Sketchy Decisions: Convex Low-Rank Matrix Optimization with Optimal Storage

Alp Yurtsever EPFL
Madeleine Udell Cornell
Joel A. Tropp Caltech
Volkan Cevher EPFL

Abstract

This paper concerns a fundamental class of convex matrix optimization problems. It presents the first algorithm that uses optimal storage and provably computes a low-rank approximation of a solution. In particular, when all solutions have low rank, the algorithm converges to a solution. This algorithm, SketchyCGM, modifies a standard convex optimization scheme, the conditional gradient method, to store only a small randomized sketch of the matrix variable. After the optimization terminates, the algorithm extracts a low-rank approximation of the solution from the sketch. In contrast to nonconvex heuristics, the guarantees for SketchyCGM do not rely on statistical models for the problem data. Numerical work demonstrates the benefits of SketchyCGM over heuristics.

1 MOTIVATION

This paper discusses a fundamental class of convex matrix optimization problems with low-rank solutions. We argue that the main obstacle that prevents us from solving these problems at scale is not arithmetic, but storage. We exhibit the first provably correct algorithm for these problems with optimal storage.

1.1 Vignette: Matrix Completion

To explain the challenge, we consider the problem of low-rank matrix completion.

Let $\boldsymbol{X}_\natural \in \mathbb{R}^{m\times n}$ be an unknown matrix, but assume that a bound r on the rank of $\boldsymbol{X}_\natural$ is available, where $r \ll \min\{m, n\}$. Suppose that we record noisy observations of a subset E of entries from the matrix:

$$b_{ij} = (\boldsymbol{X}_\natural)_{ij} + \xi_{ij} \quad \text{for } (i,j) \in E.$$

The variables $\xi_{ij} \in \mathbb{R}$ model (unknown) noise. The goal is to approximate the full matrix $\boldsymbol{X}_\natural$.

Matrix completion arises in machine learning applications, such as recommendation systems [32].

We can frame the matrix completion problem as a rank-constrained optimization:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \sum_{(i,j)\in E} (x_{ij} - b_{ij})^2 \quad \text{s.t.} \quad \operatorname{rank} \boldsymbol{X} \le r. \quad (1)$$

In general, the formulation (1) is intractable. Instead, we retrench to a tractable convex problem [7, 32]:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \sum_{(i,j)\in E} (x_{ij} - b_{ij})^2 \quad \text{s.t.} \quad \|\boldsymbol{X}\|_{S_1} \le \alpha. \quad (2)$$

The Schatten 1-norm $\|\cdot\|_{S_1}$ returns the sum of the singular values of its argument; it is an effective proxy for the rank [13]. Adjusting the value of the parameter α modulates the rank of a solution $\boldsymbol{X}_\star$ of (2). If we have enough data and choose α well, we expect that each solution $\boldsymbol{X}_\star$ approximates the target matrix $\boldsymbol{X}_\natural$.

The convex problem (2) is often a good model for matrix completion when the number of observations $|E| = \tilde{O}(r(m+n))$, where $\tilde{O}$ suppresses log-like factors; see [7, 32]. We can write a rank-r approximation to a solution $\boldsymbol{X}_\star$ using $\Theta(r(m+n))$ parameters. Thus, we can express the problem and an approximate solution with $\tilde{O}(r(m+n))$ storage.

Nevertheless, we need fully mn numbers to express the decision variable $\boldsymbol{X}$ for the optimization problem (2). The cost of storing the decision variable prevents us from solving large-scale instances of (2), even without worrying about arithmetic.

This discrepancy raises a question: **Is there an algorithm that computes an approximate solution to** (2) **using the optimal storage** $\tilde{O}(r(m+n))$**?**

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

1.2 Vignette: Phase Retrieval

Here is another instance of the same predicament.

Fix a vector $\boldsymbol{x}_\natural \in \mathbb{C}^n$. Suppose that we acquire d noisy quadratic measurements of $\boldsymbol{x}_\natural$ with the form

$$b_i = |\langle \boldsymbol{a}_i, \boldsymbol{x}_\natural \rangle|^2 + \xi_i \quad \text{for } i = 1, 2, \dots, d. \tag{3}$$

The $\boldsymbol{a}_i \in \mathbb{C}^n$ are known measurement vectors, and the $\xi_i \in \mathbb{R}$ model measurement noise. Given the data $\boldsymbol{b}$ and the vectors $\boldsymbol{a}_i$, the phase retrieval problem asks us to reconstruct $\boldsymbol{x}_\natural$ up to a global phase shift.

Phase retrieval problems are prevalent in imaging science because it is easier to measure the intensity of light than its phase. In practice, the vectors $\boldsymbol{a}_i$ are structured because they reflect the physics of the imaging system. See the supplement and [3, 8, 10, 21].

Let us outline a convex approach [3, 8, 10, 21] to the phase retrieval problem. The data (3) satisfies

$$b_i = \boldsymbol{a}_i^* \boldsymbol{X}_\natural \boldsymbol{a}_i + \xi_i \quad \text{where} \quad \boldsymbol{X}_\natural = \boldsymbol{x}_\natural \boldsymbol{x}_\natural^*.$$

Thus, we can formulate phase retrieval as

$$\begin{array}{ll} \underset{\boldsymbol{X} \in \mathbb{C}^{n \times n}}{\text{minimize}} & \sum_{i=1}^d (\boldsymbol{a}_i^* \boldsymbol{X} \boldsymbol{a}_i - b_i)^2 \\ \text{s.t.} & \operatorname{rank} \boldsymbol{X} = 1, \quad \boldsymbol{X} \succcurlyeq \boldsymbol{0}. \end{array} \tag{4}$$

Now, pass to the convex problem

$$\begin{array}{ll} \underset{\boldsymbol{X} \in \mathbb{C}^{n \times n}}{\text{minimize}} & \sum_{i=1}^d (\boldsymbol{a}_i^* \boldsymbol{X} \boldsymbol{a}_i - b_i)^2 \\ \text{s.t.} & \operatorname{tr} \boldsymbol{X} \le \alpha, \quad \boldsymbol{X} \succcurlyeq \boldsymbol{0}. \end{array} \tag{5}$$

We can estimate the parameter $\alpha \in \mathbb{R}_+$ from $\boldsymbol{a}_i$ and $\boldsymbol{b}$; see [37, Sec. II]. To approximate the true vector $\boldsymbol{x}_\natural$, we compute a top eigenvector $\boldsymbol{x}_\star$ of a solution to (5).

