replace, in the demonstration, the inequality $\|b_{\text{n,PC}}\|_2 \leq \|Ax_0\|_2$ by $\|b_{\text{n,PC}}\|_2 \leq \alpha\|Ax_0\|_2$ with $\alpha = D - (2C + \sqrt{2})/(1 + \sqrt{2})$. Also, the demonstration of this theorem is based on the fact that, when V_{PC} solves (13), one can construct some $V_{PL} = V_{PC}^{/\!/}$ close to V_{PC}, which is an approximate solution of (14). It is natural to wonder whether, conversely, from a solution V_{PL} of (14), one can construct an approximate solution V_{PC} of (13). It does not seem to be the case. One could for example imagine setting $V_{PC} = \mathbf{diag}(R)V_{PL}\,\mathbf{diag}(R)$, where $R_i = b_i/\sqrt{V_{PL\,ii}}$. Then V_{PC} would not necessarily minimize (13) but at least belong to $\mathcal{H}_b$. But $\|V_{PC} - V_{PL}\|_2$ might be quite large: (14) implies that $\|\,\mathbf{diag}(V_{PL}) - b^2\|_s$ is small but, if some coefficients of b are very small, some R_i may still be huge, so $\mathbf{diag}(R) \not\approx \mathbf{I}$. This does happen in practice (see § 5.5).

To conclude this section, we relate this definition of stability to the one introduced in [Candes and Li, 2012]. Suppose that A is a matrix of random gaussian independant measurements such that $\mathbf{E}[|A_{i,j}|^2] = 1$ for all i, j. We also suppose that $n \geq c_0 p$ (for some c_0 independent of n and p). In the noisy setting, Candes and Li [2012] showed that the minimizer X of a modified version of PhaseLift satisfies with high probability

$$||X - x_0x_0^*||_2 \leq C_0\frac{||\,|Ax_0|^2 - b^2\,||_1}{n} \tag{17}$$

for some C_0 independent of all variables. Assuming that the Weak PhaseLift solution in (14) behaves as PhaseLift in a noisy setting and that (17) also holds for Weak PhaseLift, then

$$\begin{aligned}||AXA^* - (Ax_0)(Ax_0)^*||_2 &\leq ||A||_\infty^2 ||X - x_0x_0^*||_2 \\ &\leq C_0 \frac{||A||_\infty^2}{n} || \, |Ax_0|^2 - b^2 \, ||_1 \\ &\leq C_0 \frac{||A||_\infty^2}{n} (2||Ax_0||_2 + ||b_{\text{noise}}||_2)||b_{\text{noise}}||_2\end{aligned}$$

Consequently, for any $C > 2C_0 \frac{||A||_\infty^2}{n}$, Weak PhaseLift is C-stable in all x_0. With high probability, $||A||_\infty^2 \leq (1 + 1/8)n$ (it is a corollary of [Candes and Li, 2012, Lemma 2.1]) so Weak PhaseLift (and thus also PhaseCut) is C-stable with high probability for some C independent of all parameters of the problem.

4.5. **Perturbation Results.** We recall here sensitivity analysis results for semidefinite programming from Todd and Yildirim [2001]; Yildirim [2003], which produce explicit bounds on the impact of small perturbations in the observation vector b^2 on the solution V of the semidefinite program (11). Roughly speaking, these results show that if $b^2 + b_{noise}$ remains in an explicit ellipsoid (called Dikin's ellipsoid), then interior point methods converge back to the solution in one full Newton step, hence the impact on V is linear, equal to the Newton step. These results are more numerical in nature than the stability bounds detailed in the previous section, but they precisely quantify both the size and, perhaps more importantly, the geometry of the stability region.

4.6. **Complexity Comparisons.** Both the relaxation in PhaseLift and that in PhaseCut are semidefinite programs and we highlight below the relative complexity of solving these problems depending on algorithmic choices and precision targets. Note that, in their numerical experiments, [Candes et al., 2011b] solve a penalized formulation of problem PhaseLift, written

$$\min_{X \succeq 0} \sum_{i=1}^{n} (\mathbf{Tr}(a_i a_i^* X) - b_i^2)^2 + \lambda \, \mathbf{Tr}(X) \tag{18}$$

in the variable $X \in \mathbf{H}_p$, for various values of the penalty parameter $\lambda > 0$.

The trace norm promotes a low rank solution, and solving a sequence of weighted trace-norm problems has been shown to further reduce the rank in [Fazel et al., 2003; Candes et al., 2011b]. This method replaces $\mathbf{Tr}(X)$ by $\mathbf{Tr}(W_k X)$ where W_0 is initialized to the identity I. Given a solution X_k of the resulting semidefinite program, the weighted matrix is updated to $W_{k+1} = (X_k + \eta I)^{-1}$ (see Fazel et al. [2003] for details). We denote by K the total number of such iterations, typically of the order of 10. Trace minimization is not needed for the semidefinite program (PhaseCut), where the trace is fixed because we optimize over a normalized phase vector. However, weighted trace-norm iterations could potentially improve performance in PhaseCut as well.

Recall that p is the size of the signal and n is the number of measured samples with $n = Jp$ in the examples reviewed in Section 5. In the numerical experiments in [Candes et al., 2011b] as well as in this paper, $J = 3, 4, 5$. The complexity of solving the PhaseCut and PhaseLift relaxations in PhaseLift using generic semidefinite programming solvers such as SDPT3 [Toh et al., 1999], *without exploiting structure*, is given by

$$O\left(J^{4.5} \, p^{4.5} \log \frac{1}{\epsilon}\right) \quad \text{and} \quad O\left(K \, J^2 \, p^{4.5} \log \frac{1}{\epsilon}\right)$$

for PhaseCut and PhaseLift respectively [Ben-Tal and Nemirovski, 2001, § 6.6.3]. The fact that the constraint matrices have only one nonzero coefficient in PhaseCut can be exploited (the fact that the constraints $a_i a_i^*$ are rank one in PhaseLift helps, but it does not modify the principal complexity term) so we get

$$O\left(J^{3.5} \, p^{3.5} \log \frac{1}{\epsilon}\right) \quad \text{and} \quad O\left(K \, J^2 p^{4.5} \log \frac{1}{\epsilon}\right)$$

for PhaseCut and PhaseLift respectively using the algorithm in Helmberg et al. [1996] for example. If we use first-order solvers such as TFOCS [Becker et al., 2012], based on the optimal algorithm in [Nesterov, 1983], the dependence on the dimension can be further reduced, to become

$$O\left(\frac{J^3\,p^3}{\epsilon}\right) \quad \text{and} \quad O\left(\frac{K\,J\,p^3}{\epsilon}\right)$$

for solving a penalized version of the PhaseCut relaxation and the penalized formulation of PhaseLift in (18). While the dependence on the signal dimensions p is somewhat reduced, the dependence on the target precision grows from $\log(1/\epsilon)$ to $1/\epsilon$. Finally, the iteration complexity of the block coordinate descent Algorithm 3 is substantially lower and its convergence is linear, but no fully explicit bounds on the number of iterations are known in our case. The complexity of the method is then bounded by $O\left(\log \frac{1}{\epsilon}\right)$ but the constant in this bound depends on n here, and the dependence cannot be quantified explicitly.

