Abstract

This work proposes a rapid algorithm, BM-Global, for nuclear-norm-regularized convex and low-rank matrix optimization problems. BM-Global efficiently decreases the objective value via low-cost steps leveraging the nonconvex but smooth Burer-Monteiro (BM) decomposition, while effectively escapes saddle points and spurious local minima ubiquitous in the BM form to obtain guarantees of fast convergence rates to the global optima of the original nuclear-norm-regularized problem through aperiodic inexact proximal gradient steps on it. The proposed approach adaptively adjusts the rank for the BM decomposition and can provably identify an optimal rank for the BM decomposition problem automatically in the course of optimization through tools of manifold identification. BM-Global hence also spends significantly less time on parameter tuning than existing matrix-factorization methods, which require an exhaustive search for finding this optimal rank. Extensive experiments on real-world large-scale problems of recommendation systems, regularized kernel estimation, and molecular conformation confirm that BM-Global can indeed effectively escapes spurious local minima at which existing BM approaches are stuck, and is a magnitude faster than state-of-the-art algorithms for low-rank matrix optimization problems involving a nuclear-norm regularizer.
1. Introduction

Consider the following regularized convex matrix optimization problem

$$\min_{X \in \mathbb{R}^{m \times n}} F(X) := f(X) + \Psi(X), \quad (CVX)$$

where the loss term $f$ is lower-bounded, convex, and differentiable with Lipschitz-continuous gradient, and the regularizer $\Psi$ is convex and has the form

$$\Psi(X) := \lambda \|X\|_* + \delta_X(X), \quad X \in \mathbb{R}^{m \times n}, \quad (1)$$

with $X$ being a closed and convex subset of $\mathbb{R}^{m \times n}$. Clearly, in this case, $\Psi$ is nonsmooth, convex, proper, and closed.\(^1\) Without loss of generality, we assume that $m \leq n$ throughout, which can be achieved easily by conducting a matrix transpose if necessary. For our case of (1), as long as $\lambda$ is properly selected, a low-rank optimal solution to (CVX) exists, since the nuclear norm is exactly applying the $\ell_1$-norm to the singular values of the given matrix. In practice, singular value decomposition (SVD) for a non-symmetric matrix $X$ is calculated through the eigendecomposition of the symmetric matrix $XX^\top$ (as we assume $m \leq n$), and thus computation of SVDs and of eigendecompositions are nearly identical. We will therefore summarize these two situations simply as requiring eigendecompositions.

We focus on large-scale problems such that $mn$ is extremely large, so forming a (possibly dense) matrix $X \in \mathbb{R}^{m \times n}$ explicitly is spatially and computationally expensive, if not infeasible, and thus a low-rank solution is necessary for practical reasons. The nuclear-norm regularization hence serves to induce a low-rank structure in any solution $X^*$ to (CVX). On the other hand, we assume that $\nabla f(X)$ is either structured or extremely sparse so that $\nabla f(X)v$ for any vector $v$ can be computed efficiently. This is necessary for the execution of an inexact proximal gradient (PG) step; see details in Section 3.

To deal with the high problem dimensionality in (CVX), a popular approach is the Burer-Monteiro (BM) decomposition (Burer and Monteiro, 2003) that explicitly writes $X$ as a product of two low-rank matrices with a pre-specified rank $k \leq m$. Explicitly, we get

$$\min_{W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{n \times k}} F(W, H) := f(WH^\top) + \Psi(WH^\top). \quad (BM)$$

The spatial cost of $O(mn)$ for storing $X$ in (CVX) is then reduced to $O((m+n)k)$ in (BM). Numerous efficient algorithms for solving (BM) are therefore proposed.

Among applications of (CVX), one of the most prominent example, and also our motivating problem, is the following low-rank matrix completion problem whose target is to recover the whole ground truth matrix $A \in \mathbb{R}^{m \times n}$ from its observed entries enumerated by an index set $\Omega$:

$$\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \|P_\Omega(X - A)\|_F^2 + \lambda \|X\|_* \quad (MC)$$

where $\|\cdot\|_F$ is the Frobenius norm and

$$(P_\Omega(A))_{i,j} = \begin{cases} A_{i,j} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}.$$ 

\(^1\) Note that when $m = n$ and $X$ is a subset of $\mathbb{S}^n_+$, $\Psi$ is convex, proper, and closed even if $\lambda < 0$, as in that case the nuclear norm becomes the trace of $X$, which is an affine function of $X$. Otherwise, we will need $\lambda \geq 0$ to make $\Psi$ convex.
It is well-known that the factorized form in (BM) of (MC) for a given rank $k$ is

$$\min_{W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{n \times k}} \frac{1}{2} \| P_\Omega (W^T H - A) \|^2_F + \frac{\lambda}{2} (\| W \|^2_F + \| H \|^2_F),$$

which is often called the matrix factorization (MF) problem in the machine learning community. One can easily show that the global optimal objectives of (MC) and (MF) coincide whenever $k$ is large enough such that there is at least one optimal solution $X^*$ of (MC) with $\text{rank}(X^*) \leq k$. (See Lemma 1.) Apparently, even the objective evaluation for (MC) requires an eigendecomposition that costs $O(m^3)$, while for (MF), objective evaluation and variable updates all require cost of only $O((m + n)k)$.

Unfortunately, the lower computational and spatial costs of solving (MF) and thus the more general (BM) comes with a price. A disadvantage of the BM method is that the rank $k$ needs to be specified in advance, and a good value for $k$ can be hard to estimate a priori. Another even more severe issue is that the problem (MF) is a nonconvex one, meaning that algorithms for solving it could get stuck at spurious local optima (local but not global optima) or saddle points that have a zero gradient. Such points can give terrible performance for predicting missing entries. The simplest example would be that for any $k > 0$, letting $W$ and $H$ be matrices of all zeros in (MF) will directly generate a saddle point, but this clearly is not a solution in general. It is also recently shown by Yalcin et al. (2022); O’Carroll et al. (2022) that there are indeed problems with a sufficiently large $k$ that still possess spurious local minima with a terribly large objective value, and thus how to escape from such saddle points and spurious local minima is critical issue for the BM approach to produce satisfactory performance.

On the other hand, the convex problem (MC) or (CVX) can be solved directly through PG-type algorithms that are able to find the global optima (Toh and Yun, 2010), but the cost of the eigendecomposition in the proximal operation is extremely expensive, even if only a subset of singular values/vectors is required. State-of-the-art methods for (MC) like those by Hsieh and Olsen (2014); Yao and Kwok (2018) thus resort to approximate eigendecompositions computed through the power method to reduce the time cost of eigendecompositions. The power method also has the benefit that only $XV$ for some thin matrix $V$ is needed at each iteration, so explicit computation of $X$ is not needed. However, we observe that in practice, usually the convex approaches based on PG tend to be rather slow because their computational cost per iteration is still not comparable to those for (MF) or (BM).

In this work, we propose a highly-efficient algorithm, BM-Global, that combines the advantages of both approaches with theoretical guarantees. Our method fully utilizes the computational and spatial advantages of (BM) to have a running time similar to the state-of-the-art for (BM). Meanwhile, BM-Global also possesses guarantees for convergence to the global optima just like those approaches for solving (CVX). With suitable inexactness conditions for the PG steps, we also obtain a sublinear convergence rate for the general convex case, and further get faster rates when the objective function satisfies the Kurdyka-Lojasiewicz (KL) condition (Kurdyka, 1998; Lojasiewicz, 1963). Our algorithm attains these appealing guarantees through sporadically resorting to convex lifting steps that conduct one iteration of inexact PG on (CVX) to escape from saddle points and spurious local optima at which existing methods for (BM) got stuck at. We emphasize that for alternating between
inexact PG and other update steps, our convergence and rate guarantees are novel, as existing analyses for convergence and rates of inexact PG rely on geometrical properties of the PG iterates that will be destroyed when other updates are inserted.

Through the techniques of manifold identification (Hare and Lewis, 2004; Lewis and Zhang, 2013) in our analysis, another major and novel contribution of this work is that the optimal rank for (BM) will provably be found by the proposed method automatically, so no additional parameter tuning for the optimization side is required for attaining satisfactory results for practical applications of (CVX). Numerical results also show that our method is significantly faster than state-of-the-art solvers for (CVX), and can effectively escape from saddle points and spurious local minima of (BM).

1.1 Related Works

Methods for (CVX). The convex problem (CVX) falls in the category of regularized optimization, and many efficient algorithms in the literature are available. However, most methods for regularized optimization concentrate on the scenario that the proximal operation can be conducted efficiently but obtaining information of the smooth term is the major computational bottleneck, which is apparently not the case for (CVX). Practical methods specifically designed for (CVX) all take into serious account the expensive SVD involved in the nuclear norm (Toh and Yun, 2010; Hsieh and Olsen, 2014; Yao and Kwok, 2018), and they all focus on the most popular setting that \( f \) is a quadratic term. In this case, high-order methods like proximal (quasi-)Newton is useless because the subproblem has the same form as the original problem itself. Therefore, these works all consider first-order methods such as the (inexact) PG or accelerated PG (APG) methods (Nesterov, 2013; Beck and Teboulle, 2009a,b). Toh and Yun (2010) used the Lanczos method to conduct approximate SVD, and Hsieh and Olsen (2014) proposed to apply the power method for approximate SVD and to use the rank \( k \) decided by their inexact PG method to conduct another convex optimization step with respect to a subproblem of dimension \( k \times k \) after each PG step. The power method in Hsieh and Olsen (2014) effectively uses the current iterate as warmstart and is much more efficient than the Lanczos approach in Toh and Yun (2010), but the additional convex optimization step turns out to be time-consuming. To improve the efficiency of Toh and Yun (2010) and Hsieh and Olsen (2014), Yao and Kwok (2018) combined the two approaches to turn to inexact APG using the power method.

More recently, motivated by PG’s ability of manifold identification, Bareilles et al. (2022) proposed to alternate between an exact PG step and a Riemannian (truncated) Newton step on the currently identified manifold for general regularized optimization, and applied this method to a toy problem of (CVX) in their experiments. Different from ours, their usage of manifold identification is for showing that their method could obtain superlinear convergence, but their algorithm is not feasible for large-scale problems considered in this work because they considered exact PG only and required the explicit computation of \( X_t \).

Convergence of Inexact Proximal Gradient. The global convergence of our method is achieved through the safeguard of inexact PG steps, but we do not require the inexact PG step to always decrease the objective value. This feature combined with the BM phase makes the analysis difficult. Existing analyses for inexact PG Combettes (2004); Schmidt et al. (2011); Jiang et al. (2012); Hamadouche et al. (2022) utilize telescope sums of inequalities in
the form of $\|X_{t+1} - X^*\|^2 \leq \text{error term} + \|X_t - X^*\|^2 - \alpha (F(X_t) - F(X^*))$ for any $X^* \in \Omega^*$ and some $\alpha > 0$ to prove convergence and rates. Therefore, those approaches cannot allow for alternating between inexact PG and other steps because that will nullify the technique of telescope sums. On the other hand, analyses compatible with other steps like those in Scheinberg and Tang (2016); Bonettini et al. (2016); Lee and Wright (2019) require strict decreasing of the objective in the inexact PG step (either explicitly or implicitly), which imposes an additional burden.

The work of Yang et al. (2022) that applies inexact PG to the semidefinite programming (SDP) relaxation of polynomial optimization problems is probably the closest to our approach in that their method alternates between nonmonotone inexact PG and an alternative step. However, their alternative step is only accepted when it decreases the objective by an absolute amount $\epsilon > 0$, so eventually the alternative steps are always rejected when the objective converges, but we do not have such restrictions. Our analysis also provides more comprehensive convergence guarantees under different conditions as well as identification of the optimal rank using techniques totally different from that of Yang et al. (2022).

**Methods for escaping saddle points.** There has recently been a thriving interest in studying smooth optimization methods that can escape strict saddle points with at least one negative eigenvalue in the Hessian (Lee et al., 2019; Jin et al., 2017; Royer and Wright, 2018; Carmon et al., 2018; Royer et al., 2020; Agarwal et al., 2017). However, these methods are unable to deal with degenerate saddle points where the smallest eigenvalue of the Hessian is exactly zero, and neither could they deal with spurious local minima that might be arbitrarily far away from the global ones. On the other hand, our method can handle all such difficult cases appearing in (BM) by resorting to convex lifting, and Lemma 1 together with Theorem 2 show that our method indeed will converge to the global optima. Moreover, existing methods for escaping strict saddles are mainly of theoretical interest, and their empirical performance is usually not very impressive, while our method is designed for practical large-scale usage and greatly outperforms state-of-the-art methods for (CVX) and (BM) by a large margin on real-world data, as we shall see in the numerical experiments later.