This procedure is often an effective approach for phase retrieval when the number of measurements $d = \Theta(n)$; see [34, Sec. 2.8]. Once again, we recognize a discrepancy. The problem data $\boldsymbol{b} \in \mathbb{R}^d$ and the approximate solution $\boldsymbol{x}_\star \in \mathbb{C}^n$ use storage $\Theta(n)$, but the matrix variable in (5) requires $\Theta(n^2)$ storage.

We may ask: **Is there an algorithm that computes an approximate solution to** (5) **using the optimal storage** $\Theta(n)$**?**

1.3 Low-Rank Matrix Optimization Methods

Matrix completion and phase retrieval are examples of *convex low-rank matrix optimization* (CLRO) problems. Informally, this class contains convex optimization problems whose decision variable is a matrix and whose solutions are (close to) low rank. These problems often arise as convex relaxations of rank-constrained problems; however, the convex formulations are important in their own right.

There has been extensive empirical and theoretical work to validate the use of CLROs in a spectrum of applications. For example, see [7, 8, 13, 16, 21].

Over the last 20 years, optimization researchers have developed a diverse collection of algorithms for CLRO problems. Surprisingly, every extant method lacks guarantees on storage or convergence (or both).

Convex optimization algorithms dominated the early literature on algorithms for CLRO. The initial efforts, such as [20], focused on interior-point methods, whose storage and arithmetic costs are forbidding. To resolve this issue, researchers turned to first-order convex algorithms, including bundle methods [19], (accelerated) proximal gradient methods [2, 30, 33], and the conditional gradient method (CGM) [11, 15, 18, 22, 24].

Convex algorithms are guaranteed to solve a CLRO. They come with a complete theory, including rigorous stopping criteria and bounds on convergence rates. They enjoy robust performance in practice. On the other hand, convex algorithms from the literature do not scale well enough to solve large CLRO problems because they operate on and store full-size matrices.

The CGM iteration is sometimes touted as a low-storage method for CLRO [22]. Indeed, CGM is guaranteed to increase the rank of an iterate by at most one per iteration. Nevertheless, the algorithm converges slowly, so intermediate iterates can have very high rank. CGM variants, such as [29, 37], that control the rank of iterates lack storage guarantees or may not converge to a global optimum.

Recently, many investigators have sought recourse in nonconvex heuristics for solving CLROs. This line of work depends on the factorization idea of Burer & Monteiro [6], which rewrites the matrix variable as a product of two low-rank factors. There are many heuristic procedures, e.g., [4–6, 23], that use clever initialization and nonlinear programming schemes in an attempt to optimize the factors directly. The resulting algorithms can have optimal storage costs, and they may achieve a fast rate of local convergence.

There has been an intensive effort to justify the application of nonconvex heuristics for CLRO. To do so, researchers often frame unverifiable statistical assumptions on the problem data. For example, in the matrix completion problem (2), it is common to assume that the entries of the matrix are revealed according to some ideal probability distribution [7, 23]. When these assumptions fail, nonconvex heuristics can converge to the wrong point, or they may even diverge.

Contributions. This paper explains how to extend the convex optimization algorithm CGM to obtain an approximate solution to a class of CLRO problems using optimal storage. Our algorithm operates much like CGM, but it never forms the matrix variable explicitly. Instead, we maintain a small randomized sketch of the matrix variable over the course of the iteration by means of a bespoke sketching method [35]. After the optimization method converges, we extract an approximate solution from the sketch. This technique achieves optimal storage, yet it converges under the same conditions and with the same guarantees as CGM.

In summary, this paper presents a solution to the problems posed above: the first algorithm for convex low-rank matrix optimization problems that provably uses optimal storage to compute an approximate solution.

1.4 Notation

We write $\|\cdot\|$ for the Euclidean norm, $\|\cdot\|_{\mathrm{F}}$ for the Frobenius norm, and $\|\cdot\|_{S_1}$ for the Schatten 1-norm (aka the *trace norm* or the *nuclear norm*). Depending on context, $\langle\cdot,\cdot\rangle$ refers to the Euclidean or Frobenius inner product. The symbol * denotes the conjugate transpose of a vector or matrix, as well as the adjoint of a linear map. The dagger † refers to the pseudoinverse. The symbol $[\boldsymbol{M}]_r$ stands for a best rank-r Frobenius-norm approximation of the matrix $\boldsymbol{M}$. The function $\mathrm{dist}_{\mathrm{F}}(\boldsymbol{M}; S)$ returns the minimum Frobenius-norm distance from $\boldsymbol{M}$ to a set S. The symbol $\succcurlyeq$ denotes the semidefinite order. We use the computer science interpretation of the order notation $O, \tilde{O}, \Omega, \Theta$.

2 A LOW-RANK MATRIX OPTIMIZATION PROBLEM

Let us begin with a generalization of the convex matrix completion formulation (2). In §5.5, we return to the psd setting of the phase retrieval problem (5).

We consider a convex program with a matrix variable:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}}\ f(\mathcal{A}\boldsymbol{X}) \quad \text{s.t.} \quad \|\boldsymbol{X}\|_{S_1} \le \alpha. \tag{6}$$

The linear operator $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^d$ and its adjoint $\mathcal{A}^* : \mathbb{R}^d \to \mathbb{R}^{m\times n}$ take the form

$$\begin{aligned} \mathcal{A}\boldsymbol{X} &= \begin{bmatrix}\langle \boldsymbol{A}_1, \boldsymbol{X}\rangle & \dots & \langle \boldsymbol{A}_d, \boldsymbol{X}\rangle\end{bmatrix}; \\ \mathcal{A}^*\boldsymbol{z} &= \sum\nolimits_{i=1}^{d} z_i \boldsymbol{A}_i. \end{aligned} \tag{7}$$

Each coefficient matrix $\boldsymbol{A}_i \in \mathbb{R}^{m\times n}$.

We interpret $\mathcal{A}\boldsymbol{X}$ as a set of linear measurements of the matrix $\boldsymbol{X}$. For example, in the matrix completion problem (2), the image $\mathcal{A}\boldsymbol{X}$ lists the entries of $\boldsymbol{X}$ indexed by the set E.