Algorithmic choices are ultimately guided by precision targets. If ϵ is large enough so that a first-order solver or a block coordinate descent can be used, the complexity of PhaseCut is not significantly better than that of PhaseLift. On the contrary, when ϵ is small, we must use an interior point solver, for which PhaseCut's complexity is an order of magnitude lower than that of PhaseLift because its constraint matrices are singletons. In practice, the target value for ϵ strongly depends on the sampling matrix A. For example, when A corresponds to the convolution by 6 Gaussian random filters (§5.2), to reconstruct a Gaussian white noise of size 64 with a relative precision of η, we typically need $\epsilon \sim 2.10^{-1}\eta$. For 4 Cauchy wavelets (§5.3), it is twenty times less, with $\epsilon \sim 10^{-2}\eta$. For other types of signals than Gaussian white noise, we may even need $\epsilon \sim 10^{-3}\eta$.

4.7. **Greedy Refinement.** If the PhaseCut or PhaseLift algorithms do not return a rank one matrix then an approximate solution of the phase recovery problem is obtained by extracting a leading eigenvector v. For PhaseCut and PhaseLift, $\tilde{x} = A^\dagger\,\mathbf{diag}(b)v$ and $\tilde{x} = v$ are respectively approximate solutions of the phase recovery problem with $|A\tilde{x}| \neq b = |Ax|$. This solution is then refined by applying the Gerchberg-Saxton algorithm initialized with $\tilde{x}$. If $\tilde{x}$ is sufficiently close to x then, according to numerical experiments of Section 5, this greedy algorithm converges to $\lambda\,x$ with $|\lambda| = 1$. These greedy iterations require much less operations than PhaseCut and PhaseLift algorithms, and thus have no significant contribution to the computational complexity.

4.8. **Sparsity.** Minimizing $\mathbf{Tr}(X)$ in the PhaseLift problem means looking for signals which match the modulus constraints and have minimum ℓ_2 norm. In some cases, we have a priori knowledge that the signal we are trying to reconstruct is sparse, i.e. $\mathbf{Card}(x)$ is small. The effect of imposing sparsity was studied in e.g. [Moravec et al., 2007; Osherovich et al., 2012; Li and Voroninski, 2012].

Assuming $n \leq p$, the set of solutions to $\|Ax - \mathbf{diag}(b)u\|_2$ is written $x = A^\dagger\,\mathbf{diag}(b)u + Fv$ where F is a basis for the nullspace of A. In this case, when the rows of A are independent, $AA^\dagger = \mathbf{I}$ and the reconstruction problem with a ℓ_1 penalty promoting sparsity is then written

$$\begin{array}{ll} \text{minimize} & \|A^\dagger\,\mathbf{diag}(b)u + Fv\|_1^2 \\ \text{subject to} & |u_i| = 1, \end{array}$$

in the variables $u \in \mathbb{C}^p$ and $y \in \mathbb{C}^{p-n}$. Using the fact that $\|y\|_1^2 = \|yy^*\|_{\ell_1}$, this can be relaxed as

$$\begin{array}{ll} \text{minimize} & \|VUV^*\|_{\ell_1} \\ \text{subject to} & U \succeq 0,\ |U_{ii}| = 1, \quad i = 1, \ldots, n, \end{array}$$

which is a semidefinite program in the (larger) matrix variable $U \in \mathbf{H}_p$ and $V = (A^\dagger\,\mathbf{diag}(b), F)$.

On the other hand, when $n > p$ and A is injective, the matrix F disappears, taking sparsity into account simply amounts to adding an l_1 penalization to PhaseCut. As noted in [Voroninski, 2012] however, the effect of an ℓ_1 penalty on least-squares solutions is not completely clear.

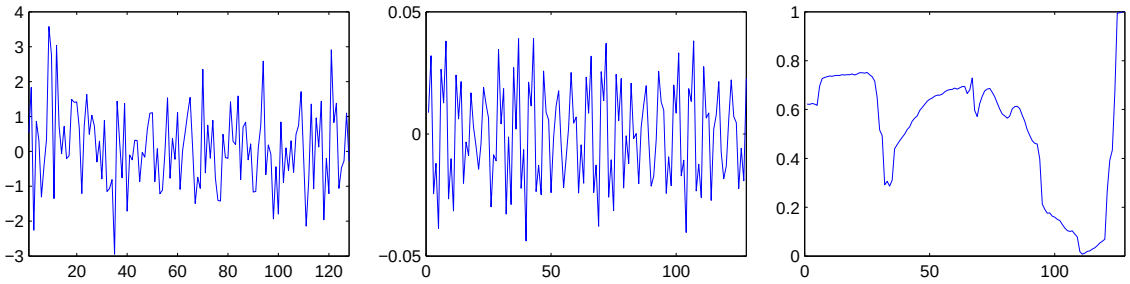

FIGURE 2. Real parts of sample test signals. (a) Gaussian white noise. (b) Sum of 6 sinuoids of random frequency and random amplitudes. (c) Scan-line of an image.

5. NUMERICAL RESULTS

In this section, we compare the numerical performance of the Gerchberg-Saxton (greedy), PhaseCut and PhaseLift algorithms on various phase recovery problems. As in [Candes et al., 2011b], the PhaseLift problem is solved using the package in [Becker et al., 2012], with reweighting, using $K = 10$ outer iterations and 1000 iterations of the first order algorithm. The PhaseCut and Gerchberg-Saxton algorithms described here are implemented in a public software package available at

`http://www.cmap.polytechnique.fr/scattering/code/phaserecovery.zip`

These phase recovery algorithms computes an approximate solution $\tilde{x}$ from $|Ax|$ and the reconstruction error is measured by the relative Euclidean distance up to a complex phase given by

$$\epsilon(x, \tilde{x}) \triangleq \min_{c \in \mathbb{C}, |c|=1} \frac{\|x - c\,\tilde{x}\|}{\|x\|}. \tag{19}$$

We also record the error over measured amplitudes, written

$$\epsilon(|Ax|, |A\tilde{x}|) \triangleq \frac{\||Ax| - |A\tilde{x}|\|}{\|Ax\|}. \tag{20}$$

Note that when the phase recovery problem either does not admit a unique solution or is unstable, we usually have $\epsilon(|Ax|, |A\tilde{x}|) \ll \epsilon(x, \tilde{x})$. In the next three subsections, we study these reconstruction errors for three different phase recovery problems, where A is defined as an oversampled Fourier transform, as multiple filterings with random filters, or as a wavelet transform. Numerical results are computed on three different types of test signals x: realizations of a complex Gaussian white noise, sums of complex exponentials $a_\omega\, e^{i\omega m}$ with random frequencies ω and random amplitudes a_ω (the number of exponentials is random, around 6), and signals whose real and imaginary parts are scan-lines of natural images. Each signal has $p = 128$ coefficients. Figure 2 shows the real part of sample signals, for each signal type.