**Absence of spurious stationary points for (MF).** To cope with possible spurious local minima and degenerate saddle points of matrix factorization, there is also a growing interest in analyzing its optimization landscape. Most such works consider the quadratic loss:

$$f(X) = \|P_{\Omega} (A - X)\|_F^2.$$  

(2)

Ge et al. (2016); Sun and Luo (2016); Ge et al. (2017) confined their analyses to (2) and proved the absence of spurious local minima for only the ideal case in which each $(i, j)$ belongs to $\Omega$ with a fixed probability $p \in (0, 1]$ and the observations are noiseless. But these assumptions generally fail in practice. In particular, for applications like recommendation systems, elements of $\Omega$ are already biased selections by an existing system and will never obey the independent random assumption. In addition, real-world data always contain noisy observations and measurement corruption. Chen et al. (2019); Ye and Du (2021) focused on (2) with $\Omega = \{1, \ldots, m\} \times \{1, \ldots, n\}$ only, and their techniques could be hard to generalize to other problems. Moreover, these two works discussed only the gradient descent method, which is barely used (if any) in large-scale real applications. Different from these works, we do not have any assumption on the underlying data, as it has been shown recently in Yalcin
et al. (2022); O’Carroll et al. (2022) that some problem classes of matrix factorization and SDP indeed contain numerous spurious local minima, yet our method can still find the global optima in the presence of non-strict saddle points and spurious local optima with ultra-high practical efficiency.

1.2 Organization

This paper is organized as follows. Preliminaries for our algorithmic development are provided in Section 2. In Section 3, we describe our main algorithmic framework. Section 4 provides a comprehensive theoretical analysis of our algorithm, including its global convergence to the global optima, convergence rates, and identification of the right rank for (BM) within a finite number of iterations. We then give more details in Section 5 several realizations of (CVX) in applications. Numerical experiments on real-world problems in such applications are conducted in Section 6, and finally Section 7 concludes this work. Detailed implementations for the applications used in our numerical experiments and additional experimental results are provided in the supplementary materials starting from Page 36.

2. Preliminaries

This section first lays out our notations in this work, and then provides background knowledge that will facilitate further descriptions in our development of algorithm and theory. We use $\text{tr} (\cdot)$ to denote the trace of a square matrix. For any $x \in \mathbb{R}^m$, $\text{diag}(x)$ is the $m \times m$ diagonal matrix whose diagonal entries are those in $x$. We use $\langle \cdot, \cdot \rangle$ to denote the standard inner product and $\| \cdot \|$ to denote its induced norm. In particular, for vectors $x, y \in \mathbb{R}^n$, this is the standard inner product such that $\langle x, y \rangle := \sum_{i=1}^n x_i y_i$, with the norm being the Euclidean norm, and for matrices $A, B \in \mathbb{R}^{m \times n}$, this inner product is defined as $\langle A, B \rangle := \text{tr} (A^\top B)$, where $A^\top$ is the transpose of $A$ and the corresponding norm is the Frobenius norm. We denote by $\mathbb{R}_+^m$ the nonnegative orthant in the $m$-dimensional Euclidean space, by $\mathbb{S}^n$ the set of $n \times n$ symmetric real matrices, and by $\mathbb{S}^n_+$ the cone of symmetric positive semidefinite matrices in $\mathbb{S}^n$. $I_n$ denotes the identity matrix with dimension $n \times n$, and $e_n \in \mathbb{R}^n$ is the vector of ones. The subscript $n$ is often omitted when the dimensionality is clear. For $x \in \mathbb{R}^m$, we let $\lfloor x \rfloor_+$ be its Euclidean projection onto $\mathbb{R}_+^m$, and for $X \in \mathbb{S}^n$, $[X]_+$ is its Euclidean projection onto $\mathbb{S}^n_+$. In particular, if $X$ admits an eigendecomposition $X = U \text{diag}(\Sigma) U^\top$, where $U \in \mathbb{R}^{n \times n}$ is orthonormal such that $UU^\top = U^\top U = I_n$ and $\Sigma \in \mathbb{R}^n$, we have that $[X]_+ = U \text{diag} ( [\Sigma]_+ ) U^\top$. We note that SVDs and eigendecompositions are unique up to permutations of the eigenvalues or the singular values, so we will simply say “the” SVD or “the” eigendecomposition to refer to the one such that the singular values or eigenvalues are sorted in descending order. (And this can be an arbitrary one when there are repeated singular values.) Given any set $\mathcal{S}$, we use $\delta_{\mathcal{S}}$ to denote its indicator function such that

$$
\delta_{\mathcal{S}}(x) = \begin{cases} 
0 & \text{if } x \in \mathcal{S}, \\
\infty & \text{otherwise}.
\end{cases}
$$

For any convex function $h$, we use $\partial h$ to denote its subdifferential.
Throughout this work, we will heavily use the proximal operation. Given a function \( \Psi \), this operation is defined as
\[
\text{prox}_\Psi(X) := \arg \min_X \frac{1}{2} \| X - \tilde{X} \|^2 + \Psi(\tilde{X}).
\] (3)

For \( \Psi \) convex, proper, and closed, it is well-known that (3) is well-defined and single-valued everywhere. When \( \Psi = \lambda \| \cdot \|_* \) for some \( \lambda > 0 \), (3) has a closed-form solution (Lewis and Overton, 1996). Given a matrix \( X \in \mathbb{R}^{m \times n} \) with rank \( k \) and with its SVD written as \( X = U \text{diag}(\Sigma)V^\top \) for some \( U \in \mathbb{R}^{m \times k} \), \( V \in \mathbb{R}^{n \times k} \) that are orthogonal and \( \Sigma \in \mathbb{R}^k_+ \), we have
\[
\text{prox}_{\beta\| \cdot \|_*}(X) = U \text{diag}\left( [\Sigma - \beta e]_+ \right) V^\top
\] (4)
for any \( \beta > 0 \). Similarly, if \( X \in \mathbb{S}^n \) admits an eigendecomposition \( X = U \text{diag}(\Sigma)U^\top \), we have from Lewis and Overton (1996) that for any \( \beta > 0 \),
\[
\text{prox}_{\beta (\| \cdot \|_* + \delta_{\mathbb{S}^n} \)}(X) = [X]_+ = U \text{diag}\left( [\Sigma - \beta \lambda e]_+ \right) U^\top.
\] (5)

In this paper, we focus on the two scenarios of \( \Psi = \lambda \| \cdot \|_* \) and \( \Psi = \lambda \| \cdot \|_* + \delta_{\mathbb{S}^n} \) (for \( m = n \)) for some closed and convex set \( S \). We will see in Lemma 1 that the former results in the following form of (BM):
\[
\min_{U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{n \times k}} \tilde{F}(W, H) := f\left( W H^\top \right) + \frac{\lambda}{2} \left( \| W \|_F^2 + \| H \|_F^2 \right),
\] (BM-nuclear)
while the latter leads to
\[
\min_{W \in \mathbb{R}^{n \times k}} \tilde{F}(W) := f\left( W W^\top \right) + \lambda \| W \|_F^2 + \delta_S\left( W W^\top \right).
\] (BM-PSD)

For the latter case, when we deal with (CVX), for easier calculation, we will sometimes consider the smooth term as \( f(X) := f(X) + \lambda(I, X) \) and the regularizer as \( \tilde{\Psi}(X) := \delta_{\mathbb{S}^n}(X) \) instead, which is equivalent to the original problem because the nuclear norm on a positive semidefinite matrix is exactly the trace of the same matrix.

The equivalence between the nuclear norm in (CVX) and the Frobenius norm squared in (BM-nuclear) and (BM-PSD) is formally stated in the following lemma.

**Lemma 1** Given any \( X \in \mathbb{R}^{m \times n} \), we have
\[
\| X \|_* = \min_{W,H:WH^\top = X} \frac{1}{2} \left( \| W \|_F^2 + \| H \|_F^2 \right).
\] (6)

Moreover, if the SVD of \( X \) is
\[
X = USV^\top = \sum_{i=1}^k \sigma_i u_i v_i^\top,
\]
where \( S = \text{diag}(\sigma_1, \ldots, \sigma_k) \) and \( \sigma_i > 0 \) are the singular values, \( k = \text{rank}(X) \), \( U = [u_1, \ldots, u_k] \in \mathbb{R}^{m \times k} \) and \( V = [v_1, \ldots, v_k] \in \mathbb{R}^{n \times k} \) are both orthonormal, the minima of the right-hand side of (6) are exactly those
\[
\hat{W} := \left[ \sqrt{\sigma_{\tau(1)}} u_{\tau(1)}, \ldots, \sqrt{\sigma_{\tau(k)}} u_{\tau(k)} \right], \quad \hat{H} := \left[ \sqrt{\sigma_{\tau(1)}} v_{\tau(1)}, \ldots, \sqrt{\sigma_{\tau(k)}} v_{\tau(k)} \right],
\] (7)
where $\tau$ is an arbitrary random permutation of $\{1, \ldots, k\}$. Therefore, for any global optimum $(W^*, H^*)$ of (BM-nuclear), $X^* \coloneqq W^*(H^*)^\top$ is also a global optimum to (CVX) with $\Psi = \lambda \|\cdot\|_*$, provided that there is a global optimum $\hat{X}$ of (CVX) with $\text{rank}(\hat{X}) \leq k$, and for any optimal solution $X^*$ of (CVX) with SVD $X^* = U^*\Sigma^*(V^*)^\top$, 

$$W^* = U^*(\Sigma^*)^{\frac{1}{2}}, \quad H^* = V^*(\Sigma^*)^{\frac{1}{2}}$$

form a global solution of (BM-nuclear) for any $k \geq \text{rank}(X^*)$. Likewise, for any global optimum $W^*$ of (BM-PSD), $X^* \coloneqq W^*(W^*)^\top$ is also a global optimum to (CVX) with $\Psi = \lambda \|\cdot\|_* + \delta_{\mathbb{S}_n}(\cdot) + \delta_{\mathbb{S}}(\cdot)$, provided that there is a global optimum $\hat{X}$ of (CVX) with $\text{rank}(\hat{X}) \leq k$.

(7) is directly from Rennie and Srebro (2005), and the rest of Lemma 1 are also well-known.

3. Algorithmic framework

In this section, we present a detailed description of the proposed BM-Global that can be split into two phases: the BM phase and the convex lifting phase. The high-level idea of the proposed framework is to fully utilize the efficiency in solving (BM) from the smoothness of the objective function whenever possible. When only limited progress can be further made in the BM phase with the current iterate $(\tilde{W}_t, \tilde{H}_t)$, we turn to the convex lifting phase, which conducts one step of inexact proximal gradient (PG) on (CVX) from the iterate $\tilde{X}_t = \tilde{W}_t\tilde{H}_t^\top$. In the inexact PG step, an approximate eigendecomposition algorithm is employed to obtain the next iterate $X_{t+1} = W_{t+1}H_{t+1}^\top$. The rank of the approximate eigendecomposition is dynamically increased (and the proximal step decreases the rank of its output) to guarantee that the correct rank at the optimum can be found within finite iterations of our algorithm.

We note that (7) provides a convenient way to transform an iterate of (CVX) to that of (BM-nuclear) or (BM-PSD). (Transforming the other way round is straightforward.) Although this transformation requires an eigendecomposition, we will see shortly that our convex lifting step will generate the exact eigendecomposition of its output (which is obtained from conducting an exact proximal operation on an approximate eigendecomposition of the input matrix), so the transformation can be done almost for free. We emphasize that the matrix $X = WH^\top$ is never explicitly formed when executing BM-Global.

The only requirement we put on the BM phase is a mild and implementable nonincreasing objective condition:

$$\tilde{F}(\tilde{W}_t, \tilde{H}_t) \leq F(W_t, H_t) \iff F(\tilde{W}_t\tilde{H}_t^\top) \leq F(W_tH_t^\top).$$

Therefore, we can also skip the BM phase from time to time without violating (8) when this skipping is deemed useful.

In the convex lifting phase, we emphasize that since we never explicitly store the dense matrix variable $X$ due to the high spatial cost, exact decomposition for computing the eigendecomposition becomes impractical. This is one of the major reasons to consider approximate eigendecompositions that are computed through an iterative process where each iteration of which only requires the computation of matrix-matrix products of the
form $XV$ for some thin matrix $V$. This product $XV$ can be computed efficiently without explicitly forming $X$ if $X$ can be decomposed into the sum of a low-rank matrix and a highly sparse matrix. This is another reason to consider low-rank problems promoted by the nuclear norm, as the proximal operation of the nuclear norm often leads to a low-rank iterate. See more details in (4) and (5), and Theorem 6 in Section 4.

Although state-of-the-art for (CVX) such as the works of Toh and Yun (2010); Yao and Kwok (2018) utilizes the APG method (Beck and Teboulle, 2009a; Nesterov, 2013) to obtain theoretical and practical convergence faster than that of the PG method, APG is not applicable in our framework because the meaning of its extrapolation step becomes unclear when we insert other update steps between two APG iterations. Moreover, although using a vanilla PG method in our algorithm results in a slower worst-case convergence rate than the APG method, our major workhorse in reducing the objective value efficiently is actually the BM phase, while our PG step mainly serves as a safeguard for global convergence and the mechanism to identify the correct rank. Therefore, we do not expect the PG step to provide much objective decrease empirically. In the numerical experiments, we will also see that the added BM phase indeed effectively decreases the objective value with a short running time, making the proposed algorithm outperform the APG method of Toh and Yun (2010); Yao and Kwok (2018) significantly.