The function $f : \mathbb{R}^d \to \mathbb{R}$ is convex and continuously differentiable. In many situations, it is natural to regard the objective function as a loss: $f(\mathcal{A}\boldsymbol{X}) = \mathrm{loss}(\mathcal{A}\boldsymbol{X}; \boldsymbol{b})$ for a vector $\boldsymbol{b} \in \mathbb{R}^d$ of measured data.

By choosing the parameter $\alpha \in \mathbb{R}_+$ to be sufficiently small, we can often ensure that each minimizer of (6) is low-rank or close to low-rank.

Our goal is to develop a practical algorithm that provably computes a low-rank approximation of a solution to the problem (6).

To validate (6) as a model for a given application, one must undertake a separate empirical or theoretical study. We do not engage this question in our work.

2.1 Storage Issues

Suppose that we want to produce a low-rank approximation to a solution of a generic instance of the problem (6). What kind of storage can we hope to achieve?

It is clear that we need $\Theta(r(m+n))$ numbers to express a rank-r approximate solution to (6). We must also understand how much extra storage is incurred because of the specific problem instance $(\mathcal{A}, f)$.

It is natural to instate a *black-box model* for the linear map $\mathcal{A}$, its adjoint $\mathcal{A}^*$, and the objective function f. For arbitrary vectors $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{z} \in \mathbb{R}^d$, assume we have routines that can compute

$$\mathcal{A}(\boldsymbol{u}\boldsymbol{v}^*) \quad \text{and} \quad \boldsymbol{u}^*(\mathcal{A}^*\boldsymbol{z}) \quad \text{and} \quad (\mathcal{A}^*\boldsymbol{z})\boldsymbol{v}. \tag{8}$$

We also assume routines for evaluating the function f and its gradient ∇f for any argument $\boldsymbol{z} \in \mathbb{R}^d$. We may neglect the storage used to compute these primitives. Every algorithm based on these primitives must use storage $\Omega(m+n+d)$ just to represent their outputs.

Thus, under the black-box model, any algorithm that produces a rank-r solution to a generic instance of (6) must use storage $\Omega(d + r(m+n))$. We say that an algorithm is *storage optimal* if it achieves this bound.

The parameter d often reflects the amount of data that we have acquired, and it is usually far smaller than the dimension mn of the matrix variable in (6).

The problems that concern us are data-limited; that is, $d \ll mn$. This is the situation where a strong structural prior (e.g., low rank or small Schatten 1-norm) is essential for fitting the data. This challenge is common in machine learning problems (e.g., matrix completion for recommendation systems), as well as in scientific applications (e.g., phase retrieval).

To the best of our knowledge, no extant algorithm for (6) is guaranteed to produce an approximation of an optimal point and also enjoys optimal storage cost.

3 CONDITIONAL GRADIENT

To develop our algorithm for the model problem (6), we must first describe a standard algorithm called the *conditional gradient method* (CGM). Classic and contemporary references include [11, 15, 18, 22, 24].

3.1 The CGM Iteration

Here is the CGM algorithm for (6). Start with a feasible point, such as

$$\boldsymbol{X}_0 = \mathbf{0} \in \mathbb{R}^{m \times n}. \tag{9}$$

At each iteration $t = 0, 1, 2, \dots$, compute an update direction $\boldsymbol{H}_t$ using the formulas

$$\begin{aligned} (\boldsymbol{u}_t, \boldsymbol{v}_t) &= \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\mathcal{A}\boldsymbol{X}_t))); \\ \boldsymbol{H}_t &= -\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*. \end{aligned} \tag{10}$$

`MaxSingVec` returns a left/right pair of maximum singular vectors. Update the decision variable:

$$\boldsymbol{X}_{t+1} = (1 - \eta_t)\boldsymbol{X}_t + \eta_t \boldsymbol{H}_t \tag{11}$$

where $\eta_t = 2/(t+2)$. The convex combination (11) remains feasible for (6) because $\boldsymbol{X}_t$ and $\boldsymbol{H}_t$ are feasible.

CGM is a valuable algorithm for (6) because we can efficiently find the rank-one update direction $\boldsymbol{H}_t$ by means of the singular vector computation (10). The weak point of CGM is that the rank of $\boldsymbol{X}_t$ typically increases with t, and the peak rank of an iterate $\boldsymbol{X}_t$ is often much larger than the rank of the solution of (6).

3.2 The CGM Stopping Rule

The CGM algorithm admits a simple stopping criterion. Given a suboptimality parameter $\varepsilon > 0$, we halt the CGM iteration when the duality gap $\delta_t \le \varepsilon$:

$$\delta_t = \langle \mathcal{A}\boldsymbol{X}_t - \mathcal{A}\boldsymbol{H}_t, \nabla f(\mathcal{A}\boldsymbol{X}_t) \rangle \le \varepsilon. \tag{12}$$

Let $\boldsymbol{X}_\star$ be an optimal point for (6). It is not hard to show [22, Sec. 2] that

$$f(\mathcal{A}\boldsymbol{X}_t) - f(\mathcal{A}\boldsymbol{X}_\star) \le \delta_t. \tag{13}$$

Thus, the condition (12) ensures that the objective value $f(\mathcal{A}\boldsymbol{X}_t)$ is ε-suboptimal. The CGM iterates satisfy (12) within $O(\varepsilon^{-1})$ iterations [22, Thm. 1].

3.3 The Opportunity

The CGM iteration (9)–(11) requires $\Theta(mn)$ storage because it maintains the $m \times n$ matrix decision variable $\boldsymbol{X}_t$. We develop a remarkable extension of CGM that provably computes a rank-r approximate solution to (6) with working storage $\Theta(d + r(m+n))$. Our approach depends on two efficiencies:

- We use the low-dimensional "dual" variable $\boldsymbol{z}_t = \mathcal{A}\boldsymbol{X}_t \in \mathbb{R}^d$ to drive the iteration.
- Instead of storing $\boldsymbol{X}_t$, we maintain a small randomized sketch with size $\Theta(r(m+n))$.