5.1. **Oversampled Fourier Transform.** The discrete Fourier transform $\hat{y}$ of a signal y of q coefficients is written

$$\hat{y}_k = \sum_{m=0}^{q-1} y_m \exp(\frac{-i2\pi km}{q}) \,.$$

In X-ray crystallography or diffraction imaging experiments, compactly supported signals are estimated from the amplitude of Fourier transforms oversampled by a factor $J \geq 2$. The corresponding operator A computes an oversampled discrete Fourier transform evaluated over $n = Jp$ coefficients. The signal x of size p is extended into x^J by adding $(J-1)p$ zeros and

$$(Ax)_k = \hat{x}^J_k = \sum_{m=1}^{p} x_m \exp(-\frac{i2\pi km}{n}).$$

	Fourier	Random Filters	Wavelets
Gerchberg-Saxton	5%	49%	0%
PhaseLift with reweighting	3%	100%	62%
PhaseCut	4%	100%	100%

TABLE 1. Percentage of perfect reconstruction from $|Ax|$, over 300 test signals, for the three different operators A (columns) and the three algorithms (rows).

	Fourier	Random Filters	Wavelets
Gerchberg-Saxton	0.9	1.2	1.3
PhaseLift with reweighting	0.8	exact	0.5
PhaseCut	0.8	exact	exact

TABLE 2. Average relative signal reconstruction error $\epsilon(\tilde{x}, x)$ over all test signals that are not perfectly reconstructed, for each operator A and each algorithm.

	Fourier	Random Filters	Wavelets
Gerchberg-Saxton	9.10^{-4}	0.2	0.3
PhaseLift with reweighting	5.10^{-4}	exact	8.10^{-2}
PhaseCut	6.10^{-4}	exact	exact

TABLE 3. Average relative error $\epsilon(|A\tilde{x}|, |Ax|)$ of coefficient amplitudes, over all test signals that are not perfectly reconstructed, for each operator A and each algorithm.

For this oversampled Fourier transform, the phase recovery problem does not have a unique solution [Akutowicz, 1956]. In fact, one can show [Sanz, 1985] that there are as many as 2^{p-1} solutions $\tilde{x} \in \mathbb{C}^p$ such that $|A\tilde{x}| = |Ax|$. Moreover, increasing the oversampling factor J beyond 2 does not reduce the number of solutions.

Because of this intrinsic instability, we will observe that all algorithms perform similarly on this type of reconstruction problems and Table 1 shows that the percentage of perfect reconstruction is below 5% for all methods. The signals which are perfectly recovered are sums of few sinusoids. Because these test signals are very sparse in the Fourier domain, the number of signals having identical Fourier coefficient amplitudes is considerably smaller than in typical sample signals. As a consequence, there is a small probability (about 5%) of exactly reconstructing the original signal given an arbitrary initialization. None of the Gaussian random noises and image scan lines are exactly recovered. Note that we say that an exact reconstruction is reached when $\epsilon(x, \tilde{x}) < 10^{-2}$ because a few iterations of the Gerchberg-Saxton algorithm from such an approximate solution $\tilde{x}$ will typically converges to x. Numerical results are computed with 100 sample signals in each of the 3 signal classes.

Table 2 gives the average relative error $\epsilon(x, \tilde{x})$ over signals that are not perfectly reconstructed, which is of order one here. Despite this large error, Table 3 shows that the relative error $\epsilon(|Ax|, |A\tilde{x}|)$ over the Fourier modulus coefficients is below 10^{-3} for all algorithms. This is due to the non-uniqueness of the phase recovery from Fourier modulus coefficients. Recovering a solution $\tilde{x}$ with identical or nearly identical oversampled Fourier modulus coefficients as x does not guarantee that $\tilde{x}$ is proportional to x. Overall, in this set of ill-posed Fourier experiments, recovery performance is very poor for all methods and the PhaseCut and PhaseLift relaxations do not improve much on the results of the faster Gerchberg-Saxton algorithm.

5.2. **Multiple Random Illumination Filters.** To guarantee uniqueness of the phase recovery problem, one can add independent measurements by "illuminating" the object through J filters h^j in the context of X-ray

imaging or crystallography [Candes et al., 2011a]. The resulting operator A is the discrete Fourier transform of x multiplied by each filter h^j of size p

$$(Ax)_{k+pj} = \widehat{(xh^j)}_k = (\widehat{x} \star \hat{h}^j)_k \quad \text{for } 1 \leq j \leq J \text{ and } 0 \leq k < p,$$

where $\hat{x} \star \hat{h}^j$ is the circular convolution between $\hat{x}$ and $\hat{h}^j$. Candes et al. [2011a] proved that, for some constant $C > 0$ large enough, Cp Gaussian independent measurements are sufficient to perfectly reconstruct a signal of size p, with high probability. Similarly, we would expect that, picking the filters h^j as realizations of independent Gaussian random variables, perfect reconstruction will be guaranteed with high probability if J is large enough (and independent of p). This result has not yet been proven because Gaussian filters do not give independent measurements but Candes et al. [2011b] observed that, empirically, for signals of size $p = 128$, with $J = 4$ filters, perfect recovery is achieved in 100% of experiments.

Table 1 confirms this behavior and shows that the PhaseCut algorithm achieves perfect recovery in all our experiments. As predicted by the equivalence results presented in the previous section, we observe that PhaseCut and PhaseLift have identical performance in these experiments. With 4 filters, the solutions of these two SDP relaxations are not of rank one but are "almost" of rank one, in the sense that their first eigenvector v has an eigenvalue much larger than the others, by a factor of about 5 to 10. Numerically, we observe that the corresponding approximate solutions, $\tilde{x} = \mathbf{diag}(v)b$, yield a relative error $\epsilon(|Ax|, |A\tilde{x}|)$ which, for scan-lines of images and especially for Gaussian signals, is of the order of the ratio between the largest and the second largest eigenvalue of the matrix U. The resulting solutions $\tilde{x}$ are then sufficiently close to x so that a few iterations of the Gerchberg-Saxton algorithm started at $\tilde{x}$ will converge to x.

Table 1 shows however that directly applying the Gerchberg-Saxton algorithm starting from a random initialization point yields perfect recovery in only about 50% of our experiments. This percentage decreases as the signal size p increases. The average error $\epsilon(x, \tilde{x})$ on non-recovered signals in Table 2 is 1.3 whereas on the average error on the modulus $\epsilon(|Ax|, |A\tilde{x}|)$ is 0.2.

5.3. **Wavelet Transform.** Phase recovery problems from the modulus of wavelet coefficients appear in audio signal processing where this modulus is used by many audio and speech recognition systems. These moduli also provide physiological models of cochlear signals in the ear [Chi et al., 2005] and recovering audio signals from wavelet modulus coefficients is an important problem in this context.