### 3.1 Inexact Proximal Gradient Step

Given the iterate $\tilde{X}_t$ and a step size $\alpha_t > 0$ at the $t$th iteration, the exact PG step $\hat{X}_t^+(\alpha_t)$ is computed by

$$\hat{X}_t^+(\alpha_t) = \text{prox}_{\alpha_t \Psi}(\tilde{X}_t - \alpha_t \nabla f(\tilde{X}_t))$$

$$= \arg \min_Y \{ Q^{\alpha_t}(Y) := \langle \nabla f(\tilde{X}_t), Y - \tilde{X}_t \rangle + \frac{1}{2\alpha_t} \|Y - \tilde{X}_t\|^2_F + \Psi(Y) - \Psi(\tilde{X}_t) \}. \quad (9)$$

After finding a suitable $\alpha_t$ ensuring a sufficient decrease in $F$, the exact PG method then assigns $X_{t+1} = \hat{X}_t^+(\alpha_t)$. For our inexact scheme, we focus on the scheme that the computation of $\nabla f(\tilde{X}_t)$ and the proximal operation in (4) or (5) are exact, and the inexactness in the PG step comes from the approximate eigendecomposition. In particular, the approximate eigendecomposition is the exact eigendecomposition of a matrix approximating the original one, and thus we can easily conduct exact eigenvalue/singular value truncation in (4) or (5) of this approximation matrix.

We can therefore view the calculation of our inexact PG step with such an inexactness quantified by some $\epsilon_t \geq 0$ as

$$X_{t+1} = X_t^+(\alpha_t) = \text{prox}_{\alpha_t \Psi} \left( \tilde{Z}_t \right) = \arg \min_{\tilde{Y}} \langle \nabla f(\tilde{X}_t) + \mathcal{E}_t, \tilde{Y} - \tilde{X}_t \rangle + \frac{1}{2\alpha_t} \|\tilde{Y} - \tilde{X}_t\|^2_F + \Psi(\tilde{Y}),$$

for some $\mathcal{E}_t$ and $\tilde{Z}_t$ that satisfy

$$\|\mathcal{E}_t\|_F \leq \epsilon_t, \quad \tilde{Z}_t = \tilde{X}_t - \alpha_t \nabla f(\tilde{X}_t) + \alpha_t \mathcal{E}_t. \quad (11)$$

Such an inexactness can also be described by the following abstract representation.

$$X_{t+1} = X_t^+(\alpha_t) \in \text{prox}_{\alpha_t \Psi} \left( \tilde{X}_t - \alpha_t \nabla f(\tilde{X}_t) \right) \iff \min_{G \in \partial Q^{\alpha_t}_t(X_t^+(\alpha_t))} \|G\|_F \leq \epsilon_t. \quad (12)$$
For the stepsize choices, the only requirement of our framework is that the final stepsize \( \alpha_t \) is uniformly bounded and satisfies either of the following criteria for some given \( \delta \in (0, 1) \).

\[
f(X_{t+1}) \leq f(\tilde{X}_t) + \langle \nabla f(\tilde{X}_t), X_{t+1} - \tilde{X}_t \rangle + \frac{\delta}{\alpha_t} \| X_{t+1} - \tilde{X}_t \|_F^2, \quad \text{or} \quad (13)
\]

\[
F(X_{t+1}) \leq F(\tilde{X}_t) + \delta Q^\alpha_t(X_{t+1}). \quad (14)
\]

We note that (13) does not necessarily imply monotonically decreasing objective values, and it is actually in general independent of how accurately (12) is solved. On the other hand, although the value of \( Q^\alpha_t(X_{t+1}) \) is affected by \( \epsilon_t \), it is not necessarily negative (although the minimum of \( Q^\alpha_t(\cdot) \) is), and thus the objective value might still be nonmonotone.

To ensure that \( \alpha_t \) is bounded, we need to specify an upper \( \alpha_{\max} \) and a lower bound \( \alpha_{\min} \). It is known that if \( \nabla f \) is \( L \)-Lipschitz continuous, then (13) holds for any \( \alpha_t < 2\delta/L \), and thus we assign \( \alpha_{\min} < 2\delta/L \) to ensure that our algorithm is well-defined. As for (14), note that when \( \alpha_t \leq 1/L \), \( Q^\alpha_t(Y) \) is a majorization of \( F(Y) - F(\tilde{X}_t) \), so (14) holds. Thus, we can set \( \alpha_{\min} < 1/L \).

Since the inexact PG step is not the major tool for reducing the objective value, we simply assign a fixed step size \( \alpha_t \equiv \alpha \) in our implementation. We have also experimented with the approach of SpaRSA (Wright et al., 2009) that combines a spectral initialization of Barzilai and Borwein (1988) with backtracking line search, but its empirical performance is worse than the fixed-step variant (see the supplementary materials), likely due to the additional eigendecompositions in backtracking.

We summarize the version of our algorithm for (BM-nuclear) in Algorithm 1, and the version for (BM-PSD) can be obtained by considering only one matrix \( W \in \mathbb{R}^{m \times k} \) and replacing SVDs with eigendecompositions.

**Algorithm 1: BM-Global**

\[
\begin{align*}
\text{input} : & \lambda, \delta, \alpha_{\max} \geq \alpha_{\min} > 0 \text{ with } \alpha_{\min} < \delta/(2L) \text{ or } \alpha_{\min} < 1/L, \text{ initial rank } k, \text{ a nonnegative sequence } \{ \epsilon_t \} \text{ with } \epsilon_t \rightarrow 0 \\
\text{1 initialization: } & W_0 \in \mathbb{R}^{m \times k}, H_0 \in \mathbb{R}^{n \times k} \text{ such that } W_0 H_0^\top \in \text{dom}(\Psi). \\
\text{2 for } t = 0, \ldots \text{ do} \\
\text{3 } & (\text{BM phase}) \text{ Compute } \tilde{W}_t \in \mathbb{R}^{m \times k}, \tilde{H}_t \in \mathbb{R}^{n \times k} \text{ as an approximate solution to } (\text{BM}) \text{ (starting from } (W_t, H_t) \text{) satisfying } \tilde{W}_t \tilde{H}_t^\top \in \text{dom}(\Psi) \text{ and } (8) \\
\text{4 } & (\text{Convex lifting phase}) \text{ Decide } \alpha_t \in [\alpha_{\min}, \alpha_{\max}] \text{ and obtain the SVD } U_t, \Sigma_t, V_t \text{ of } X_{t+1} \text{ (without forming } X_{t+1} \text{ explicitly) through } (4) \text{ such that } (12) \text{ holds and either } (13) \text{ or } (14) \text{ is satisfied} \\
\text{5 } & W_{t+1} \leftarrow U_t \text{diag}(\sqrt{\Sigma_t}), H_{t+1} \leftarrow V_t \text{diag}(\sqrt{\Sigma_t}) \\
\text{6 } & \text{If a partial/inexact eigendecomposition is used, the number of eigenvalues to compute is updated to } k + k_{\text{add}} \text{ where } k_{\text{add}} \geq 0 \\
\text{7 } & (\text{Rank update}) \ k = \text{rank(diag}(\Sigma_t)).
\end{align*}
\]

4. Analysis

This section provides theoretical guarantees for Algorithm 1. In particular, we first give suitable conditions for \( \epsilon_t \) to guarantee global convergence, and then further obtain convergence
rates by imposing further requirements on \( \epsilon_t \). Next, rank identification of Algorithm 1 is proven under a nondegeneracy condition, which shows that for any subsequence \( \{X_{t_i}\}_{i=0}^{\infty} \) of the iterates that converge to a solution \( X^* \), \( \text{rank}(X_{t_i}) = \text{rank}(X^*) \) for all \( i \) large enough, so the rank \( k_t \) in (BM-nuclear) will eventually be automatically adjusted to the optimal value.

4.1 Global Convergence and Worst-case Rates

Our first main theoretical result is the global convergence of our algorithm. In our proofs, we let \( \Omega^* \) denote the solution set to (CVX), and \( F^* \) the optimal objective. For notational simplicity, we denote

\[
\text{dist}(X, \Omega^*) := \inf_{X^* \in \Omega^*} \|X - X^*\|_F.
\]

**Theorem 2** Consider (CVX) with \( \Psi \) defined in (1). Then \( \Omega^* \) is compact and nonempty, and \( F \) is coercive. If \( \nabla f \) is Lipschitz continuous and

\[
\sum_{t=0}^{\infty} \epsilon_t^2 < \infty
\]

in Algorithm 1 with the condition (13) being enforced, then for any initialization \( W_0, H_0 \), we always have \( \text{dist}(X_t, \Omega^*) \rightarrow 0 \), there is at least one limit point of \( \{X_t\} \), any such limit point is a global solution to (CVX), and \( F(X_t) \rightarrow F^* \). Moreover, the same results also apply to \( \{\tilde{X}_t\} \).

**Proof** The coerciveness of \( F \) follows directly from the fact that the nuclear norm is coercive and that \( f \) is lower-bounded. This implies that the level sets of \( F \) are bounded, and thus so is \( \Omega^* \). Moreover, because \( F \) is lower semicontinuous, it attains its minimum confined to any compact set, so we see that \( \Omega^* \) is nonempty.

From (15), we have

\[
\infty > c^2 := \sum_{t=0}^{\infty} \epsilon_t^2.
\]

We first show that \( \{X_t\} \) admits at least one limit point. From (12), we have that there exists \( G_t \in \mathbb{R}^{m \times n} \) such that

\[
G_t \in \nabla f \left( \tilde{X}_t \right) + \frac{1}{\alpha_t} \left( X_{t+1} - \tilde{X}_t \right) + \partial \Psi(X_{t+1}), \quad \|G_t\|_F \leq \epsilon_t.
\]

From the convexity of \( \Psi(\cdot) \), we have

\[
\Psi(X_{t+1}) \leq \langle \Xi, X_{t+1} - X \rangle + \Psi(X), \quad \forall \Xi \in \partial \Psi(X_{t+1}), \quad \forall X.
\]

By combining (18) with \( X = \tilde{X}_t \), (13) and (17), and defining \( \gamma := (1 - \delta)/\alpha_{\text{max}} \), we get the following inequality:

\[
F(X_{t+1}) \leq F(\tilde{X}_t) + \langle G_t, X_{t+1} - \tilde{X}_t \rangle - \frac{1 - \delta}{\alpha_t} \|X_{t+1} - \tilde{X}_t\|_F^2
\]

\[
\leq F(\tilde{X}_t) + \|G_t\|_F \|X_{t+1} - \tilde{X}_t\|_F - \frac{1 - \delta}{\alpha_{\text{max}}} \|X_{t+1} - \tilde{X}_t\|_F^2
\]

\[
\leq F(\tilde{X}_t) + \epsilon_t \|X_{t+1} - \tilde{X}_t\|_F - \gamma \|X_{t+1} - \tilde{X}_t\|_F^2
\]

\[
\leq F(X_t) + \epsilon_t \|X_{t+1} - \tilde{X}_t\|_F - \gamma \|X_{t+1} - \tilde{X}_t\|_F^2,
\]

(19)
where the last inequality is from (8). (20) implies that
\[
\gamma \|X_{t+1} - \tilde{X}_t\|_F^2 \leq F(X_t) - F(X_{t+1}) + \epsilon_t \|X_{t+1} - \tilde{X}_t\|_F.
\] (21)

By summing (21) from \(t = 0\) to \(t = k\), we have that
\[
\gamma \sum_{t=0}^{k} \|X_{t+1} - \tilde{X}_t\|_F^2 \leq F(X_0) - F(X_{k+1}) + \sum_{t=0}^{k} \epsilon_t \|X_{t+1} - \tilde{X}_t\|_F
\] (22)
\[
\leq F(X_0) - F^* + \sum_{t=0}^{k} \epsilon_t \|X_{t+1} - \tilde{X}_t\|_F
\]
\[
\leq F(X_0) - F^* + \sqrt{\sum_{t=0}^{k} \epsilon_t^2 \sum_{t=0}^{k} \|X_{t+1} - \tilde{X}_t\|_F^2},
\] (23)
\[
\leq F(X_0) - F^* + c \sqrt{\sum_{t=0}^{k} \|X_{t+1} - \tilde{X}_t\|_F^2},
\] (24)

where (23) is from the Cauchy-Schwarz inequality. By applying the quadratic formula to (24), we obtain that
\[
\sqrt{\sum_{t=0}^{k} \|X_{t+1} - \tilde{X}_t\|_F^2} \leq \frac{c + \sqrt{c^2 + 4\gamma (F(X_0) - F^*)}}{2\gamma}.
\] (25)

This implies that for all \(k \geq 0\), we get
\[
\sum_{t=0}^{k} \epsilon_t \|X_{t+1} - \tilde{X}_t\|_F \leq \sqrt{\sum_{t=0}^{k} \epsilon_t^2 \sum_{t=0}^{k} \|X_{t+1} - \tilde{X}_t\|_F^2} \leq \frac{c^2 + c \sqrt{c^2 + 4\gamma (F(X_0) - F^*)}}{2\gamma}.
\] (26)

Combining (26) and (22), we have that \(\{F(X_t)\}\) is upper-bounded. Then from the coerciveness of \(F\), the sequence \(\{X_t\}\) is bounded, and thus it has at least one limit point.