It is easy to express the CGM iteration in terms of the "dual" variable $\boldsymbol{z}_t = \mathcal{A}\boldsymbol{X}_t$. We can obviously rewrite the formula (10) for computing the rank-one update direction $\boldsymbol{H}_t$ in terms of $\boldsymbol{z}_t$. We obtain an update rule for $\boldsymbol{z}_t$ by applying the linear map $\mathcal{A}$ to (11). Likewise, the stopping criterion (12) can be evaluated using $\boldsymbol{z}_t$ and $\boldsymbol{H}_t$. Under the black-box model (8), the dual formulation of CGM has storage cost $\Theta(m + n + d)$.

Yet the dual formulation has a flaw: it "solves" the problem (6), but we do not know the solution!

Indeed, we must also track the evolution (11) of the primal decision variable $\boldsymbol{X}_t$. In the next subsection, we summarize a randomized sketching method [35] that allows us to compute an accurate rank-r approximation of $\boldsymbol{X}_t$ but operates with storage $\Theta(r(m+n))$.

4 MATRIX SKETCHING

Suppose that $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ is a matrix that is presented to us as a stream of linear updates, as in (11). For a parameter $r \ll \min\{m, n\}$, we wish to maintain a small sketch that allows us to compute a rank-r approximation of the final matrix $\boldsymbol{X}$. Let us summarize an approach developed in our paper [35].

4.1 The Randomized Sketch

Draw and fix two independent standard normal matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$ where

$$\begin{aligned} \boldsymbol{\Omega} &\in \mathbb{R}^{n \times k} \quad \text{with} \quad k = 2r + 1; \\ \boldsymbol{\Psi} &\in \mathbb{R}^{\ell \times m} \quad \text{with} \quad \ell = 4r + 3. \end{aligned} \tag{14}$$

The sketch consists of two matrices $\boldsymbol{Y}$ and $\boldsymbol{W}$ that capture the range and co-range of $\boldsymbol{X}$:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega} \in \mathbb{R}^{m \times k} \quad \text{and} \quad \boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{X} \in \mathbb{R}^{\ell \times n}. \tag{15}$$

We can efficiently update the sketch $(\boldsymbol{Y}, \boldsymbol{W})$ to reflect a rank-one linear update to $\boldsymbol{X}$ of the form

$$\boldsymbol{X} \leftarrow \beta_1 \boldsymbol{X} + \beta_2 \boldsymbol{u}\boldsymbol{v}^*. \tag{16}$$

Both the storage cost for the sketch and the arithmetic cost of an update are $\Theta(r(m+n))$.

4.2 The Reconstruction Algorithm

The following procedure yields a rank-r approximation $\hat{\boldsymbol{X}}$ of the matrix $\boldsymbol{X}$ stored in the sketch (15).

$$\boldsymbol{Q} = \texttt{orth}(\boldsymbol{Y});\ \boldsymbol{B} = (\boldsymbol{\Psi}\boldsymbol{Q})^\dagger \boldsymbol{W};\ \hat{\boldsymbol{X}} = \boldsymbol{Q}[\boldsymbol{B}]_r. \tag{17}$$

The matrix $\boldsymbol{Q}$ has orthonormal columns that span the range of $\boldsymbol{Y}$. The extra storage costs of the reconstruction are negligible; its arithmetic cost is $\Theta(r^2(m+n))$. See [35, §4.2] for the intuition behind this method. It achieves the following error bound.

Theorem 1 (Reconstruction error [35, Thm. 5.1])**.** *Fix a target rank r. Let $\boldsymbol{X}$ be a matrix, and let $(\boldsymbol{Y}, \boldsymbol{W})$ be a sketch of $\boldsymbol{X}$ of the form* (14)–(15)*. The procedure* (17) *yields a rank-r matrix $\hat{\boldsymbol{X}}$ with*

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}} \leq 3\sqrt{2}\,\|\boldsymbol{X} - [\boldsymbol{X}]_r\|_{\mathrm{F}}\,.$$

Similar bounds hold with high probability.

Remarks. The sketch size parameters (k, ℓ) appearing in (14) are recommended to balance storage against reconstruction quality. See [35] and our follow-up work for more details and for other sketching methods.

5 SKETCHING + CGM

We are now prepared to present SketchyCGM, a storage-optimal extension of the CGM algorithm for the convex problem (6). This method delivers a provably accurate low-rank approximation to a solution of (6). See Algorithm 1 for complete pseudocode.

5.1 The SketchyCGM Iteration

Fix the suboptimality ε and the rank r. Draw and fix standard normal matrices $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{\ell\times m}$ as in (14). Initialize the iterate and the sketches:

$$\boldsymbol{z}_0 = \boldsymbol{0}_d;\ \boldsymbol{Y}_0 = \boldsymbol{0}_{m\times k};\ \text{and}\ \boldsymbol{W}_0 = \boldsymbol{0}_{\ell\times n}. \tag{18}$$

At each iteration $t = 0, 1, 2, \ldots$, compute an update direction via Lanczos or via randomized SVD [17]:

$$\begin{aligned}(\boldsymbol{u}_t, \boldsymbol{v}_t) &= \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\boldsymbol{z}_t)));\\ \boldsymbol{h}_t &= \mathcal{A}(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*).\end{aligned} \tag{19}$$

Set the learning rate $\eta_t = 2/(t+2)$. Update the iterate and the two sketches:

$$\begin{aligned}\boldsymbol{z}_{t+1} &= (1-\eta_t)\boldsymbol{z}_t + \eta_t \boldsymbol{h}_t;\\ \boldsymbol{Y}_{t+1} &= (1-\eta_t)\boldsymbol{Y}_t + \eta_t(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*)\boldsymbol{\Omega};\\ \boldsymbol{W}_{t+1} &= (1-\eta_t)\boldsymbol{W}_t + \eta_t \boldsymbol{\Psi}(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*).\end{aligned} \tag{20}$$

The iteration continues until it triggers the stopping criterion:

$$\langle \boldsymbol{z}_t - \boldsymbol{h}_t,\, \nabla f(\boldsymbol{z}_t)\rangle \leq \varepsilon. \tag{21}$$

At any iteration t, we can form a rank-r approximate solution $\hat{\boldsymbol{X}}_t$ of the model problem (6) by invoking the procedure (17) with $\boldsymbol{Y} = \boldsymbol{Y}_t$ and $\boldsymbol{W} = \boldsymbol{W}_t$.