To simplify experiments, we consider wavelets dilated by dyadic factors 2^j which have a lower frequency resolution than audio wavelets. A discrete wavelet transform is computed by circular convolutions with discrete wavelet filters, i.e.

$$(Ax)_{k+jp} = (x \star \psi^j)_k = \sum_{m=1}^{p} x_m \psi^j_{k-m} \quad \text{for } 1 \leq j \leq J-1 \text{ and } 1 \leq k \leq p$$

where ψ^j_m is a p periodic wavelet filter. It is defined by dilating, sampling and periodizing a complex wavelet $\psi \in \mathbf{L^2}(\mathbb{C})$, with

$$\psi^j_m = \sum_{k=-\infty}^{\infty} \psi(2^j(m/p - k)) \quad \text{for } 1 \leq m \leq p.$$

Numerical computations are performed with a Cauchy wavelet whose Fourier transform is

$$\hat{\psi}(\omega) = \omega^d \, e^{-\omega} \, \mathbf{1}_{\omega > 0},$$

up to a scaling factor, with $d = 5$. To guarantee that A is an invertible operator, the lowest signal frequencies are carried by a suitable low-pass filter ϕ and

$$(Ax)_{k+Jp} = (x \star \phi)_k \quad \text{for } 1 \leq k \leq p.$$

One can prove that x is always uniquely determined by $|Ax|$, up to a multiplication factor. When x is real, the reconstruction appears to be numerically stable. Recall that the results of §3.6.4 allow us to explicitly impose the condition that x is real in the PhaseCut recovery algorithm. For PhaseLift in Candes et al.

[2011b], this condition is enforced by imposing that $X = xx^*$ is real. For the Gerchberg-Saxton algorithm, when x is real, we simply project at each iteration on the image of $\mathbb{R}^p$ by A, instead of projecting on the image of $\mathbb{C}^p$ by A.

Numerical experiments are performed on the real part of the complex test signals. Table 1 shows that Gerchberg-Saxton does not reconstruct exactly any real test signal from the modulus of its wavelet coefficients. The average relative error $\epsilon(\tilde{x}, x)$ in Table 2 is 1.2 where the coefficient amplitudes have an average error $\epsilon(|A\tilde{x}|, |Ax|)$ of 0.3 in Table 3.

PhaseLift reconstructs 62% of test signals, but the reconstruction rate varies with signal type. The proportions of exactly reconstructed signals among random noises, sums of sinusoids and image scan-lines are 27%, 60% and 99% respectively. Indeed, image scan-lines have a large proportion of wavelet coefficients whose amplitudes are negligible. The proportion of phase coefficients having a strong impact on the reconstruction of x is thus much smaller for scan-line images than for random noises, which reduces the number of significant variables to recover. Sums of sinuoids of random frequency have wavelet coefficients whose sparsity is intermediate between image scan-lines and Gaussian white noises, which explains the intermediate performance of PhaseLift on these signals. The overall average error $\epsilon(\tilde{x}, x)$ on non-reconstructed signals is 0.5. Despite this relatively important error, $\tilde{x}$ and x are usually almost equal on most of their support, up to a sign switch, and the importance of the error is precisely due to the local phase inversions (which change signs).

The PhaseCut algorithm reconstructs exactly all test signals. Moreover, the recovered matrix U is always of rank one and it is therefore not necessary to refine the solution with Gerchberg-Saxton iterations. At first sight, this difference in performance between PhaseCut and PhaseLift may seem to contradict the equivalence results of §4.3 (which are valid when x is real and when x is complex). It can be explained however by the fact that 10 steps of reweighing and 1000 inner iterations per step are not enough to let PhaseLift fully converge. In these experiments, the precision required to get perfect reconstruction is very high and, consequently, the number of first-order iterations required to achieve it is too large (see §4.6). With an interior-point-solver, this number would be much smaller but the time required per iteration would become prohibitively large. The much simpler structure of the PhaseCut relaxation allows us to solve these larger problems more efficiently.

5.4. **Impact of Trace Minimization.** We saw in §4.1 that, in the absence of noise, PhaseCut was very similar to a simplified version of PhaseLift, Weak PhaseLift, in which no trace minimization is performed. Here, we confirm empirically that Weak PhaseLift and PhaseLift are essentially equivalent. Minimizing the trace is usually used as rank minimization heuristic, with recovery guarantees in certain settings [Fazel et al., 2003; Candes and Recht, 2008; Chandrasekaran et al., 2012] but it does not seem to make much difference here. In fact, Demanet and Hand [2012] recently showed that in the independent experiments setting, Weak PhaseLift has a unique (rank one) solution with high probability, i.e. the feasible set of PhaseLift is a singleton and trace minimization has no impact. Of course, from a numerical point of view, solving the feasibility problem Weak PhaseLift is about as hard as solving the trace minimization problem PhaseLift, so the result [Demanet and Hand, 2012] simplifies analysis but does not really affect numerical performance.

Figure 3 compares the performances of PhaseLift and Weak PhaseLift as a function of n (the number of measurements). We plot the percentage of successful reconstructions (*left*) and the percentage of cases where the relaxation was exact, i.e. the reconstructed matrix X was rank one (*right*). The plot shows a clear phase transitions when the number of measurements increases. For PhaseLift, these transitions happen respectively at $n = 155 \approx 2.5p$ and $n = 285 \approx 4.5p$, while for Weak PhaseLift, the values become $n = 170 \approx 2.7p$ and $n = 295 \approx 4.6p$, so the transition thresholds are very similar. Note that, in the absence of noise, Weak PhaseLift and PhaseCut have the same solutions, up to a linear transformation (see §4.2), so we can expect the same behavior in the comparison PhaseCut versus PhaseCutMod.

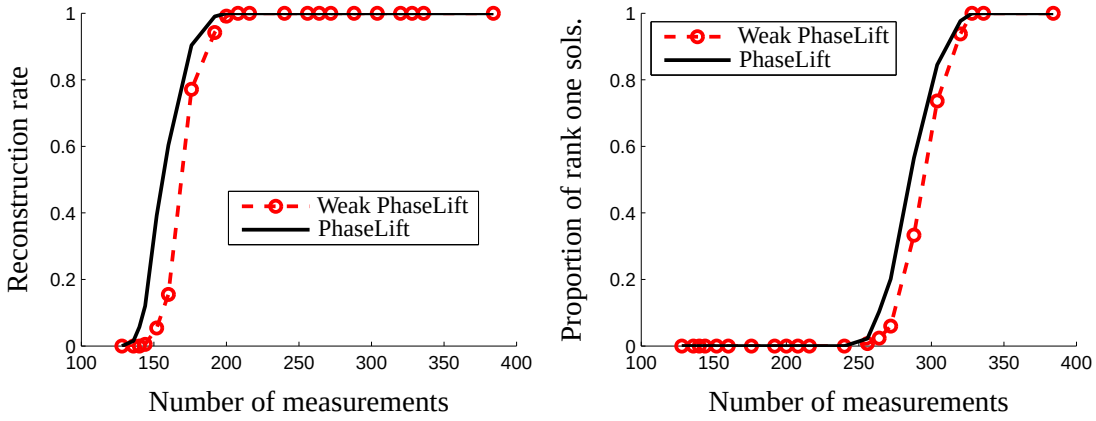

FIGURE 3. Comparison of PhaseLift and Weak PhaseLift performance, for 64-sized signals, as a function of the number of measurements. Reconstruction rate, after Gerchberg-Saxton iterations *(left)* and proportion of rank one solutions *(right)*.