Next, we prove that \(\text{dist}(X_t, \Omega^*) \to 0\) by contradiction. Suppose this statement is false, then there exists \(\sigma > 0\) and a subsequence \(\{X_{t_k}\}\) such that \(\text{dist}(X_{t_k}, \Omega) \geq \sigma > 0\) for any \(k\). Since \(\{X_{t_k}\}\) is also a bounded sequence, we have that there exists a subsequence \(\{X_{t_{k'}}\} \subseteq \{X_{t_k}\}\) such that
\[
X_{t_{k'}} \to \bar{X}^* \notin \Omega^*.
\] (27)

From (17), we have that
\[
G_t + \nabla f(X_{t+1}) - \nabla f(\tilde{X}_t) - \frac{1}{\alpha_t} (X_{t+1} - \tilde{X}_t) \in \partial F(X_{t+1}).
\] (28)

From (15), we have \(\epsilon_t \to 0\) and hence \(G_t \to 0\). From (25), we have that \(X_{t+1} - \tilde{X}_t \to 0\). This together with the Lipschitz continuity of \(\nabla f\) and the boundedness of \(\alpha_t\) implies that \(\nabla f(X_{t+1}) - \nabla f(\tilde{X}_t) \to 0\) and \(\alpha_t^{-1} (X_{t+1} - \tilde{X}_t) \to 0\). These results together imply that
\[
G_t + \nabla f(X_{t+1}) - \nabla f(\tilde{X}_t) - \frac{1}{\alpha_t} (X_{t+1} - \tilde{X}_t) \to 0.
\] (29)

From (17), we have that
\[
G_t + \nabla f(X_{t+1}) - \nabla f(\tilde{X}_t) - \frac{1}{\alpha_t} (X_{t+1} - \tilde{X}_t) \to 0.
\] (29)
From (27)–(29) and the outer semi-continuity of \( \partial F \) (see Rockafellar and Wets, 2009, Proposition 8.7) and (Bauschke and Combettes, 2017, Proposition 20.37) in (28), we have that \( 0 \in \partial F(\tilde{X}^*) \). From the convexity of \( F \), we thus get \( \tilde{X}^* \in \Omega^* \), contradicting (27). Therefore, we conclude \( \text{dist}(X_t, \Omega^*) \to 0 \). This also implies that any limit point of \( \{X_t\} \) must lie in \( \Omega^* \).

Finally, we prove the convergence of \( \{F(X_t)\} \). From the convexity of \( F \) and (28), we see that for any \( X^* \in \Omega^* \),

\[
0 \leq F(X_{t+1}) - F(X^*) \\
\leq (G_t + \nabla f(X_{t+1}) - \nabla f(\tilde{X}_t) - \alpha_t^{-1}(X_{t+1} - \tilde{X}_t), X_{t+1} - X^*) \\
\leq \left\| G_t + \nabla f(X_{t+1}) - \nabla f(\tilde{X}_t) - \alpha_t^{-1}(X_{t+1} - \tilde{X}_t) \right\|_F \|X_{t+1} - X^*\|_F. \tag{30}
\]

By the boundedness of \( \{X_t\} \), \( \|X_{t+1} - X^*\|_F \) is upper bounded. Recall that the first norm term in (30) approaches to 0 as shown in (29). Thus, \( F(X_t) \to F^* \) by the sandwich lemma.

For the part of \( \{\tilde{X}_t\} \), we see that the boundedness of the iterates and the convergence of the objective value follow from (8) and again the coerciveness of \( F \). From this boundedness we then conclude the existence of a limit point, and convergence of \( \text{dist}(\tilde{X}_t, \Omega^*) \) follows from the same argument above.

Although our global convergence is guaranteed by the inexact PG step, existing analyses for the inexact PG method, like those by Combettes (2004); Schmidt et al. (2011); Jiang et al. (2012); Hamadouche et al. (2022), utilize the geometry of the iterates, and are hence not applicable to our algorithm because our additional BM phase could move the iterates arbitrarily within the level sets and this may make such geometry properties no longer valid. Therefore, another contribution of this work is in developing new proof techniques for obtaining global convergence guarantees for alternating between general nonmonotone inexact PG steps and some other descent optimization steps.

We next provide convergence rate guarantees for Algorithm 1 in the coming two theorems. For such results, we use the definitions below.

\[
\Delta F_t := F(X_t) - F^*, \quad \Delta \tilde{F}_t := F(\tilde{X}_t) - F^*.
\]

The following theorem and its proof are partially motivated by Scheinberg and Tang (2016).

**Theorem 3** Suppose the conditions in Theorem 2 hold. Then there exists a constant \( \beta > 0 \) such that the following inequality holds:

\[
\Delta \tilde{F}_{t+1} \leq \max \left\{ \xi_t, \frac{\Delta \tilde{F}_t}{1 + \beta^{-1}\Delta F_t} \right\}, \quad \xi_t := \sqrt{\beta \psi_t + \psi_t}, \quad \psi_t := \epsilon_t(\|X_{t+1} - \tilde{X}_t\|_F + \gamma \epsilon_t), \tag{31}
\]

where \( \gamma := (1 - \delta)/\alpha_{\text{max}} \). Moreover, if \( \epsilon_t = O(t^{-2}) \), then \( \Delta F_t = O(t^{-1}) \) and \( \Delta \tilde{F}_t = O(t^{-1}) \).

**Proof** From the boundedness of \( \{F(X_t)\} \) and \( \{F(\tilde{X}_t)\} \) obtained in the proof of Theorem 2, we know that there is a value \( F_0 \) such that \( F(X_t) \leq F_0 \) and \( F(\tilde{X}_t) \leq F_0 \) for all \( t \), and thus
from the coerciveness of $F$ from Theorem 2, there is a nonnegative and finite constant $R_0$ such that
\[ R_0 := \max_{X_1, X_2 \in \{X \mid F(X) \leq F_0\}} \|X_1 - X_2\|_F < \infty. \] (32)

Next, from the convexity of $f$, we have that for any $X \in \mathbb{R}^{m \times n}$,
\[ f(\tilde{X}_t) \leq f(X) + \langle \nabla f(\tilde{X}_t), \tilde{X}_t - X \rangle. \] (33)

Add up (13) and (33), we get
\[ f(X_{t+1}) \leq f(X) + \frac{\delta}{\alpha_t} \|X_{t+1} - \tilde{X}_t\|_F^2. \] (34)

Add up (18) and (34), we get
\[ F(X_{t+1}) \leq F(X) + \frac{1}{\alpha_t} \langle \tilde{X}_t - X_{t+1}, X_{t+1} - X \rangle + \frac{\delta}{\alpha_t} \|X_{t+1} - \tilde{X}_t\|_F^2 + \langle G_t, X_{t+1} - X \rangle \] (35)
for $G_t$ defined in (17). Choose $X = X^*$ for some $X^* \in \Omega^*$ in (35) and use (32), we get
\[ F(X_{t+1}) \leq F(X^*) + \frac{R_0}{\alpha_t} \|\tilde{X}_t - X_{t+1}\|_F + \frac{\delta R_0}{\alpha_t} \|\tilde{X}_t - X_{t+1}\|_F + \epsilon_t R_0 \]
\[ \leq F(X^*) + \tilde{c}(\|X_{t+1} - \tilde{X}_t\|_F + \epsilon_t), \] (36)
where $\tilde{c} := R_0(1 + \delta + \alpha_{\min})/\alpha_{\min} \in [0, \infty)$ is a constant. Note that $F$ is Lipschitz continuous in any bounded region, so we can further obtain from (36) that
\[ F(\tilde{X}_t) \leq F(X^*) + \tilde{c}(\|X_{t+1} - \tilde{X}_t\|_F + \epsilon_t), \] (37)
where $\tilde{c} > 0$ is a constant. From (37), we further get that
\[ (F(\tilde{X}_t) - F(X^*))^2 \leq 2\tilde{c}^2 \|X_{t+1} - \tilde{X}_t\|_F^2 + 2\epsilon_t^2, \] (38)
Substitute (38) into (19) and use (31), we get
\[ F(\tilde{X}_{t+1}) \leq F(X_{t+1}) \leq F(\tilde{X}_t) + \psi_t - \frac{\gamma}{2\tilde{c}^2} (F(\tilde{X}_t) - F(X^*))^2, \] (39)
Let $\beta = 4\tilde{c}^2/\gamma$, we then have the following two cases from (39):

**Case 1.** $\psi_t > \beta^{-1} \Delta F_t^2$.
We have that $\Delta \tilde{F}_t \leq \sqrt{\beta} \psi_t$. Combine this and (39), we get
\[ \Delta \tilde{F}_{t+1} \leq \Delta F_{t+1} \leq \sqrt{\beta} \psi_t + \psi_t. \] (40)

**Case 2.** $\psi_t \leq \beta^{-1} \Delta \tilde{F}_t^2$.
Substitute this into (39), we get
\[ \Delta \tilde{F}_{t+1} \leq \Delta F_{t+1} \leq \Delta \tilde{F}_t - \frac{1}{\beta} \Delta \tilde{F}_t^2 \leq \Delta \tilde{F}_t. \] (41)
From (41), we have that
\[ \Delta\tilde{F}_{t+1} \leq \Delta\tilde{F}_t - \beta^{-1}\Delta\tilde{F}_t \Delta\tilde{F}_{t+1}, \]
which implies that
\[ \Delta\tilde{F}_{t+1} \leq \frac{\Delta\tilde{F}_t}{1 + \beta^{-1}\Delta\tilde{F}_t}. \tag{42} \]
Combine (40) and (42), we get (31).

Now, assume that \( \epsilon_t = O\left(t^{-2}\right) \). From (31) and (32), we see that \( \xi_t = O\left(t^{-1}\right) \). Namely, there exists \( \kappa \geq 0 \) such that \( \xi_t \leq \kappa/t \) for all \( t \geq 1 \). For any \( t \in \mathbb{N} \), we first consider the case in which there is some index \( t_0 \in \{1, \ldots, t\} \) such that the first term in (31) is larger than the second one and let \( t_0 \) be the maximum of such indices. Thus, for any \( k \in \{t_0 + 1, \ldots, t\} \), we have that \( \Delta\tilde{F}_{k+1} \leq \Delta\tilde{F}_k / (1 + \beta^{-1}\Delta\tilde{F}_k) \), which implies that
\[ \frac{1}{\Delta F_k} + \frac{1}{\beta} \leq \frac{1}{\Delta F_{k+1}}. \]
Summing the inequality above from \( k = t_0 + 1 \) to \( k = t \) and telescoping, we have that
\[ \frac{1}{\Delta F_{t+1}} \geq \frac{1}{\Delta F_{t_0+1}} + \frac{(t - t_0)}{\beta} \geq \frac{t_0}{\kappa} + \frac{t - t_0}{\beta} \geq \frac{t}{\max\{\kappa, \beta\}}, \tag{43} \]
which implies
\[ \Delta\tilde{F}_{t+1} \leq \frac{\max\{\beta, \kappa\}}{t} = O(t^{-1}). \tag{44} \]
Now let us turn to the case in which the second term in (31) is larger than the first one for all \( k \in \{1, \ldots, t\} \). Then an analysis analogous to (43) leads to the following inequality:
\[ \Delta\tilde{F}_{t+1} \leq \frac{1}{\Delta F_{t+1}} + \beta^{-1}t = O(t^{-1}). \tag{45} \]
Combine (44) and (45), we obtain that \( \Delta\tilde{F}_t = O(t^{-1}) \). Finally, viewing from (39) and the optimality of \( F^* \), we get from (44), (45), and \( \psi_t = O(t^{-1}) \) that \( \Delta F_{t+1} \leq \Delta \tilde{F}_t + \psi_t = O(t^{-1}) \).

In the next result, we show faster convergence rates under a Hölderian error-bound condition
\[ \zeta \text{dist}(X, \Omega^*) \leq (F(X) - F^*)^\theta, \quad \forall X \tag{46} \]
for some \( \zeta > 0 \) and some \( \theta \in [0, 1] \). In particular, when \( \theta \geq 1/2 \), we obtain linear convergence for the objective. Under convexity of \( F \), it is shown by Bolte et al. (2017) that (46) is equivalent to the Kurdyka-Lojasiewicz (KL) condition (Kurdyka, 1998; Lojasiewicz, 1963).

**Theorem 4** Consider (CVX) with \( \Psi \) defined in (1). Suppose the line-search criterion (14) is used with \( \delta \in (0, 1) \) in Algorithm 1. If \( F \) satisfies (46), then the following convergence results hold.