5.2 Guarantees

Suppose that the CGM iteration (9)–(11) generates the sequence $(\boldsymbol{X}_t : t = 0, 1, 2, \ldots)$ of decision variables and the sequence $(\boldsymbol{H}_t : t = 0, 1, 2, \ldots)$ of update directions. It is easy to verify that the SketchyCGM iteration (18)–(20) maintains the loop invariants

$$\begin{aligned}\boldsymbol{z}_t &= \mathcal{A}\boldsymbol{X}_t \quad \text{and} \quad \boldsymbol{h}_t = \mathcal{A}\boldsymbol{H}_t;\\ \boldsymbol{Y}_t &= \boldsymbol{X}_t\boldsymbol{\Omega} \quad \text{and} \quad \boldsymbol{W}_t = \boldsymbol{\Psi}\boldsymbol{X}_t.\end{aligned} \tag{22}$$

In view of the inequality (13) and the invariant (22), the stopping rule (21) ensures that $\boldsymbol{X}_t$ is an ε-suboptimal solution to (6) when the iteration halts. Furthermore, Theorem 1 ensures that the computed solution $\hat{\boldsymbol{X}}_t$ is a near-optimal rank-r approximation of $\boldsymbol{X}_t$ at each time t.

5.3 Storage Costs

The total storage cost is $\Theta(d + r(m+n))$ for the dual variable $\boldsymbol{z}_t$, the random matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$, and the sketch $(\boldsymbol{Y}, \boldsymbol{W})$. Owing to the black-box assumption (8), the algorithm completes the singular vector computations in (19) with $\Theta(d+m+n)$ working storage. At no point during the iteration do we instantiate an $m \times n$ matrix! Arithmetic costs are on the same order as the standard version of CGM.

5.4 Convergence Results for SketchyCGM

SketchyCGM is a provably correct method for computing a low-rank approximation of a solution to (6).

Theorem 2. *Suppose that the iterates $\boldsymbol{X}_t$ from the CGM iteration* (9)–(11) *converge to a matrix $\boldsymbol{X}_{\mathrm{cgm}}$. Let $\hat{\boldsymbol{X}}_t$ be the rank-r reconstruction of $\boldsymbol{X}_t$ produced by SketchyCGM. Then*

$$\lim_{t\to\infty} \mathbb{E}\,\|\hat{\boldsymbol{X}}_t - \boldsymbol{X}_{\mathrm{cgm}}\|_{\mathrm{F}} \leq 3\sqrt{2}\,\|\boldsymbol{X}_{\mathrm{cgm}} - [\boldsymbol{X}_{\mathrm{cgm}}]_r\|_{\mathrm{F}}\,.$$

In particular, if $\operatorname{rank}(\boldsymbol{X}_{\mathrm{cgm}}) \leq r$*, then*

$$\mathbb{E}\,\|\hat{\boldsymbol{X}}_t - \boldsymbol{X}_{\mathrm{cgm}}\|_{\mathrm{F}} \to 0.$$

SketchyCGM always works in the fundamental case where each solution of (6) has low rank.

Theorem 3. *Suppose that the solution set $S_\star$ of the problem* (6) *contains only matrices with rank r or less. Then SketchyCGM attains* $\mathbb{E}\,\operatorname{dist}_{\mathrm{F}}(\hat{\boldsymbol{X}}_t, S_\star) \to 0$.

Suppose that the optimal point of (6) is stable in the sense that the value of the objective function increases as we depart from optimality. Then the SketchyCGM estimates converge at a controlled rate.

Theorem 4. *Fix $\kappa > 0$ and $\nu > 0$. Suppose the (unique) solution $\boldsymbol{X}_\star$ of* (6) *has* $\operatorname{rank}(\boldsymbol{X}_\star) \leq r$ *and*

$$f(\mathcal{A}\boldsymbol{X}) - f(\mathcal{A}\boldsymbol{X}_\star) \geq \kappa\,\|\boldsymbol{X} - \boldsymbol{X}_\star\|_{\mathrm{F}}^{\nu} \tag{23}$$

Algorithm 1 SketchyCGM for model problem (6)

Input: Data for (6); suboptimality ε; target rank r
Output: Rank-r approximate solution $\hat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ of (6) in factored form

function SketchyCGM
  SketchInit(m, n, r)
  $\boldsymbol{z} \leftarrow \boldsymbol{0}$
  for $t \leftarrow 0, 1, 2, 3, \ldots$ **do**
    $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\boldsymbol{z})))$
    $\boldsymbol{h} \leftarrow \mathcal{A}(-\alpha \boldsymbol{u}\boldsymbol{v}^*)$
    if $\langle \boldsymbol{z} - \boldsymbol{h}, \nabla f(\boldsymbol{z}) \rangle \le \varepsilon$ **then break for**
    $\eta \leftarrow 2/(t+2)$
    $\boldsymbol{z} \leftarrow (1-\eta)\boldsymbol{z} + \eta \boldsymbol{h}$
    SketchCGMUpdate$(-\alpha \boldsymbol{u}, \boldsymbol{v}, \eta)$
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow$ SketchReconstruct( )
  return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$

Methods for Sketch object

function SketchInit$(m,\ n,\ r)$
  $k \leftarrow 2r+1$ and $\ell \leftarrow 4r+3$
  $\boldsymbol{\Omega} \leftarrow \texttt{randn}(n, k)$ and $\boldsymbol{\Psi} \leftarrow \texttt{randn}(\ell, m)$
  $\boldsymbol{Y} \leftarrow \texttt{zeros}(m, k)$ and $\boldsymbol{W} \leftarrow \texttt{zeros}(\ell, n)$

function SketchCGMUpdate$(\boldsymbol{u},\ \boldsymbol{v},\ \eta)$
  $\boldsymbol{Y} \leftarrow (1-\eta)\boldsymbol{Y} + \eta \boldsymbol{u}(\boldsymbol{v}^*\boldsymbol{\Omega})$
  $\boldsymbol{W} \leftarrow (1-\eta)\boldsymbol{W} + \eta(\boldsymbol{\Psi}\boldsymbol{u})\boldsymbol{v}^*$

function SketchReconstruct( )
  $\boldsymbol{Q} \leftarrow \texttt{orth}(\boldsymbol{Y})$
  $\boldsymbol{B} \leftarrow (\boldsymbol{\Psi}\boldsymbol{Q}) \backslash \boldsymbol{W}$
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \texttt{svds}(\boldsymbol{B}, r)$
  return $(\boldsymbol{Q}\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$

for all feasible $\boldsymbol{X}$. Then we have the error bound

$$\mathbb{E}\,\|\hat{\boldsymbol{X}}_t - \boldsymbol{X}_\star\|_{\mathrm{F}} \le 6\left(\frac{2C\kappa^{-1}}{t+2}\right)^{1/\nu} \quad \text{for } t = 0, 1, 2, \ldots$$

where C is the curvature constant [22, Eqn. (3)] of the problem (6).