5.5. **Reconstruction in the Presence of Noise.** Numerical stability is crucial for practical applications. In this last subsection, we suppose that the vector b of measurements is of the form

$$b = |Ax| + b_{\text{noise}}$$

with $\|b_{\text{noise}}\|_2 = o(\|Ax\|_2)$. In our experiments, b_{noise} is always a Gaussian white noise. Two reasons can explain numerical instabilities in the solution $\tilde{x}$. First, the reconstruction problem itself can be unstable, with $\|\tilde{x} - cx\| \gg \||A\tilde{x}| - |Ax|\|$ for all $c \in \mathbb{C}$. Second, the algorithm may fail to reconstruct $\tilde{x}$ such that $\||A\tilde{x}| - b\| \approx \|b_{\text{noise}}\|$. No algorithm can overcome the first cause but good reconstruction methods will overcome the second one. In the following paragraphs, to complement the results in §4.4, we will demonstrate empirically that PhaseCut is stable, and compare its performances with PhaseLift. We will observe in particular that PhaseCut appears to be more stable than PhaseLift when b is sparse.

5.5.1. *Wavelet transform.* Figure 4 displays the performance of PhaseCut in the wavelet transform case. It shows that PhaseCut is stable up to around $5 - 10\%$ of noise. Indeed, the reconstructed $\tilde{x}$ usually satisfies $\epsilon(|Ax|, |A\tilde{x}|) = \| \, |Ax| - |A\tilde{x}| \, \|_2 \leq \|b_{\text{noise}}\|_2$, which is the best we can hope for. Wavelet transform is a case where the underlying phase retrieval problem may present instabilities, therefore the reconstruction error $\epsilon(x, \tilde{x})$ is sometimes much larger than $\epsilon(|Ax|, |A\tilde{x}|)$. This remark applies especially to sums of sinusoids, which represent the most unstable case.

When all coefficients of Ax have approximately the same amplitude, PhaseLift and PhaseCut produce similar results, but when Ax is sparse, PhaseLift appears less stable. We gave a qualitative explanation of this behavior at the end of §4.4 which seems to be confirmed by the results in Figure 4. This boils down to the fact that the values of the phase variables in PhaseCut corresponding to zeros in b can be set to zero so the problem becomes much smaller. Indeed, the performance of PhaseLift and PhaseCut are equivalent in the case of Gaussian random filters (where measurements are never sparse), they are a bit worse in the case of sinusoids (where measurements are sometimes sparse) and quite unsatisfactory for scan-lines of images (where measurements are always sparse).

5.5.2. *Multiple random illumination filters.* Candes and Li [2012] proved that, if A was a Gaussian matrix, the reconstruction problem was stable with high probability, and PhaseLift reconstructed a $\tilde{x}$ such that

$$\epsilon(\tilde{x}, x) \leq O\left(\frac{\|b_{\text{noise}}\|_2}{\|Ax\|_2}\right).$$

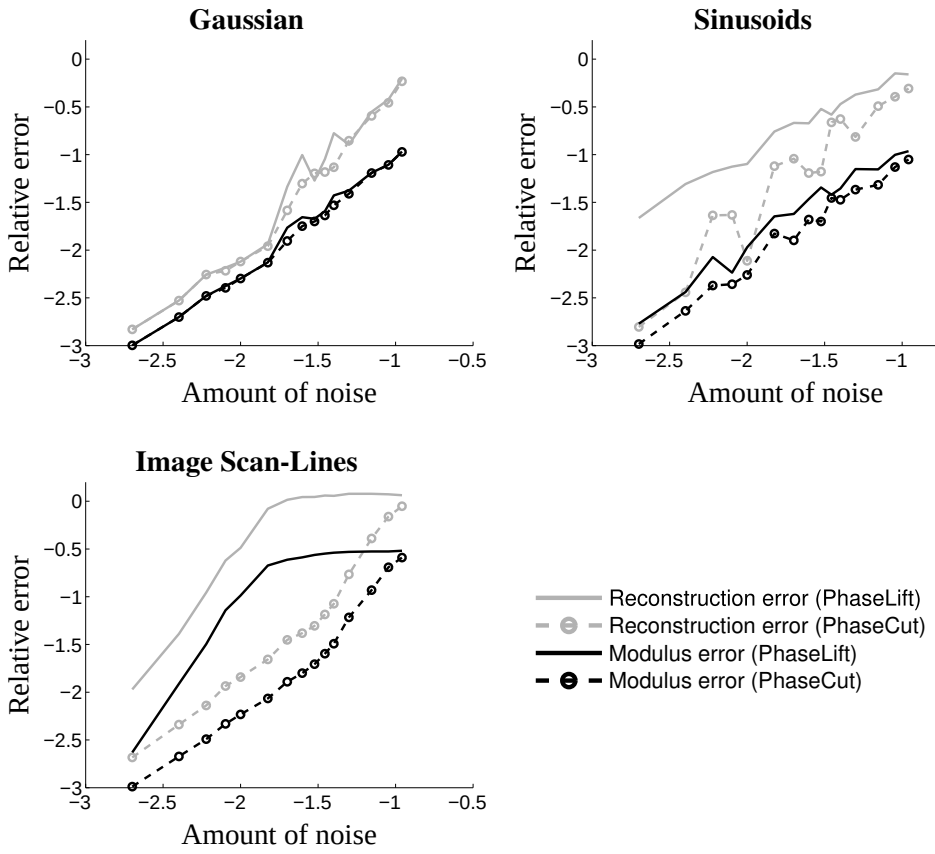

FIGURE 4. Mean reconstruction errors versus amount of noise for PhaseLift and PhaseCut, both in decimal logarithmic scale, for three types of signals: Gaussian white noises, sums of sinusoids and scan-lines of images. Both algorithms were followed by a few hundred Gerchberg-Saxton iterations.

The same result seems to hold for A corresponding to Gaussian random illumination filters (cf. §5.2). Moreover, PhaseCut is as stable as PhaseLift. Actually, up to 20% of noise, when followed by some Gerchberg-Saxton iterations, PhaseCut and PhaseLift almost always reconstruct the same function. Figure 5 displays the corresponding empirical performance, confirming that both algorithms are stable. The relative reconstruction errors are approximately linear in the amount of noise, with

$$\epsilon(|A\tilde{x}|, |Ax|) \approx 0.8 \times \frac{\|b_{\text{noise}}\|_2}{\|Ax\|_2} \quad \text{and} \quad \epsilon(\tilde{x}, x) \approx 2 \times \frac{\|b_{\text{noise}}\|_2}{\|Ax\|_2}$$

in our experiments.