(i) When \( \theta = 1/2 \): Let
\[ M := \min_{\mu \in [0, 1]} 1 - \delta \mu + \frac{\delta \mu^2}{2\zeta^2 \sigma_{\min}} < 1. \tag{47} \]
If
\[ \sum_{t=1}^{\infty} \epsilon_t^2 / M_t < \infty, \] (48)

then \( \Delta \tilde{F}_t = O(M'). \)

(ii) When \( 0 \leq \theta < 1/2: \) If
\[ \epsilon_t^2 = O\left(t^{-\frac{2-2\theta}{1-2\theta}}\right), \] (49)

then
\[ \Delta \tilde{F}_t = O\left(t^{-\frac{1}{1-2\theta}}\right). \]

(iii) When \( \theta > 1/2: \) If we use (13) instead of (14) and \( \epsilon_t^2 \) is summable, then there is \( T_0 \geq 0 \) such that \( X_{T_0} \in \Omega^*. \)

Proof Let \( X_t^* \) and \( Q_t^* \) be the unique minimizer and minimum of \( Q_t^{\alpha t}(Y) \) respectively. Because \( Q_t^{\alpha t}(Y) \) is a strongly-convex function with modulus \( \alpha_t^{-1} \), from (12), there exist \( G_t \) such that
\[ \frac{1}{\alpha_t} \|X_{t+1} - X_t^*\|_F \leq \|G_t\|_F \leq \epsilon_t. \] (50)

From (14), we have that
\[ F(X_{t+1}) - F(\tilde{X}_t) \leq \delta Q_t^{\alpha t}(X_{t+1}) \]
\[ \leq \delta (Q_t^* + (G_t, X_{t+1} - X_t^*)) \]
\[ \leq \delta (Q_t^* + \|G_t\|_F \|X_{t+1} - X_t^*\|_F) \]
\[ \leq \delta (Q_t^* + \alpha_t \epsilon_t^2), \] (51)

where the second inequality comes from the convexity of \( Q_t^{\alpha t}(\cdot) \), and the last one comes from (50). From Lemma 5 in Lee and Wright (2019) (also see equation (70) of Lee (2023)), we have that
\[ Q_t^* \leq \mu(F^* - F(\tilde{X}_t)) + \frac{\mu^2}{2\alpha_t} \text{dist}(\tilde{X}_t, \Omega^*)^2, \quad \forall \mu \in [0, 1]. \] (52)

Substitute (52) into (51) and use (46), we get
\[ F(\tilde{X}_{t+1}) - F(\tilde{X}_t) \leq F(X_{t+1}) - F(\tilde{X}_t) \]
\[ \leq \delta \min_{\mu \in [0, 1]} \left( \mu(F^* - F(\tilde{X}_t)) + \frac{\mu^2}{2\alpha_t} \text{dist}(\tilde{X}_t, \Omega^*)^2 + \alpha_t \epsilon_t^2 \right) \]
\[ \leq \delta \min_{\mu \in [0, 1]} \left( \mu(F^* - F(\tilde{X}_t)) + \frac{\mu^2}{2\zeta^2\alpha_t} (F(\tilde{X}_t) - F^* )^{2\theta} + \alpha_t \epsilon_t^2 \right). \] (53)

Proof of (i). Let
\[ M_t := \min_{\mu \in [0, 1]} 1 - \delta \mu + \frac{\delta \mu^2}{2\zeta^2\alpha_t}, \quad \forall t \geq 0. \]

Because \( 0 < \delta < 1 \) and \( \alpha_t \leq \alpha_{\text{max}} \), we have that \( 0 < M_t < M < 1 \) for all \( t \geq 0 \). From (53) and the assumption that \( \theta = 1/2 \), we get
\[ \Delta \tilde{F}_{t+1} / M_t^{t+1} \leq \Delta \tilde{F}_{t} / M_t + \delta \alpha_t \epsilon_t^2 / M_t^{t+1}. \] (54)
Because $\epsilon_t^2/M^t$ is summable from (48) and $\alpha_t \leq \alpha_{\text{max}} < \infty$, (54) clearly shows $\Delta \tilde{F}_t = O(M^t)$.

**Proof of (ii).** From (53), we have

$$\Delta \tilde{F}_{t+1} \leq \min_{\mu \in [0,1]} \left\{ 1 - \delta \mu + \frac{\delta \mu^2}{2\zeta^2\alpha_t} \right\} \Delta \tilde{F}_t + \delta \alpha_t \epsilon_t^2. \quad (55)$$

Clearly, the minimizer of $\mu$ in (55) is $\min\{\zeta^2 \alpha_t \Delta \tilde{F}_t^{1-2\theta}, 1\}$. Substitute this into (55), we get

$$\Delta \tilde{F}_{t+1} \leq \left( 1 - \frac{\delta \zeta^2 \alpha_t}{2} \min \left\{ \frac{\Delta \tilde{F}_t^{1-2\theta}}{\alpha_t}, \frac{1}{\zeta^2 \alpha_t} \right\} \right) \Delta \tilde{F}_t + \delta \alpha_t \epsilon_t^2. \quad (56)$$

We first show that $\lim_{t \to \infty} \Delta \tilde{F}_t = 0$. Assume on the contrary that there exists $\eta > 0$ such that $\Delta \tilde{F}_t \geq \eta$ for all $t \geq 0$. Then, from (56), we have that

$$\Delta \tilde{F}_{t+1} \leq \left( 1 - \frac{\delta \zeta^2 \alpha_t}{2} \min \left\{ \frac{\eta^{1-2\theta}}{\alpha_t}, \frac{1}{\zeta^2 \alpha_t} \right\} + \frac{\delta \alpha_t \epsilon_t^2}{\eta} \right) \Delta \tilde{F}_t,$$

which implies that $\Delta \tilde{F}_t \to 0$ with a linear rate when $t$ is sufficiently large to make $\epsilon_t$ small enough. This contradicts to $\Delta \tilde{F}_t \geq \eta > 0$. Now, since $\lim_{t \to \infty} \Delta \tilde{F}_t = 0$ and $\sum_{t=1}^{\infty} \alpha_t \epsilon_t^2 < \infty$, there exists $T_0 \geq 0$ such that $\Delta \tilde{F}_{T_0} \leq C_1$ and $\sum_{t=T_0}^{\infty} \delta \alpha_t \epsilon_t^2 < C_1$, where

$$C_1 := \frac{1}{2}(\zeta^2 \alpha_{\text{max}})^{-\frac{1}{1-2\theta}}.$$

From (56), we thus get

$$\Delta \tilde{F}_t \leq \Delta \tilde{F}_{T_0} + \sum_{k=T_0}^{t-1} \delta \alpha_t \epsilon_t^2 < 2C_1 \leq (\zeta^2 \alpha_t)^{-\frac{1}{1-2\theta}}, \quad \forall t \geq T_0. \quad (57)$$

Thus,

$$\min \left\{ \frac{\Delta \tilde{F}_t^{1-2\theta}}{\alpha_t}, \frac{1}{\zeta^2 \alpha_t} \right\} = \frac{\Delta \tilde{F}_t^{1-2\theta}}{\alpha_t}, \quad \forall t \geq T_0.$$

Let $M_1 := \delta \zeta^2 \alpha_{\text{min}}/2$, $M_2 := \delta \alpha_{\text{max}}$, then from the equation above and (56), we have that

$$\Delta \tilde{F}_{t+1} \leq (1 - M_1) \Delta \tilde{F}_t + M_2 \epsilon_t^2, \quad \forall t \geq T_0. \quad (58)$$

From (57) and that $0 < \delta < 1$, we have that $M_1 \Delta \tilde{F}_t^{1-2\theta} < 1$ for any $t \geq T_0$. Now we choose $D > 0$ to be a sufficiently large number such that the following three conditions hold.

$$\Delta \tilde{F}_{T_0} \leq DT_0^{\frac{1}{1-2\theta}}, \quad (59a)$$

$$M_2 \epsilon_t^2 + \left( \frac{2M_2}{M_1} \right)^{\frac{1}{1-2\theta}} \leq D(t+1)^{\frac{1}{1-2\theta}}, \quad \forall t \geq T_0, \quad (59b)$$

$$D^{-1} \leq \left( 1 - 2\theta \right) M_1 \leq \frac{2}{2}, \quad (59c)$$

where (59b) is guaranteed by (49) and $\theta < 1/2$, and (59c) can be guaranteed by $\theta < 1/2$.

Now, we use mathematical induction to prove that $\Delta \tilde{F}_t \leq DT_0^{\frac{1}{1-2\theta}}$ for all $t \geq T_0$. The case

}\]
\( t = T_0 \) directly comes from (59a). Suppose the inequality holds for some \( t \geq T_0 \), we have the following two cases.

**Case 1.** \( M_2 \epsilon_t^2 \leq M_1 \tilde{F}_t^{2-2\theta}/2 \).

From (58), we have that
\[
\Delta \tilde{F}_{t+1} \leq \left( 1 - \frac{M_1}{2} \Delta \tilde{F}_t^{1-2\theta} \right) \Delta \tilde{F}_t,
\]
which leads to
\[
\Delta \tilde{F}_{t+1}^{-(1-2\theta)} \geq \left( 1 - \frac{M_1}{2} \Delta \tilde{F}_t^{1-2\theta} \right)^{-(1-2\theta)} \Delta \tilde{F}_t^{-(1-2\theta)}
\]
\[
\geq \left( 1 + \frac{(1-2\theta)M_1}{2} \Delta \tilde{F}_t^{1-2\theta} \right) \Delta \tilde{F}_t^{-(1-2\theta)}
\]
\[
= \Delta \tilde{F}_t^{-(1-2\theta)} + \frac{(1-2\theta)M_1}{2}
\]
\[
\geq D^{-1-2\theta} t + \frac{(1-2\theta)M_1}{2}
\]
\[
\geq D^{-1-2\theta} (t + 1),
\]
where the second inequality comes from the fact that \((1-x)^{-p} \geq 1 + px\) for any \( x < 1 \) and \( p > 0 \), and the last inequality comes from (59c). (60) implies that \( \Delta \tilde{F}_{t+1} \leq D(t+1)^{1-2\theta} \).

**Case 2.** \( M_2 \epsilon_t^2 > M_1 \tilde{F}_t^{2-2\theta}/2 \).

In this case, we have that
\[
\Delta \tilde{F}_t \leq \left( \frac{2M_2}{M_1} \epsilon_t^2 \right)^{\frac{1}{2-2\theta}}.
\]
By substituting (61) into (58), we obtain
\[
\Delta \tilde{F}_{t+1} \leq \Delta \tilde{F}_t + M_2 \epsilon_t^2 \leq \left( \frac{2M_2}{M_1} \epsilon_t^2 \right)^{\frac{1}{2-2\theta}} + M_2 \epsilon_t^2 \leq D(t+1)^{1-2\theta},
\]
where the last inequality comes from (59b).

Combining Cases 1 and 2, we get \( \Delta \tilde{F}_t \leq Dt^{1-2\theta} \) for any \( t \geq T_0 \), as desired.

**Proof of (iii).** From Theorem 2, we get that \( \text{dist}(X_t, \Omega^*) \to 0 \), and therefore (Yue et al., 2019, Proposition 1) implies \( \|X_t - \text{prox}_\Psi(X_t - \nabla f(X_t))\|_F \to 0 \). On the other hand, from (Bolte et al., 2017, Theorem 5) and (Mordukhovich et al., 2022, Proposition 2.4), the condition (46) together with the convexity of \( F \) implies that there is \( \kappa \geq 0 \) such that
\[
\text{dist}(X, \Omega^*) \leq \kappa \|X - \text{prox}_\Psi(X - \nabla f(X))\|_F^{\frac{\theta}{1-\theta}}.
\]
Moreover, as \( \theta > 1/2 \), we get \( \theta/(1-\theta) > 1 \). Therefore, we can apply (Lee and Wright, 2022, Theorem 3) to obtain the desired conclusion to complete the proof.

By setting \( \theta = 0 \) in Theorem 4, we recover the same convergence rate in Theorem 3, but instead of \( \epsilon_t = O(t^{-2}) \) in Theorem 3, Theorem 4 only needs \( \epsilon_t = O(t^{-1}) \). The difference between the two theorems is that Theorem 3 uses (13) that allows a more aggressive step.
size selection but with the price of a higher accuracy in the PG step, while Theorem 4 uses (14) that leads to a more conservative step size to trade for less time spent on computing the approximate SVD. Moreover, for Theorem 4, (46) with \( \theta = 0 \) directly assumes that the iterates are bounded.

### 4.2 Rank identification

We proceed on to show that under a nondegeneracy condition, the rank of \( X_t \) for any convergent subsequence will eventually become fixed and equivalent to the point of convergence. First, we need the definition below of convex partly smooth functions. This definition involves the usage of \( C^2 \)-manifold, which means the system of equations defining such a manifold is \( C^2 \).