See the supplement for the proofs of these results.

5.5 SketchyCGM for PSD Optimization

Next, we present a generalization of the convex phase retrieval problem (5). Consider the convex template

$$\underset{\boldsymbol{X} \in \mathbb{C}^{n\times n}}{\text{minimize}}\ f(\mathcal{A}\boldsymbol{X}) \quad \text{s.t.} \quad \operatorname{tr}\boldsymbol{X} \le \alpha,\ \boldsymbol{X} \succcurlyeq \boldsymbol{0}. \tag{24}$$

As before, $\mathcal{A} : \mathbb{C}^{n\times n} \to \mathbb{C}^d$ is a linear map, and $f : \mathbb{C}^d \to \mathbb{R}$ is a differentiable convex function.

It is easy to adapt SketchyCGM to handle (24) instead of (6). To sketch the complex psd matrix variable, we follow the approach described in [35, Sec. 7.3]. We also make small changes to the SketchyCGM iteration. Replace the computation (19) with

$$(\lambda_t, \boldsymbol{u}_t) = \texttt{MinEig}(\mathcal{A}^*(\nabla f(\boldsymbol{z}_t)));$$
$$\boldsymbol{h}_t = \begin{cases} \mathcal{A}(\alpha \boldsymbol{u}_t \boldsymbol{u}_t^*), & \lambda_t \le 0 \\ \boldsymbol{0}, & \text{otherwise.} \end{cases}$$

`MinEig` returns the minimum eigenvalue λ_t and an associated eigenvector $\boldsymbol{u}_t$ of a conjugate symmetric matrix. This variant behaves the same as SketchyCGM.

6 COMPUTATIONAL EVIDENCE

In this section, we demonstrate that SketchyCGM is a practical algorithm for convex low-rank matrix optimization. We focus on phase retrieval problems because they provide a dramatic illustration of the power of storage-optimal convex optimization. The supplementary material adduces additional examples, including some matrix completion problems.

6.1 Synthetic Phase Retrieval Problems

To begin, we show that our approach to solving the convex phase retrieval problem (5) has better memory scaling than other convex optimization methods.

We compare five convex optimization algorithms: the classic proximal gradient method (PGM) [30]; the Auslender–Teboulle (AT) accelerated method [2]; the classic CGM algorithm [22]; a storage-efficient CGM variant (ThinCGM) [37] based on low-rank SVD updating; and the psd variant of SketchyCGM from §5.5 with rank parameter $r = 1$.

All five methods solve (24) reliably. The proximal methods (PGM and AT) perform a full eigenvalue decomposition of the iterate at each step, but they can be accelerated by adaptively choosing the number of eigenvectors to compute. The methods based on CGM only need the top eigenvector, so they perform less arithmetic per iteration.

To compare the storage costs of the five algorithms, let us consider a synthetic phase retrieval problem. We draw a vector $\boldsymbol{x}_\natural \in \mathbb{C}^n$ from the complex standard normal distribution. Then we acquire $d = 10n$ phaseless measurements (3), corrupted with independent Gaussian noise so that the SNR is 20 dB. The measurement vectors $\boldsymbol{a}_i$ derive from a coded diffraction pattern; see the supplement for details. We solve the convex problem (5) with α equal to the average of the measurements b_i; see [37, Sec. II]. Then we compute a top eigenvector $\boldsymbol{x}_\star$ of the solution.

Figure 1(A) displays storage costs for each algorithm as the signal length n increases. We approximate memory

usage by reporting the total workspace allocated by MATLAB for the algorithm. PGM, AT, and CGM have static allocations, but they use a matrix variable of size n^2. ThinCGM attempts to maintain a low-rank approximation of the decision variable, but the rank increases steadily, so the algorithm fails after $n = 10^5$. In contrast, SketchyCGM has a static memory allocation of $\Theta(n)$. It already offers the best memory footprint for $n = 10^2$, and it still works when $n = 10^6$.

In fact, SketchyCGM can tackle even larger problems. We were able to reconstruct an image with $n = 3{,}264 \times 2{,}448 \approx 7.99 \cdot 10^6$ pixels, treated as a vector $\boldsymbol{x}_\natural \in \mathbb{C}^n$, given $d = 20n$ noiseless coded diffraction measurements, as above. Figure 1(B) plots the convergence of the relative error: $\min_{\phi \in \mathbb{R}} \|\mathrm{e}^{\mathrm{i}\phi}\hat{\boldsymbol{x}}_t - \boldsymbol{x}_\natural\| \,/\, \|\boldsymbol{x}_\natural\|$, where $\hat{\boldsymbol{x}}_t$ is a top eigenvector of the SketchyCGM iterate $\hat{\boldsymbol{X}}_t$. After 150 iterations, the algorithm produced an image with relative error 0.0290 and with PSNR 36.19 dB. Thus, we can solve (24) when the psd matrix variable $\boldsymbol{X}$ has $6.38 \cdot 10^{13}$ complex entries!

As compared to other convex optimization algorithms, the main weakness of SketchyCGM is the $O(\varepsilon^{-1})$ iteration count. Some algorithms, such as AT, can achieve $O(\varepsilon^{-1/2})$ iteration count, but they are limited to smaller problems. Closing this gap is an important question for future work.

6.2 Fourier Ptychography

Up to now, it has not been possible to attack phase retrieval problems of a realistic size by solving the convex formulation (5). As we have shown, current convex optimization algorithms cannot achieve scale. Instead, many researchers apply nonconvex heuristics to solve phase retrieval problems [9, 14, 21, 26]. These heuristics can produce reasonable solutions, but they require extensive tuning and have limited effectiveness. In this section, we demonstrate that, without any modification, SketchyCGM can solve a phase retrieval problem from an application in microscopy. Furthermore, it produces an image that is superior to the results obtained using major nonconvex heuristics.