The impact of the sparsity of b discussed in the last paragraph may seem irrelevant here: if A and x are independently chosen, Ax is never sparse. However, if we do not choose A and x independently, we may achieve partial sparsity. We performed tests for the case of five Gaussian random filters, where we chose $x \in \mathbb{C}^{64}$ such that $(Ax)_k = 0$ for $k \leq 60$. This choice has no particular physical interpretation but it allows us to check that the influence of sparsity in $|Ax|$ over PhaseLift is not specific to the wavelet transform. Figure 5 displays the relative error over the reconstructed matrix in the sparse and non-sparse cases. If we denote by $X_{\text{pl}} \in \mathbb{C}^{p \times p}$ (resp. $X_{\text{pc}} \in \mathbb{C}^{n \times n}$) the matrix reconstructed by PhaseLift (resp. PhaseCut), this

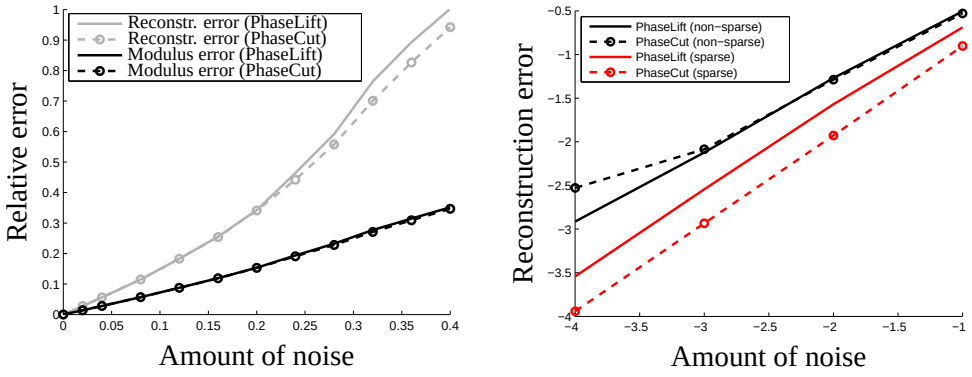

FIGURE 5. *Left:* Mean performances of PhaseLift and PhaseCut, followed by Gerchberg-Saxton iterations, for four Gaussian random illumination filters. The x-axis represents the relative noise level, $\|b_{\text{noise}}\|_2/\|Ax\|_2$ and the y-axis the relative error on the result, which is either $\epsilon(\tilde{x}, x)$ or $\epsilon(|A\tilde{x}|, |Ax|)$. *Right:* Loglog plot of the relative error over the matrix reconstructed by PhaseLift (resp. PhaseCut) when A represents the convolution by five Gaussian filters. Black curves correspond to Ax non-sparse, red ones to sparse Ax.

relative error is defined by

$$\epsilon = \frac{\|AX_{\text{pl}}A^* - (Ax)(Ax)^*\|_2}{\|(Ax)(Ax)^*\|_2} \qquad \text{(for PhaseLift)}$$

$$\epsilon = \frac{\|\,\mathbf{diag}(b)X_{\text{pc}}\,\mathbf{diag}(b) - (Ax)(Ax)^*\|_2}{\|(Ax)(Ax)^*\|_2} \qquad \text{(for PhaseCut)}$$

In the non-sparse case, both algorithms yield very similar error $\epsilon \approx 7\|b_{\text{noise}}\|_2/\|Ax\|_2$ (the difference for a relative noise of 10^{-4} may come from a computational artifact). In the sparse case, there are less phases to reconstruct, because we do not need to reconstruct the phase of null measurements. Consequently, the problem is better constrained and we expect the algorithms to be more stable. Indeed, the relative errors over the reconstructed matrices are smaller. However, in this case, the performance of PhaseLift and PhaseCut do not match anymore: $\epsilon \approx 3\|b_{\text{noise}}\|_2/\|Ax\|_2$ for PhaseLift and $\epsilon \approx 1.2\|b_{\text{noise}}\|_2/\|Ax\|_2$ for PhaseCut. This remark has no practical impact in our particular example here because taking a few Gerchberg-Saxton iterations would likely make both methods converge towards the same solution, but it confirms the importance of accounting for the sparsity of $|Ax|$.

ACKNOWLEDGMENTS

The authors are grateful to Richard Baraniuk, Emmanuel Candès, Rodolphe Jenatton, Amit Singer and Vlad Voroninski for very constructive comments. In particular, Vlad Voroninski showed in [Voroninski, 2012] that the argument in the first version of this paper, proving that PhaseCutMod is tight when PhaseLift is, could be reversed under mild technical conditions and pointed out an error in our handling of sparsity constraints. AA would like to acknowledge support from a starting grant from the European Research Council (project SIPA), and SM acknowledges support from ANR grant BLAN 012601.

APPENDIX A. TECHNICAL LEMMAS

We now prove two technical lemmas used in the proof of Theorem 4.7.

Lemma A.1. *Under the assumptions and notations of Theorem 4.7, we have*

$$\|V_{PC}^{/\!/} - (Ax_0)(Ax_0)^*\|_2 > 2C\|Ax_0\|_2\|b_{\text{n,PC}}\|_2$$

Proof. We first give an upper bound of $\|V_{PC} - V_{PC}^{/\!/}\|_2$. We use the Cauchy-Schwarz inequality : for every positive matrix X and all x, y, $|x^*Xy| \leq \sqrt{x^*Xx}\sqrt{y^*Xy}$. Let $\{f_i\}$ be an hermitian base of range(A) diagonalizing $V_{PC}^{/\!/}$ and $\{g_i\}$ an hermitian base of range$(A)^\perp$ diagonalizing $V_{PC}^\perp$. As $\{f_i\} \cap \{g_i\}$ is an hermitian base of $\mathbb{C}^n$, we have

$$\begin{aligned}
\|V_{PC} - V_{PC}^{/\!/}\|_2^2 &= \sum_{i,i'} |f_i^*(V_{PC} - V_{PC}^{/\!/})f_{i'}|^2 + \sum_{i,j} |f_i^*(V_{PC} - V_{PC}^{/\!/})g_j|^2 \\
&\quad + \sum_{i,j} |g_j^*(V_{PC} - V_{PC}^{/\!/})f_i|^2 + \sum_{j,j'} |g_j^*(V_{PC} - V_{PC}^{/\!/})g_{j'}|^2 \\
&= 2\sum_{i,j} |f_i^*(V_{PC})g_j|^2 + \sum_i |g_i^*(V_{PC}^\perp)g_i|^2 \\
&\leq 2\sum_{i,j} |f_i^*(V_{PC})f_i\|g_j^*(V_{PC})g_j| + (\sum_i g_i^*(V_{PC}^\perp)g_i)^2 \\
&= 2\,\mathbf{Tr}\,V_{PC}^{/\!/}\,\mathbf{Tr}\,V_{PC}^\perp + (\mathbf{Tr}\,V_{PC}^\perp)^2 \\
&\leq \left(\sqrt{2}\sqrt{\mathbf{Tr}\,V_{PC}^{/\!/}}\sqrt{\mathbf{Tr}\,V_{PC}^\perp} + \mathbf{Tr}\,V_{PC}^\perp\right)^2 \qquad (21)
\end{aligned}$$