**Definition 5 (Partly smooth (Lewis, 2002))** A convex function \( \Psi \) is partly smooth at a point \( X^* \) relative to a set \( M \) containing \( X^* \) if \( \partial \Psi(X^*) \neq \emptyset \) and:

1. Around \( X^* \), \( M \) is a \( C^2 \)-manifold and \( \Psi|_M \) is \( C^2 \).
2. The affine span of \( \partial \Psi(X) \) is a translate of the normal space to \( M \) at \( X^* \).
3. \( \partial \Psi \) is continuous at \( X^* \) relative to \( M \).

Loosely speaking, this means that \( \Psi|_M \) is smooth at \( x^* \), but the value of \( \Psi \) changes drastically along directions leaving \( M \) around \( x^* \).

It is known (Daniilidis et al., 2014) that at every \( X \in \mathbb{R}^{m \times n} \), \( \| \cdot \|_* \) is partly smooth with respect to the manifold

\[
\mathcal{M}(X) := \{ Y \in \mathbb{R}^{m \times n} \mid \text{rank}(Y) = \text{rank}(X) \}. \tag{63}
\]

Similarly, if \( X \in \mathbb{S}^n \), we also have that \( \delta_{\mathbb{S}^n} \) is partly smooth everywhere in \( \mathbb{S}^n_+ \), with respect to the manifold

\[
\mathcal{M}_2(X) := \{ Y \in \mathbb{S}^n_+ \mid \text{rank}(Y) = \text{rank}(X) \}. \tag{64}
\]

Finally, when \( S \) is a polyhedron, it is widely known that \( \delta_S \) is also partly smooth everywhere, with respect to the minimal face containing the reference point. As intersections of manifolds are still manifolds, if \( \Psi(X) = \lambda \| X \|_* + \lambda_2 \delta_{\mathbb{S}^n_+}(X) + \lambda_3 \delta_S(X) \) for \( \lambda \in \mathbb{R}, \lambda_2, \lambda_3 \in \{0, 1\} \) and some polyhedral \( S \), we have that \( \Psi \) is partly smooth everywhere, with respect to a submanifold \( \mathcal{M}(X) \subseteq \mathcal{M}(X) \).

Now we can leverage tools from partial smoothness and manifold identification to show that our algorithm will find the correct rank for (BM) that contains a global optimum.

**Theorem 6** Consider (CVX) with \( \Psi \) convex, proper, and closed. Consider the two sequences of iterates \( \{X_t\} \) and \( \{\tilde{X}_t\} \) generated by Algorithm 1 from some starting point \( X_0 = (W_0, H_0) \) with \( \epsilon_t \to 0 \) in (12). Then the following hold.

(i) For any subsequence \( \{\tilde{X}_{t_i}\} \) such that \( \tilde{X}_{t_i} \to X^* \) for some \( X^* \in \Omega^* \), \( X_{t+1} \to X^* \) as well.

(ii) For the same subsequence as above, if \( X^* \) satisfies the nondegeneracy condition

\[
0 \in \text{relint} (\partial F(X^*)) \tag{65}
\]
and $\Psi$ is as defined in (1), then there is $i_0 \geq 0$ such that $\text{rank}(X_{t,i+1}) = \text{rank}(X^*)$ for all $i \geq i_0$.

**Proof**

**Proof of (i).** Let us denote the exact solution of (9) at the $t_i$-th iteration given $\tilde{X}_{t_i}$ as $X^*_{t_i+1}$, and the real update we use from (12) as $X_{t_i+1}$. Following the proof of (Yue et al., 2019, Proposition 1), we have from the optimality of $X^*$, which implies $\text{prox}_{\alpha \psi}(X^* - \alpha \nabla f(X^*)) = X^*$ for any $\alpha > 0$, that

$$
\left\| X^*_{t,i} - \tilde{X}_{t,i} \right\|_F = \left\| \text{prox}_{\alpha \psi} \left( \tilde{X}_{t_i} - \alpha \nabla f(\tilde{X}_{t_i}) \right) - \tilde{X}_{t_i} + \left( X^* - \text{prox}_{\alpha \psi}(X^* - \alpha \nabla f(X^*)) \right) \right\|_F
\leq \left\| \tilde{X}_{t_i} - X^* \right\|_F + \left\| \text{prox}_{\alpha \psi} \left( \tilde{X}_{t_i} - \alpha \nabla f(\tilde{X}_{t_i}) \right) - \text{prox}_{\alpha \psi}(X^* - \alpha \nabla f(X^*)) \right\|_F
\leq \left\| \tilde{X}_{t_i} - X^* \right\|_F + \left\| \tilde{X}_{t_i} - \alpha \nabla f(\tilde{X}_{t_i}) - (X^* - \alpha \nabla f(X^*)) \right\|_F 
\leq 2 \left\| \tilde{X}_{t_i} - X^* \right\|_F + \alpha \| \nabla f(\tilde{X}_{t_i}) - \nabla f(X^*) \|_F
\leq (2 + L \alpha_{\max}) \left\| \tilde{X}_{t_i} - X^* \right\|_F \to 0,
$$

where (66) is from the nonexpansiveness of the proximal operation of any convex function, and (67) is from our assumption. (67) then leads to

$$
\left\| X^*_{t,i} - \tilde{X}_{t,i} \right\|_F \to 0.
$$

On the other hand, (50) shows that

$$
0 \leq \| X_{t,i+1} - X^*_{t,i} \|_F \leq \alpha \epsilon_{t,i} \to 0,
$$

where the limit is obtained from that $\epsilon_t \to 0$ and that $\alpha_t$ is upper-bounded. By combining (68) and (69), it is clear that

$$
\| X_{t,i+1} - \tilde{X}_{t,i} \| \to 0 \Rightarrow \| X_{t,i+1} - X^* \| \to 0,
$$

proving the desired result.

**Proof of (ii).** From our arguments preceding the theorem, $\Psi$ is partly smooth at every $X$ relative to a submanifold of either (63) or (64). Therefore, the result of the second item is equivalent to $X_{t,i+1} \in \mathcal{M}(X^*)$ for all $i$ large enough. As $\epsilon_t \to 0$, we see that all conditions of (Lee, 2023, Theorem 1) are satisfied, and therefore $X_{t,i+1} \in \mathcal{M}(X^*) \subseteq \mathcal{M}(X^*)$ for all $i$ large enough. The conclusion therefore follows.

Due to the flexibility in the BM step, we have less control over the iterates than ordinary PG methods. Therefore, convergence of the whole sequence of iterates cannot be directly guaranteed and we can only get subsequential convergence. However, in our experiments in Section 6, we often observe empirically that the iterates are convergent to a point, and the rank always becomes fixed after a few iterations of BM-Global.
5. Applications

We provide two applications of (CVX). One is our motivating example of matrix completion with \( X \) being the whole space, and the other one is a special class of convex quadratic semidefinite programming problems.

5.1 Matrix Completion

Our first application of (CVX) is the low-rank matrix completion problem (MC). This problem is widely seen in many machine learning tasks like recommendation systems, localization in Internet of Things (IoTs), and image denoising and compression. Interested readers are referred to a recent survey Nguyen et al. (2019) for more details of these applications.

We observe that the loss term of (MC) has an \( L \)-Lipschitz continuous gradient with \( L = 1 \), and when \( \alpha_t = L^{-1} = 1 \), \( X_t - \alpha_t \nabla f(X_t) \) is the same as replacing the entries of \( X_t \) in \( \Omega \) with \( P_\Omega(A) \), hence standard PG with \( \alpha_t \equiv L^{-1} \) is also called soft impute for this problem (Mazumder et al., 2010). Often in real applications described above, we can easily have \( m \) and \( n \) in the scale of millions with \( |\Omega| \) rather small, so indeed we are unable to explicitly form \( X_t \) and need to rely on low-rank assumptions or to force low-rank approximations for practical reasons. On the other hand, thanks to the extreme popularity and the simple forms of (MC) and (MF), there are many well-developed algorithms for them.

Theoretical analyses for (MC) and (MF) often consider the noiseless case such that the ground truth \( A \) is indeed of low rank and we observe entries without any noise, and show that under such cases, one can recover the whole \( A \) by solving (MF) with a sufficient rank. However, in practice, the observed entries are often noisy, either due to measurement errors (like in the IoTs case) or randomness in nature (rating in recommendation systems could be affected by factors beyond the users’ preference for certain items). We will see in the numerical experiments in Section 6 that it is often the case that solving (MF) alone does not guarantee convergence to a global optimum even if the correct rank is specified, and therefore the convex lifting step in Algorithm 1 is necessary.

For this problem, in the convex lifting step, we adopt a long-step variant of PG by setting \( \alpha_t \) close to \( 2L^{-1} \) to obtain a slightly better empirical performance. For the BM stage, we adopt the state-of-the-art solver polyMF-SS for (MF) developed by Wang et al. (2017) that conducts block coordinate descent with an exact line search, where each block is one column of \( W \) and one column of \( H \).

More implementation details of our algorithm tailored for this application are described in the supplementary materials.

5.2 A class of convex quadratic semidefinite programming problems

Our second application is the following convex quadratic semidefinite programming (QSDP) problem:

\[
\min_{X \in \mathbb{S}_+^n} \left( f(X) := \frac{1}{2} \| A(X) - b \|^2 + \langle C, X \rangle \right) \quad \text{s.t.} \quad \langle E, X \rangle = 0, \quad (QSDP)
\]
where $A : S^n \to \mathbb{R}^p$ is a linear mapping whose adjoint mapping is denoted by $A^*$, $b \in \mathbb{R}^p$, $C \in S^n$, and $E \in S^n$ denotes the matrix of all ones. Note that the gradient of $f$ is given by

$$\nabla f(X) = A^* (A(X) - b) + C.$$ 

Thus $\nabla f(\cdot)$ is Lipschitz continuous with modulus $L = \|A^* A\|_2$.

The QSDP problem (QSDP) arises in many important applications when one needs to find a low-rank approximation of a given matrix while preserving certain useful structures (via linear constraints). In this part, we introduce the following two data analysis problems.

• The regularized kernel estimation (RKE) problem (Lu et al., 2005): Given a set of $n$ objects and dissimilarity measures $d_{ij}^2$ for certain object pairs $(i, j) \in \Omega$, the goal of RKE is to find a positive semidefinite matrix $X$ such that the fitted squared distances between the objects induced by $X$ satisfy

$$X_{ii} + X_{jj} - 2X_{ij} \approx d_{ij}^2, \quad \forall (i, j) \in \Omega.$$ 

To obtain a low-rank solution for $X$, the following regularized semidefinite least squares problem is often considered:

$$\min_{X \in S^n_+} \frac{1}{2} \sum_{(i,j) \in \Omega} w_{ij} (X_{ii} + X_{jj} - 2X_{ij} - d_{ij}^2)^2 + \lambda \langle I, X \rangle \quad \text{s.t.} \quad \langle E, X \rangle = 0,$$ 

(71)

where $\lambda > 0$ is a positive regularization parameter and $w_{ij} > 0$ for any $(i, j) \in \Omega$. In the above, the constraint $\langle E, X \rangle = 0$ is a normalization to put the center of mass of the realized Euclidean embedding at the origin. As argued in Section 2, $\langle I, X \rangle$ is equivalent to the nuclear norm for $X \in S^n_+$, so (71) induces low-rank solutions.

• The molecular conformation problem (Fang and Toh, 2013): Given a molecule with $n$ atoms and the estimated inter-atomic distances $d_{ij}$ between some pairs $(i, j) \in \Omega$ of atoms, the goal is to determine the positions $x_1, \ldots, x_n \in \mathbb{R}^d$ of all the atoms. Mathematically, the molecular conformation problem can be stated as follows:

$$\min_{x_i \in \mathbb{R}^d, 1 \leq i \leq n} \frac{1}{2} \sum_{(i,j) \in \Omega} w_{ij} \left( \|x_i - x_j\|^2 - d_{ij}^2 \right)^2 - \frac{\rho}{2n} \sum_{i,j=1}^n \|x_i - x_j\|^2 \quad \text{s.t.} \quad \sum_{i=1}^n x_i = 0,$$

where $w_{ij} > 0$ for all $(i, j) \in \Omega$ and the second term involving $\rho > 0$ is used to maximize the pairwise separations between atoms. Define the matrix

$$X := \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix}^\top \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \in S^n,$$ 

then, it is easy to check that

$$\|x_i - x_j\|^2 = X_{ii} + X_{jj} - 2X_{ij}, \quad \sum_{i,j=1}^n \|x_i - x_j\|^2 = 2n \langle I, X \rangle,$$

and the constraint $\sum_{i=1}^n x_i = 0$ can be replaced by $\langle E, X \rangle = 0$. We therefore get the same QSDP relaxation (71) for the molecular conformation problem with $\lambda := -\rho < 0$. 

22
Although $\lambda < 0$ in this application, the problem is still in the form (CVX) with a regularizer in (1), so the objective function is still partly smooth everywhere with respect to (64). Hence, our algorithm will eventually generate iterates that have the same rank as the global optimum to which the iterates converge. If this optimum is low-rank, then so will the generated iterates be.

Driven by the fruitful and important applications of QSDPs in diverse fields, many efficient algorithms for solving them have been developed. We refer the readers to Li et al. (2018) for a comprehensive literature review and a powerful state-of-the-art solver, QSDPNAL, for the problem (QSDP).