We study a phase retrieval problem that arises from an imaging modality called Fourier ptychography (FP) [21]. The authors of [21] provided measurements of a slide containing human blood cells from a working FP system. We treat the sample as an image with $n = 25{,}600$ pixels, which we represent as a vector $\boldsymbol{x}_\natural \in \mathbb{C}^n$. The goal is to reconstruct the *phase* of the image, which roughly corresponds with the thickness of the sample at a given location.

The data consists of 29 illuminations, each containing 6,400 pixels. The number of measurements $d = 185{,}600$. The measurement vectors $\boldsymbol{a}_i$ are obtained from windowed discrete Fourier transforms. We can formulate the problem of reconstructing the sample $\boldsymbol{x}_\natural$ using the convex phase retrieval template (5) with the parameter $\alpha = 1{,}400$. In this instance, the psd matrix variable $\boldsymbol{X} \in \mathbb{C}^{n \times n}$ has $6.55 \cdot 10^8$ complex entries.

To solve (5), we run the SketchyCGM variant from §5.5 with the rank parameter $r = 1$ for 10,000 iterations. We factor the rank-one matrix output to obtain an approximation $\boldsymbol{x}_\star$ of the sample. Figure 2(A) displays the phase of the reconstruction $\boldsymbol{x}_\star$.

Figure 2 also includes comparisons with two nonconvex heuristics. The authors of [21] provided a reconstruction obtained via the Burer–Monteiro method [6]. We also applied Wirtinger Flow [9] with the recommended parameters. SketchyCGM yields a smooth and detailed phase reconstruction. Burer–Monteiro produces severe artifacts, which suggest an unphysical oscillation in the thickness of the sample. Wirtinger Flow fails completely. These results are consistent with [21, Fig. 4], which indicates 5–10 dB improvement of convex optimization over heuristics.

The quality of phase reconstruction can be essential for scientific purposes. In this particular example, some of the blood cells are infected with malaria parasites (Figure 2(A), red boxes). Diagnosis is easier when the visual acuity of the reconstruction is high.

The supplement contains further details and results on Fourier ptychographic imaging.

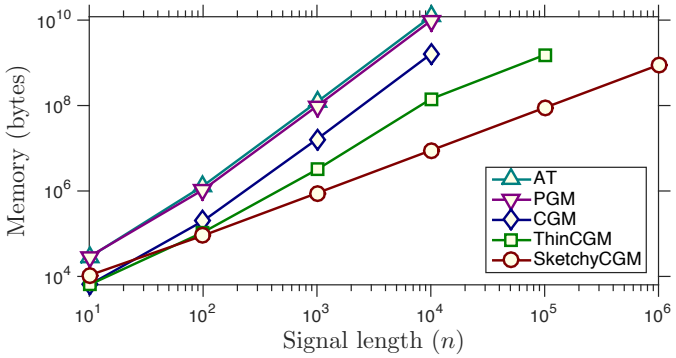
7 DISCUSSION

We have shown that it is possible to construct a low-rank approximate solution to a large-scale matrix optimization problem by sketching the decision variable. Let us contrast our approach against other low-storage techniques for large-scale optimization.

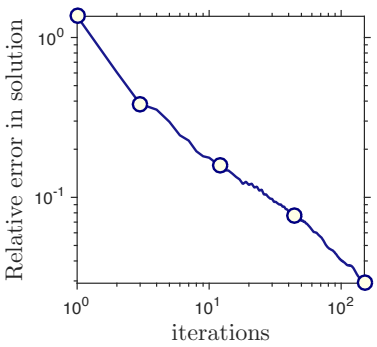
Sketchy Decisions. To the best of our knowledge, there are no direct precedents for the idea and realization of an optimization algorithm that sketches the decision variable. This work does partake in a broader vision that randomization can be used to design numerical algorithms [17, 25, 36].

Researchers have considered **sketching the problem data** as a way to reduce the size of a problem specification in exchange for additional error. This idea dates to the paper of Sarlós [31]; see also [25, 28, 36]. There are also several papers [1, 12, 27] in which researchers try to improve the computational or storage footprint of convex optimization methods by **sketching internal variables**, such as Hessians.

None of these approaches address the core issue that

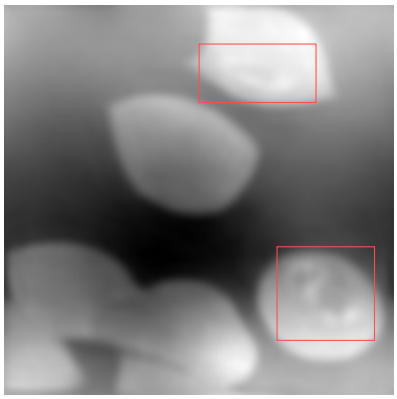


(A) Memory usage for five algorithms

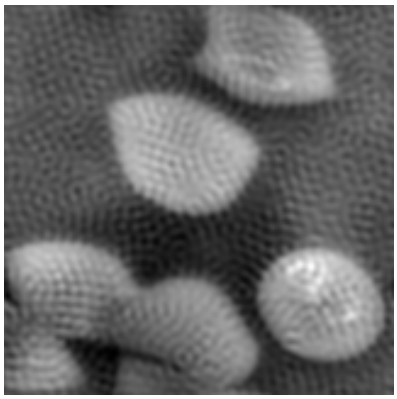


(B) Convergence on largest problem

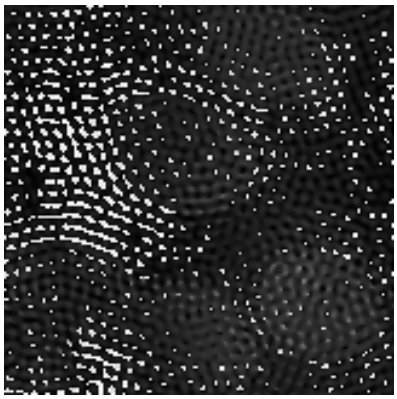
FIGURE 1: *Memory Usage and Convergence.* (A) Memory scaling for five convex optimization algorithms applied to a synthetic instance of the convex phase retrieval problem (5). (B) Relative ℓ_2 error achieved by SketchyCGM for convex phase retrieval of a synthetic signal of length $n = 8 \cdot 10^6$. See §6.2 for further details.