Let us now bound $\mathbf{Tr}\,V_{PC}^\perp$. We first note that $\mathbf{Tr}\,V_{PC}^\perp = \mathbf{Tr}((\mathbf{I} - AA^\dagger)V_{PC}(\mathbf{I} - AA^\dagger)) = \mathbf{Tr}(V_{PC}(\mathbf{I} - AA^\dagger)) = d_1(V_{PC}, \mathcal{F})$ (according to lemma 4.1). Let $u \in \mathbb{C}^n$ be such that, for all i, $|u_i| = 1$ and $(Ax_0)_i = u_i|Ax_0|_i$. We set $b = |Ax_0| + b_{\mathrm{n,PC}}$ and $V = (b \times u)(b \times u)^*$. As $V \in \mathbf{H}_n^+ \cap \mathcal{H}_b$ and V_{PC} minimizes (13),

$$\begin{aligned}
\mathbf{Tr}\,V_{PC}^\perp &= d_1(V_{PC}, \mathcal{F}) \leq d_1(V, \mathcal{F}) = d_1((Ax_0 + b_{\mathrm{n,PC}}u)(Ax_0 + b_{\mathrm{n,PC}}u)^*, \mathcal{F}) \\
&= d_1((b_{\mathrm{n,PC}}u)(b_{\mathrm{n,PC}}u)^*, \mathcal{F}) \\
&\leq \|(b_{\mathrm{n,PC}}u)(b_{\mathrm{n,PC}}u)^*\|_1 = \mathbf{Tr}(b_{\mathrm{n,PC}}u)(b_{\mathrm{n,PC}}u)^* = \|b_{\mathrm{n,PC}}\|_2^2
\end{aligned}$$

We also have $\mathbf{Tr}\,V_{PC}^{/\!/} = \mathbf{Tr}\,V_{PC} - \mathbf{Tr}\,V_{PC}^\perp$. This equality comes from the fact that, if $\{f_i\}$ is an hermitian base of range(A) and $\{g_i\}$ an hermitian base of range$(A)^\perp$, then

$$\mathbf{Tr}\,V_{PC} = \sum_i f_i V_{PC} f_i^* + \sum_i g_i V_{PC} g_i^* = \sum_i f_i V_{PC}^{/\!/} f_i^* + \sum_i g_i V_{PC}^\perp g_i^* = \mathbf{Tr}\,V_{PC}^{/\!/} + \mathbf{Tr}\,V_{PC}^\perp$$

As $V_{PC}^\perp \succeq 0$, $\mathbf{Tr}\,V_{PC}^{/\!/} \leq \mathbf{Tr}\,V_{PC} = \||Ax_0| + b_{\mathrm{n,PC}}\|_2^2$ and, by combining this with relations (21) and (22), we get

$$\begin{aligned}
\|V_{PC} - V_{PC}^{/\!/}\|_2 &\leq \sqrt{2}\||Ax_0| + b_{\mathrm{n,PC}}\|_2\|b_{\mathrm{n,PC}}\|_2 + \|b_{\mathrm{n,PC}}\|_2^2 \\
&\leq \sqrt{2}\|Ax_0\|_2\|b_{\mathrm{n,PC}}\|_2 + (1 + \sqrt{2})\|b_{\mathrm{n,PC}}\|_2^2
\end{aligned}$$

And, by reminding that we assumed $\|b_{\mathrm{n,PC}}\|_2 \leq \|Ax_0\|_2$,

$$\begin{aligned}
\|V_{PC}^{/\!/} - (Ax_0)(Ax_0)^*\|_2 &\geq \|V_{PC} - (Ax_0)(Ax_0)^*\|_2 - \|V_{PC}^{/\!/} - V_{PC}\|_2 \\
&> D\|Ax_0\|_2\|b_{\mathrm{n,PC}}\|_2 - \sqrt{2}\|Ax_0\|_2\|b_{\mathrm{n,PC}}\|_2 - (1 + \sqrt{2})\|b_{\mathrm{n,PC}}\|_2^2 \\
&\geq (D - 2\sqrt{2} - 1)\|Ax_0\|_2\|b_{\mathrm{n,PC}}\|_2 = 2C\|Ax_0\|_2\|b_{\mathrm{n,PC}}\|_2
\end{aligned}$$

which concludes the proof. ■

Lemma A.2. *Under the assumptions and notations of Theorem 4.7, we have* $\|b_{\mathrm{n,PL}}\|_2 \leq 2\|b_{\mathrm{n,PC}}\|$.

Proof. Let e_i be the i-th vector of $\mathbb{C}^n$'s canonical base. We set $e_i = f_i + g_i$ where $f_i \in \text{range}(A)$ and $g_i \in \text{range}(A)^\perp$.

$$\begin{aligned} V_{PC\,ii} &= e_i^* V_{PC} e_i \\ &= f_i^* V_{PC}^{/\!/} f_i + 2\,\mathrm{Re}(f_i^* V_{PC} g_i) + g_i^* V_{PC}^{\perp} g_i \\ &= V_{PC\,ii}^{/\!/} + 2\,\mathrm{Re}(f_i^* V_{PC} g_i) + V_{PC\,ii}^{\perp} \end{aligned}$$

Because $|f_i^* V_{PC} g_i| \leq \sqrt{f_i^* V_{PC} f_i}\sqrt{g_i^* V_{PC} g_i} = \sqrt{V_{PC\,ii}^{/\!/}}\sqrt{V_{PC\,ii}^{\perp}}$,

$$\begin{aligned} &(\sqrt{V_{PC\,ii}^{/\!/}} - \sqrt{V_{PC\,ii}^{\perp}})^2 \leq V_{PC\,ii} \leq (\sqrt{V_{PC\,ii}^{/\!/}} + \sqrt{V_{PC\,ii}^{\perp}})^2 \\ &\Rightarrow \sqrt{V_{PC\,ii}^{/\!/}} - \sqrt{V_{PC\,ii}^{\perp}} \leq \sqrt{V_{PC\,ii}} \leq \sqrt{V_{PC\,ii}^{/\!/}} + \sqrt{V_{PC\,ii}^{\perp}} \end{aligned}$$

So

$$\begin{aligned} |b_{\mathrm{n,PL},i}| &= |\sqrt{V_{PC\,ii}^{/\!/}} - |Ax_0|_i| \\ &\leq |\sqrt{V_{PC\,ii}^{/\!/}} - \sqrt{V_{PC\,ii}}| + |\sqrt{V_{PC\,ii}} - |Ax_0|_i| \\ &\leq \sqrt{V_{PC\,ii}^{\perp}} + b_{\mathrm{n,PC},i} \end{aligned}$$

and, by (22),

$$\begin{aligned} \|b_{\mathrm{n,PL}}\|_2 &\leq \| \left\{ \sqrt{V_{PC\,ii}^{\perp}} \right\}_i \|_2 + \|b_{\mathrm{n,PC}}\|_2 \\ &= \sqrt{\mathbf{Tr}\, V_{PC}^{\perp}} + \|b_{\mathrm{n,PC}}\|_2 \leq 2\|b_{\mathrm{n,PC}}\|_2 \end{aligned}$$

which concludes the proof. ■

REFERENCES

E. J. Akutowicz. On the determination of the phase of a Fourier integral, I. *Trans. Am. Math. Soc.*, 83:179–192, 1956.

S. Becker, E.J. Candes, and M. Grant. Tfocs v1. 1 user guide. 2012.

A. Ben-Tal and A. Nemirovski. *Lectures on modern convex optimization : analysis, algorithms, and engineering applications*. MPS-SIAM series on optimization. Society for Industrial and Applied Mathematics : Mathematical Programming Society, Philadelphia, PA, 2001.