To apply Algorithm 1, PG, or APG to (71), we need to perform the projection onto the feasible set

$$\mathcal{X} := \{X \in S_n^+ | \langle E, X \rangle = 0 \}. \quad (72)$$

The following lemma provides an effective way for performing such a projection.

**Lemma 7** Define $J := I_n - n^{-1}ee^\top \in S^n$, then for the set $\mathcal{X}$ defined in (72), it holds that

$$P_\mathcal{X}(G) = P_{S_n^+}(JGJ), \quad \forall G \in S^n.$$  

**Proof** First, for any $X \in \mathcal{X}$, clearly $Xe = 0$ and $e^\top X = 0$. Therefore, we observe that

$$\|X - JGJ\|_F^2 = \|X - \left( I_n - \frac{1}{n}ee^\top \right) G \left( I_n - \frac{1}{n}ee^\top \right) \|_F^2 \quad (73)$$

As a consequence, it holds that $P_\mathcal{X}(G) = P_{\mathcal{X}}(JGJ)$. Moreover, as $JGJe = JG(Je) = JG(0) = 0$, namely, $JGJ$ has an eigenvalue of 0 associated with the eigenvector $e$, we get that $P_{S_n^+}(JGJ) \in \mathcal{X}$ because the projection onto $S_n^+$ only truncates negative eigenvalues to zero in the eigen decomposition. Since $\mathcal{X} \subseteq S_n^+$, it follows that $P_{\mathcal{X}}(JGJ) = P_{S_n^+}(JGJ)$. 

From Lemma 7, we see that the computational bottleneck lies in the eigendecomposition of matrices in $S^n$, which could be highly expensive or even computationally prohibited in our high dimensional setting. Thus, we need to rely on low-rank approximate eigendecomposition to perform inexact projections.

Similar to (BM), we can also use the BM approach to solve (QSDP). In particular, the factorized problem takes the following form

$$\min_{W \in \mathbb{R}^{n \times k}} g(W) := \frac{1}{2} \|A(WW^\top) - b\|^2 + \langle C, WW^\top \rangle, \quad \text{s.t.} \quad W^\top e = 0. \quad (73)$$

The gradient of the function $g: \mathbb{R}^{n \times k} \to \mathbb{R}$ is then given as

$$\nabla g(W) = 2A^* \left( A(WW^\top) - b \right) W + 2CW,$$
and for any $D \in \mathbb{R}^{n \times k}$, the Hessian operator of $g$ performed on $D$ is given by

$$\nabla^2 g(W)[D] = 2A^* \left( A \left( WW^\top \right) - b \right) D + 2A^* \left( A \left( WD^\top + DW^\top \right) \right) W + 2CD.$$ 

Since $\{W \in \mathbb{R}^{n \times k} \mid W^\top e = 0\}$ defines a Riemannian manifold, by using the above information related to $g(\cdot)$, we can apply many efficient solvers for Riemannian optimization to solve (73). In our experiments in Section 6.2 for this QSDP problem, we use the state-of-the-art solver Manopt (Boumal et al., 2014) in our BM phase.

6. Numerical experiments

We conduct numerical experiments to exemplify the practical efficiency of the proposed algorithmic framework. In particular, we consider the two tasks discussed in Section 5 with large-scale real-world datasets in multicore environments. All algorithms are implemented in MATLAB and C/C++.

6.1 Matrix completion

The first task we consider is the matrix completion problem in the forms of (MC) and (MF). We use one toy example included in the package LIBPMF (https://www.cs.utexas.edu/~rofuyu/libpmf/) and four publicly available large-scale recommendation system data sets for this set of experiments. The only preprocessing we did was to transpose the data matrices when necessary to conform to our blanket assumption of $m \leq n$. For all data sets, we use their original training/test split. These data sets are summarized in Table 1. The column $|\Omega_{\text{test}}|$ indicates the number of entries in the test set. For the value of $\lambda$ on the real-world data, we follow the values provided by Hsieh and Olsen (2014) that were obtained through cross-validation, while the final $k$ is the rank of the final output of our algorithm, obtained by running our algorithm with the given $\lambda$ till the objective cannot be further improved. The value of $\lambda$ for the toy example is from some simple tuning to make the final rank not too far away from that of other datasets.

| Data set      | $m$    | $n$    | $|\Omega|$ | $|\Omega_{\text{test}}|$ | $\lambda$ | final $k$ |
|---------------|--------|--------|------------|---------------------------|-----------|-----------|
| toy-example   | 3952   | 6040   | 900189     | 100020                    | 36        | 62        |
| movielens100k| 943    | 1682   | 90570      | 9430                      | 15        | 68        |
| movielens10m | 65133  | 71567  | 9301274    | 698780                    | 100       | 50        |
| netflix       | 17770  | 2649429| 99072112   | 1408395                   | 300       | 68        |
| yahoo-music  | 624961 | 1000990| 252800275  | 4003960                   | 10000     | 52        |

Experiments on the first four datasets are conducted on an Amazon AWS EC2 c6i.4xlarge instance with an 8-core Intel Xeon Ice Lake processor and 32GB memory. For the larger

---

yahoo-music dataset, an m6i.4xlarge instance that has the same processor but with 64GB memory is used. Our experiments in this subsection utilize all cores available for all algorithms through parallelization by MATLAB and openMP.

For this task, we conduct four sets of experiments. First, we use the first two smaller datasets to see how different numbers of consecutive inexact proximal gradient iterations and consecutive epochs in the BM phase affect the behavior of our algorithm. Next, we empirically examine the result of Theorem 6 by checking how fast BM-Global identifies the active manifold, namely the correct rank. We then compare our whole method with its BM solver subroutine alone to see that our method is as efficient as the BM solver and can escape from stationary points of (MF) that are not global optima. In the last set of experiments, we compare our method with state of the art for (MC). We note that for this problem, $\nabla f$ is 1-Lipschitz continuous, and thus a fixed step size of $\alpha = 1.99$ can be used to satisfy (13) without any data-dependent computation. We have also tested a version that follows SpaRSA (Wright et al., 2009) to use the spectral step size initialization strategy of Barzilai and Borwein (1988) together with backtracking linesearch, but it did not result in better performance, and therefore we will use this fixed-step variant throughout. For completeness, we include the experiments with the SpaRSA variant in the supplementary materials.

To compare different methods, we consider two criteria, one from the optimization point of view and the other from the task-oriented angle. In particular, we first run our algorithm till the objective cannot be further improved, and take the obtained output $X^*$ as the numerical global optimal solution. With the knowledge of this $X^*$, our first criterion is the relative objective

$$\frac{F(X) - F(X^*)}{F(X^*)}.$$  

The second measure we use is the relative root mean squared error (RMSE), which is computed as

$$\frac{\text{RMSE}(X) - \text{RMSE}(X^*)}{\text{RMSE}(0) - \text{RMSE}(X^*)}, \quad \text{RMSE}(X) := \sqrt{\frac{\|P_{\Omega_{\text{test}}} (X - A)\|_F^2}{|\Omega_{\text{test}}|}}.$$  

Although in general the norm of the exact proximal gradient step would also be a better optimization progress measure especially because it does not require the knowledge of $F(X^*)$, its computation is impractical in this set of experiments because $mn$ is usually too huge for us to form $X_t$ explicitly and compute its exact SVD that is needed for calculating the exact proximal gradient step.

### 6.1.1 Parameter tuning for our method

We first use the toy example and movielens100k to finalize details in the parameters setting of our algorithm. In particular, we test the setting of alternating between $x$ consecutive inexact proximal gradient steps and $y$ consecutive iterations of the BM phase solver, with $x \in \{1, \ldots, 5\}$ and $y \in \{1, \ldots, 8\}$, for our fixed step variant with $\alpha \equiv 1.99$. More Details and the results are shown in the supplementary materials. Our result indicates that there is no definite best performer in all cases. But in general, $x = 1$ and $y = 3$ seems to be a rather robust choice. This observation accords with our argument that eigendecompositions
are rather expensive and the BM steps should be utilized more often than the proximal
gradient steps. We therefore will stick to this setting in all the remaining experiments in
this subsection.

6.1.2 Stabilization of the rank
We then show the rank of $X_t$ over iterations of BM-Global In Fig. 1, we use solid lines and
dash lines to respectively show the relative objective value and the rank of the iterates of
our method. The gray line represents the rank at the optimum $X^*$. We can see that the
rank of $X_t$ increases quickly at first, and eventually stabilizes at the rank of the point of
convergence in all cases. Sometimes, the rank remains fixed for a while, then is increased by
a small number, and finally stays at the new rank. This is the situation that a safeguard (see
the supplementary materials) kicks in to resolve the insufficient rank problem and ensures
that the iterates indeed converge to a global optimum. We can also see that when the rank
reaches rank($X^*$), the relative objective also drops significantly, indicating that finding the
right rank is essential in solving (CVX) to a high precision.

![Figure 1: Rank and relative objective of the iterates of BM-Global over running time.](image.png)

6.1.3 Comparison between BM-Global and the BM solver alone
We next compare BM-Global with running a BM solver only for (MF). We directly use the
solver in our BM phase, namely, the polyMF-SS method of Wang et al. (2017), with their
original random start scheme to avoid starting from the origin, which is a known saddle
point of (MF). Given any value $k$, their method starts from a randomized $W \in \mathbb{R}^{m \times k}$ and
$H \in \mathbb{R}^{n \times k}$. We favor their method to directly assign $k$ as the final rank shown in Table 1,
but we emphasize that in real-world applications, finding this $k$ will require additional effort
in parameter search.

The purpose of this experiment is to show that solving (MF) only can get the iter-
ates stuck at saddle points or spurious local minima, while BM-Global can effectively and
efficiently escape from such points. Therefore, we consider the relative objective as the
only comparison criterion in this experiment. The results of running time and number of
iterations are shown in Fig. 2. For the number of iterations, we count either one inexact
proximal gradient step or one epoch of polyMF-SS (one sweep through the whole data) as
one iteration.

We observe that in terms of iterations, PolyMF-SS has a small early advantage due
to the larger starting rank in (BM). But its convergence quickly slows down, suggesting
that likely the iterates are attracted to a saddle point or a spurious local minimum that is
strictly worse than the global optima. On the other hand, the story in the running time comparison is very different. We see that the higher rank in PolyMF-SS from the beginning on actually increases the time cost per epoch, and thus the early advantage of PolyMF-SS over BM-Global we observed in terms of iterations is not present in the time comparison. Another observation is that in the numerical experiments, the empirical convergence speed of BM-Global is nearly $Q$-linear, linking to the case of $\theta = 1/2$ in Theorem 4.

Overall speaking, BM-Global is as efficient as running a solver for (MF) alone, but it provides multiple advantages including the guarantee of convergence to the global optima. Although in this experiment, the stationary points to which the iterates of PolyMF-SS converge seem to be of good enough quality, we have no guarantee that on other datasets, or even on these datasets but with a different $\lambda$, their points of convergence will still be of satisfactory quality.

![Figure 2: Comparison between PolyMF-SS and BM-Global. Top row: iterations v.s. relative objective. Bottom row: running time (seconds) v.s. relative objective.](image)

### 6.1.4 Comparison with existing methods

Now that it is clear our method is advantageous over running a solver for (MF) alone, we proceed to compare BM-Global with state of the art for (MC). In particular, we compare BM-Global with the following:

- **Active-ALT** (Hsieh and Olsen, 2014): This method alternates between conducting an inexact PG step and solving a lower-dimensional convex subproblem. In the approximate SVD part for inexact PG, Hsieh and Olsen (2014) use the power method with warmstart from the output of the previous iteration plus some random columns as a safeguard.
• AIS-Impute (Yao and Kwok, 2018): An inexact APG method that also uses the power method for approximate SVDs. They use the combination of the outputs of the previous iteration and the iteration preceding it to form the warmstart matrix.

The inexact APG method in Toh and Yun (2010) is not included because the underlying APG part is the same as that of AIS-Impute, but their approximate SVD using Lanczos is shown by Yao and Kwok (2018) to be less efficient.

The results of relative objective and relative RMSE are shown in Fig. 3. Note that the running time for relative RMSE in Fig. 3 is in log scale to make the difference legible. Clearly, BM-Global outperforms the state-of-the-art for (MC) significantly on both criteria. Particularly, Fig. 3 exemplifies even greater efficiency difference in reaching satisfactory RMSE between BM-Global and existing methods. We can see that the proposed approach is actually magnitudes faster than state-of-the-art in this criterion.

Figure 3: Comparison between BM-Global and existing methods. Top row: running time v.s. relative objective. Bottom row: running time (log scale) v.s. relative RMSE.