(A) SketchyCGM



(B) Burer–Monteiro [6, 21]



(C) Wirtinger Flow [9]

FIGURE 2: *Imaging by Fourier Ptychography.* Three algorithms for Fourier ptychographic imaging via phase retrieval. Brightness indicates the complex phase of a pixel, which roughly corresponds with the thickness of the sample. Only relative differences in brightness are meaningful. Red boxes mark malaria parasites in blood cells.

concerns us: the decision variable may require much more storage than the solution.

Dropping Nonconvexity. We have already discussed a major trend in which researchers develop algorithms that attack **nonconvex reformulations** of a problem. For example, see [4–6, 23]. The main advantage is to reduce the size of the decision variable; some methods also have the ancillary benefit of rapid local convergence. On the other hand, these algorithms are provably correct only under strong statistical assumptions on the problem data.

Prospects. Our work shows that convex optimization need not have high storage complexity for problems with a compact specification and a simple solution. In particular, for low-rank matrix optimization, storage is no reason to drop convexity.

It has not escaped our notice that the specific pairing of sketching and CGM that we have postulated immediately suggests a possible mechanism for solving other structured convex programs using optimal storage.

Acknowledgments

JAT and MU were supported in part by ONR Awards N00014-11-1-0025 and N00014-17-1-2146 and the Gordon & Betty Moore Foundation. VC and AY were supported in part by the European Commission under Grant ERC Future Proof, SNF 200021-146750, and SNF CRSII2-147633.

References

[1] N. Agarwal, B. Bullins, and E. Hazan. Second-order stochastic optimization in linear time. Available at `http://arXiv.org/abs/1602.03943`, Feb. 2016.

[2] A. Auslender and M. Teboulle. Interior gradient and

proximal methods for convex and conic optimization. *SIAM J. Optim.*, 16(3):697–725 (electronic), 2006.

[3] R. Balan, B. G. Bodmann, P. G. Casazza, and D. Edidin. Painless reconstruction from magnitudes of frame coefficients. *J. Fourier Anal. Appl.*, 15(4): 488–501, 2009.

[4] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. *J. Mach. Learn. Res.*, 49:1–53, 2016.

[5] N. Boumal, V. Voroninski, and A.S. Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. In *Adv. Neural Information Processing Systems 29*, pages 2757–2765. Curran Associates, Inc., 2016.

[6] S. Burer and R. D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program.*, 95(2, Ser. B): 329–357, 2003.

[7] E. J. Candès and Y. Plan. Matrix completion with noise. *Proc. IEEE*, 98(6):925–936, 2010.

[8] E. J. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski. Phase retrieval via matrix completion. *SIAM J. Imaging Sci.*, 6(1):199–225, 2013.

[9] E. J. Candès, X. Li, and M. Soltanolkotabi. Phase retrieval via Wirtinger Flow: Theory and algorithms. *IEEE Trans. Inform. Theory*, 61(4):1985–2007, 2015.

[10] A. Chai, M. Moscoso, and G. Papanicolaou. Array imaging using intensity-only measurements. *Inverse Problems*, 27(1):015005, 2011.

[11] K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms*, 6(4):Art. 63, 30, 2010.

[12] M. A. Erdoğdu and A. Montanari. Convergence rates of sub-sampled Newton methods. In *Adv. Neural Information Processing Systems 28*, pages 3052–3060. Curran Associates, Inc., 2015.

[13] M. Fazel. *Matrix rank minimization with applications*. PhD Dissertation, Stanford Univ., Palo Alto, 2002.

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

[15] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Res. Logist. Quart.*, 3:95–110, 1956.

[16] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[17] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

[18] E. Hazan. Sparse approximate solutions to semidefinite programs. In *Proc. 8th Latin American Theoretical Informatics Symposium*, Apr. 2008.

[19] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, 10(3): 673–696, 2000.

[20] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optim.*, 6(2):342–361, 1996.

[21] R. Horstmeyer, R. Y. Chen, X. Ou, B. Ames, J. A. Tropp, and C. Yang. Solving ptychography with a convex relaxation. *New J. Physics*, 17(5):053044, 2015.

[22] M. Jaggi. Revisiting Frank–Wolfe: Projection-free sparse convex optimization. In *Proc. 30th Intl. Conf. Machine Learning*, Atlanta, 2013.

[23] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *STOC '13: Proc. 45th Ann. ACM Symp. Theory of Computing*, pages 665–674, Palo Alto, Dec. 2013.

[24] E. S. Levitin and B. T. Poljak. Minimization methods in the presence of constraints. *Ž. Vyčisl. Mat. i Mat. Fiz.*, 6:787–823, 1966. ISSN 0044-4669.

[25] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[26] Netrapalli, P. Jain, and S. Sanghavi. Phase retrieval using alternating minimization. *IEEE Trans. Signal Processing*, 63(18):4814–4826, 2015.

[27] M. Pilanci and M. Wainwright. Newton sketch: A linear-time optimization algorithm with linear-quadratic convergence. Available at `http://arXiv.org/abs/1505.02250`, May 2015.

[28] M. Pilanci and M. J. Wainwright. Randomized sketches of convex programs with sharp guarantees. *IEEE Trans. Inform. Theory*, 61(9):5096–5115, 2015.

[29] N. Rao, P. Shah, and S. Wright. Conditional gradient with enhancement and truncation for atomic-norm regularization. Available at `http://people.inf.ethz.ch/jaggim/NIPS-workshop-FW-greedy/papers/rao_shah_wright_final.pdf`, 2015.

[30] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5): 877–898, 1976.

[31] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proc. 47th Ann. IEEE Symp. Foundations of Computer Science*, Berkeley, 2006.

[32] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorizations. In *Adv. Neural Information Processing Systems 17*, Vancouver, Dec. 2004.

[33] K.-C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific J. Optim.*, 6(615-640):15, 2010.

[34] J. A. Tropp. Convex recovery of a structured signal from independent random linear measurements. In G. Pfander, editor, *Sampling Theory, a Renaissance*, chapter 2, pages 67–101. Birkhäuser, 2015.

[35] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Randomized single-view algorithms for low-rank matrix reconstruction. ACM Report 2017-01, Caltech, Jan. 2017. Available at `http://arXiv.org/abs/1609.00048`.

[36] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):iv+157, 2014.

[37] A. Yurtsever, Y.-P. Hsieh, and V. Cevher. Scalable convex methods for phase retrieval. In *6th IEEE Intl. Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, 2015.