A. Ben-Tal, A. Nemirovski, and C. Roos. Extended matrix cube theorems with applications to μ-theory in control. *Mathematics of Operations Research*, 28(3):497–523, 2003.

A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1998.

R. Bhatia. *Matrix analysis*, volume 169. Springer Verlag, 1997.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

O. Bunk, A. Diaz, F. Pfeiffer, C. David, B. Schmitt, D.K. Satapathy, and JF Veen. Diffractive imaging for periodic samples: retrieving one-dimensional concentration profiles across microfluidic channels. *Acta Crystallographica Section A: Foundations of Crystallography*, 63(4):306–314, 2007.

E. J. Candes, T. Strohmer, and V. Voroninski. Phaselift : exact and stable signal recovery from magnitude measurements via convex programming. *To appear in Communications in Pure and Applied Mathematics*, 2011a.

E.J. Candes and X. Li. Solving quadratic equations via phaselift when there are about as many equations as unknowns. *Arxiv preprint arXiv:1208.6247*, 2012.

E.J. Candes and B. Recht. Exact matrix completion via convex optimization. *preprint*, 2008.

E.J. Candes, Y. Eldar, T. Strohmer, and V. Voroninski. Phase retrieval via matrix completion. *Arxiv preprint arXiv:1109.0573*, 2011b.

A. Chai, M. Moscoso, and G. Papanicolaou. Array imaging using intensity-only measurements. *Inverse Problems*, 27:015005, 2011.

V. Chandrasekaran, B. Recht, P. Parrilo, and A.S. Willsky. The convex geometry of linear inverse problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012.

T. Chi, P. Ru, and S. Shamma. Multiresolution spectrotemporal analysis of complex sounds. *J. of Acoustic. Societ. of America*, 118:887–906, 2005.

A. d'Aspremont, O. Banerjee, and L. El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2006.

C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62(1): 557–574, 1993.

L. Demanet and P. Hand. Stable optimizationless recovery from phaseless linear measurements. *Arxiv preprint arXiv:1208.1803*, 2012.

N. El Karoui and A. d'Aspremont. Approximating eigenvectors by subsampling. *ArXiv:0908.0137*, 2009.

M. Fazel, H. Hindi, and S.P. Boyd. Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices. In *American Control Conference, 2003. Proceedings of the 2003*, volume 3, pages 2156–2162. Ieee, 2003.

J.R. Fienup. Phase retrieval algorithms: a comparison. *Applied Optics*, 21(15):2758–2769, 1982.

R. Gerchberg and W. Saxton. A practical algorithm for the determination of phase from image and diffraction plane pictures. *Optik*, 35:237–246, 1972.

Michel X Goemans and David P Williamson. Approximation algorithms for max-3-cut and other problems via complex semidefinite programming. *Journal of Computer and System Sciences*, 68(2):442–470, 2004.

M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.

D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 32(2):236–243, 1984.

R.W. Harrison. Phase problem in crystallography. *JOSA A*, 10(5):1046–1055, 1993.

C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior–point method for semidefinite programming. *SIAM Journal on Optimization*, 6:342–361, 1996.

T. Kato. *Perturbation theory for linear operators*. Springer, 1995.

M. Kisialiou and Z.Q. Luo. Probabilistic analysis of semidefinite relaxation for binary quadratic minimization. *SIAM Journal on Optimization*, 20:1906, 2010.

X. Li and V. Voroninski. Sparse signal recovery from quadratic measurements via convex programming. *arXiv preprint arXiv:1209.4785*, 2012.

L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

Z.Q. Luo, X. Luo, and M. Kisialiou. An efficient quasi-maximum likelihood decoder for psk signals. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 6, pages VI–561. IEEE, 2003.

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.*, 59:387–410, 2008.

M.L. Moravec, J.K. Romberg, and R.G. Baraniuk. Compressive phase retrieval. In *Proc. of SPIE Vol*, volume 6701, pages 670120–1, 2007.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Y. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization methods and software*, 9(1): 141–160, 1998.

Y. Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110 (2):245–259, 2007.

E. Osherovich, Y. Shechtman, A. Szameit, P. Sidorenko, E. Bullkich, S. Gazit, S. Shoham, E.B. Kley, M. Zibulevsky, I. Yavneh, et al. Sparsity-based single-shot subwavelength coherent diffractive imaging. In *CLEO: Science and Innovations*. Optical Society of America, 2012.

J. L. C. Sanz. Mathematical considerations for the problem of fourier transform phase retrieval from magnitude. *SIAM Journal on Applied Mathematics*, 45:651–664, 1985.

N.Z. Shor. Quadratic optimization problems. *Soviet Journal of Computer and Systems Sciences*, 25:1–11, 1987.

A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36, 2011.

A.M.C. So. Non–asymptotic performance analysis of the semidefinite relaxation detector in digital communications. 2010.

A.M.C. So, J. Zhang, and Y. Ye. On approximating complex quadratic optimization problems via semidefinite programming relaxations. *Mathematical Programming*, 110(1):93–110, 2007.

Anthony Man-Cho So and Yinyu Ye. Probabilistic analysis of semidefinite relaxation detectors for multiple-input, multiple-output. *Convex Optimization in Signal Processing and Communications*, page 166, 2010.

G.W. Stewart. *Matrix Algorithms Vol. II: Eigensystems*. Society for Industrial Mathematics, 2001.

G.W. Stewart and J. Sun. Matrix perturbation theory. 1990.

M. Todd and E. A. Yildirim. Sensitivity analysis in linear programming and semidefinite programming using interior-points methods. *Mathematical Programming*, 90(2):229–261, 2001.

K. C. Toh, M. J. Todd, and R. H. Tutuncu. SDPT3 – a MATLAB software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

V. Voroninski. A comparison between the phaselift and phasecut algorithms. *Working paper*, 2012.

I. Waldspurger and S. Mallat. Time-frequency phase recovery. *Working paper*, 2012.

Z. Wen, D. Goldfarb, and K. Scheinberg. Block coordinate descent methods for semidefinite programming. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 533–564. Springer, 2012.

EA Yildirim. An interior-point perspective on sensitivity analysis in semidefinite programming. *Mathematics of Operations Research*, 28(4):649–676, 2003.

S. Zhang and Y. Huang. Complex quadratic optimization and semidefinite programming. *SIAM Journal on Optimization*, 16(3):871–890, 2006.

D.I., ÉCOLE NORMALE SUPÉRIEURE, PARIS.
E-mail address: waldspur@clipper.ens.fr

CNRS & D.I., ÉCOLE NORMALE SUPÉRIEURE, PARIS. UMR 8548.
E-mail address: alexandre.daspremont@m4x.org

D.I., ÉCOLE NORMALE SUPÉRIEURE, PARIS.
E-mail address: mallat@cmap.polytechnique.fr