6.2 Convex QSDP

Next, we consider solving the two applications of QSDP described in Section 5.2. As mentioned before, PG and APG can be applied to solve the problem directly. However, based on our empirical experience, both methods require too many iterations and excessive runtime to reach a reasonably good solution, so their numerical results are excluded here. (Interested readers may refer to the supplementary material for the numerical results of the APG methods.) We hence only compare BM-Global with the efficient and robust QSDP solver, QSDPNAL (Li et al., 2018).³

³ Available at https://blog.nus.edu.sg/mattohkc/softwares/qsdpnal/
Regarding the termination conditions, since QSDPNAL computes both primal and dual iterates, its relative KKT residual is computable (see (Li et al., 2018, Section 5.2) for the definition). Thus, given a specific stopping tolerance \( \texttt{tol} \), QSDPNAL is terminated when the maximum relative KKT residual, denoted by \( \eta_{kkt} \), is less than \( \texttt{tol} \). Moreover, when \( n \) is large, QSDPNAL may take too much computational time (since it uses full eigendecompositions), so we also cap the running time of QSDPNAL to four hours (initialization overhead excluded) and its maximum number of iterations to 200. For \texttt{BM-Global}, we terminate it when

\[
\frac{|f(W_t W_t^\top) - f(W_{t-1} W_{t-1}^\top)|}{(1 + |f(W_{t-1} W_{t-1}^\top)|)} < \texttt{tol}.
\]

In our experiments, we set \( \texttt{tol} = 10^{-6} \) for both methods.

Recall that the first-order optimality condition for problem (QSDP) is given by

\[
X - P_X (X - \nabla f(X)) = 0, \quad X \in \mathbb{S}^n.
\]

Since we are testing problems with \( n \) that can be handled by QSDPNAL that uses full eigendecompositions, we are in fact able to check whether an approximate solution \( X \in \mathbb{S}^n \) is optimal numerically, even though this can be time-consuming. Therefore, to compare the quality of the solutions returned by \texttt{BM-Global} and QSDPNAL, we record the relative primal feasibility and the relative optimality, respectively defined as

\[
\eta_{\text{prim}}(X) := \frac{|\langle E, X \rangle|}{1 + \|X\|_F}, \quad \text{and} \quad \eta_{\text{opt}}(X) := \frac{\|X - P_X (X - \nabla f(X))\|_F}{1 + \|X\|_F + \|\nabla f(X)\|_F}.
\]

Experiments for this part are conducted on a Linux PC with an Intel Xeon E5-2650 processor and 96GB memory.

### 6.2.1 Regularized kernel estimation

We consider problems with dissimilarity measures \( d_{ij} \) for \( 1 \leq i, j \leq n \) collected in Duin and Pekalska (2009).\footnote{Data available at \url{http://prtools.tudelft.nl/Guide/37Pages/distools.html}.} In our experiments, the data \( d_{ij} \) are scaled to the interval \([0, 1]\), and the elements of the index set \( \Omega \) are randomly selected such that \(|\Omega| \approx n/20\). We set \( w_{ij} = 1 \) for all \((i,j) \in \Omega\) and \( \lambda = \sqrt{n}/10 \).

The results are presented in Table 2. Clearly, both methods are able to compute nearly feasible and low-rank solutions. In terms of the optimality measure, \texttt{BM-Global} is able to solve all the problems with \( \eta_{\text{opt}} < 10^{-3} \) while QSDPNAL fails to do so for one of the problems. In terms of efficiency, we see that \texttt{BM-Global} is faster in most cases, and \texttt{BM-Global} can even be ten times faster than QSDPNAL in the case of the largest instance.

### 6.2.2 Molecular conformation

In this experiment, we consider the molecules from the Protein Data Bank (see \url{https://www.rcsb.org/}) with given noisy and sparse distance data to simulate distances measurable by nuclear magnetic resonance (NMR) experiments. For each molecule, if the distance between two compatible atoms is less than 6Å\( (6 \times 10^{-8} \text{ cm}) \), then the distance can be measured by the NMR experiment; otherwise, we assume that no information is known.
<table>
<thead>
<tr>
<th>Name</th>
<th>( n )</th>
<th>( \eta_{\text{prim}} )</th>
<th>( \eta_{\text{opt}} )</th>
<th>( \eta_{\text{kkt}} )</th>
<th>( \text{rnk} )</th>
<th>Time</th>
<th>( \eta_{\text{prim}} )</th>
<th>( \eta_{\text{opt}} )</th>
<th>( \text{rnk} )</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrainMRI</td>
<td>124</td>
<td>6e-11</td>
<td>1e-06</td>
<td>1e-06</td>
<td>5</td>
<td>0.7</td>
<td>2e-16</td>
<td>5e-07</td>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>protein</td>
<td>213</td>
<td>4e-13</td>
<td>4e-06</td>
<td>5e-07</td>
<td>24</td>
<td>4.3</td>
<td>5e-11</td>
<td>7e-06</td>
<td>24</td>
<td>1.9</td>
</tr>
<tr>
<td>CoilDelftDiff</td>
<td>288</td>
<td>9e-13</td>
<td>4e-04</td>
<td>9e-07</td>
<td>32</td>
<td>5.7</td>
<td>4e-15</td>
<td>5e-08</td>
<td>32</td>
<td>1.6</td>
</tr>
<tr>
<td>coldefltsame</td>
<td>288</td>
<td>9e-14</td>
<td>4e-06</td>
<td>5e-07</td>
<td>32</td>
<td>6.7</td>
<td>2e-11</td>
<td>4e-06</td>
<td>32</td>
<td>2.3</td>
</tr>
<tr>
<td>CoilYork</td>
<td>300</td>
<td>2e-13</td>
<td>2e-06</td>
<td>4e-07</td>
<td>22</td>
<td>4.8</td>
<td>2e-15</td>
<td>7e-06</td>
<td>22</td>
<td>2.6</td>
</tr>
<tr>
<td>Chickenpieces-5-45</td>
<td>445</td>
<td>4e-13</td>
<td>3e-06</td>
<td>7e-07</td>
<td>27</td>
<td>11.5</td>
<td>2e-11</td>
<td>8e-06</td>
<td>28</td>
<td>7.6</td>
</tr>
<tr>
<td>protein</td>
<td>213</td>
<td>4e-13</td>
<td>4e-06</td>
<td>5e-07</td>
<td>24</td>
<td>4.3</td>
<td>5e-11</td>
<td>7e-06</td>
<td>24</td>
<td>1.9</td>
</tr>
<tr>
<td>newgroups</td>
<td>600</td>
<td>8e-13</td>
<td>3e-05</td>
<td>5e-07</td>
<td>81</td>
<td>39.6</td>
<td>2e-12</td>
<td>2e-04</td>
<td>83</td>
<td>29.6</td>
</tr>
<tr>
<td>flowcytodis</td>
<td>612</td>
<td>2e-13</td>
<td>4e-06</td>
<td>9e-07</td>
<td>15</td>
<td>14.8</td>
<td>1e-12</td>
<td>1e-06</td>
<td>15</td>
<td>13.9</td>
</tr>
<tr>
<td>DelftPedestrians</td>
<td>689</td>
<td>3e-12</td>
<td>1e-05</td>
<td>2e-07</td>
<td>69</td>
<td>43.7</td>
<td>3e-12</td>
<td>7e-06</td>
<td>69</td>
<td>32.8</td>
</tr>
<tr>
<td>WoodyPlants50</td>
<td>791</td>
<td>5e-13</td>
<td>1e-05</td>
<td>7e-07</td>
<td>48</td>
<td>44.3</td>
<td>9e-13</td>
<td>2e-07</td>
<td>48</td>
<td>78.2</td>
</tr>
<tr>
<td>deftgestures</td>
<td>1500</td>
<td>1e-14</td>
<td>7e-06</td>
<td>5e-07</td>
<td>76</td>
<td>318.3</td>
<td>3e-13</td>
<td>2e-06</td>
<td>77</td>
<td>390.1</td>
</tr>
<tr>
<td>zongker</td>
<td>2000</td>
<td>2e-11</td>
<td>1e-02</td>
<td>8e-07</td>
<td>264</td>
<td>1000.2</td>
<td>6e-12</td>
<td>1e-04</td>
<td>267</td>
<td>665.6</td>
</tr>
<tr>
<td>polydish57</td>
<td>4000</td>
<td>2e-12</td>
<td>1e-05</td>
<td>5e-07</td>
<td>100</td>
<td>3508.9</td>
<td>7e-15</td>
<td>3e-05</td>
<td>101</td>
<td>1765.2</td>
</tr>
<tr>
<td>polydism57</td>
<td>4000</td>
<td>1e-12</td>
<td>5e-04</td>
<td>8e-07</td>
<td>25</td>
<td>3286.4</td>
<td>1e-16</td>
<td>2e-07</td>
<td>26</td>
<td>305.0</td>
</tr>
</tbody>
</table>

Table 2: Computational results on regularized kernel estimation problems.

for this pair. To simulate the sparse set of distances measurable by the NMR experiment, among all the pairwise distances less than 6\( \text{Å} \), we select 25\% of them to generate our index set \( \Omega \). We then add in additional noise to the observed data as follows. Let \( \tau \) be a given noise level and \( \hat{d}_{ij} \) be the exact distance between atom \( i \) and atom \( j \) for \( (i,j) \in \Omega \), we sample two independent random variables \( \epsilon_{ij} \), \( \epsilon_{ij} \) from the normal distribution \( N(0, \pi \tau^2/2) \) and define

\[
\bar{d}_{ij} := \max\{1, (1 - |\epsilon_{ij}|) \hat{d}_{ij}\}, \quad \tilde{d}_{ij} := (1 + |\epsilon_{ij}|) \hat{d}_{ij}.
\]

Then, the input distances are set as \( d_{ij} := (\bar{d}_{ij} + \tilde{d}_{ij})/2 \). Given \( d_{ij} \), we set \( w_{ij} = 1/d_{ij}^2 \). Moreover, we let \( \lambda = -10\sqrt{n}/\sum_{(i,j)\in\Omega} d_{ij}^2 \). In our tests, we set \( \tau = 0.1 \). To measure the accuracy of the estimated positions, we record the root mean square deviation (RMSD):

\[
\text{RMSD} := \frac{1}{\sqrt{n}} \left( \sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \right)^{1/2}
\]

where \( x_i \) is the estimated position and \( \hat{x}_i \) is the actual position. Note that a smaller RMSD means a better estimation, and an RMSD of less than 2\( \text{Å} \) is considered to be good in molecular conformation.

The computational results are presented in Table 3. It is clear that both methods return nearly feasible solutions. However, we can see that BM-Global outperforms QSDPNAL in all other measures. In particular, QSDPNAL often returns solutions with a large suboptimality measure, and those solutions tend to be of a higher rank and give a larger RMSD. On the other hand, the solutions returned by BM-Global are always of low rank with very small RMSD. Moreover, by the presented computational time, we see that BM-Global is much more efficient than QSDPNAL, and its generated solutions are also often much more accurate.

The results in this and the previous subsections also suggest that the numerical performance of QSDPNAL may depend on the sign of \( \lambda \) while BM-Global is robust with respect to parameter selection of \( \lambda \).
Nuclear-norm regularized low-rank optimization through BM decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>$\eta_{prim}$</th>
<th>$\eta_{opt}$</th>
<th>$\eta_{kkt}$</th>
<th>$\eta_{rk}$</th>
<th>RMSD</th>
<th>Time</th>
<th>$\eta_{prim}$</th>
<th>$\eta_{opt}$</th>
<th>$\eta_{rk}$</th>
<th>RMSD</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1PBM</td>
<td>126</td>
<td>1e-12</td>
<td>9e-09</td>
<td>7e-07</td>
<td>10</td>
<td>2.7</td>
<td>25.0</td>
<td>1e-13</td>
<td>3e-08</td>
<td>10</td>
<td>2.7</td>
<td>2.0</td>
</tr>
<tr>
<td>1AU6</td>
<td>161</td>
<td>3e-12</td>
<td>3e-08</td>
<td>8e-07</td>
<td>12</td>
<td>1.1</td>
<td>34.1</td>
<td>1e-13</td>
<td>3e-08</td>
<td>12</td>
<td>1.1</td>
<td>4.0</td>
</tr>
<tr>
<td>1PTQ</td>
<td>402</td>
<td>2e-12</td>
<td>4e-08</td>
<td>8e-07</td>
<td>14</td>
<td>0.7</td>
<td>358.6</td>
<td>2e-16</td>
<td>3e-08</td>
<td>15</td>
<td>0.7</td>
<td>13.9</td>
</tr>
<tr>
<td>1CTF</td>
<td>487</td>
<td>3e-12</td>
<td>8e-09</td>
<td>9e-07</td>
<td>13</td>
<td>0.7</td>
<td>570.3</td>
<td>1e-15</td>
<td>3e-08</td>
<td>15</td>
<td>0.8</td>
<td>18.0</td>
</tr>
<tr>
<td>1HOE</td>
<td>558</td>
<td>1e-12</td>
<td>2e-08</td>
<td>1e-06</td>
<td>15</td>
<td>0.8</td>
<td>899.6</td>
<td>7e-16</td>
<td>1e-08</td>
<td>16</td>
<td>0.7</td>
<td>32.7</td>
</tr>
<tr>
<td>1LFB</td>
<td>641</td>
<td>5e-13</td>
<td>1e-08</td>
<td>3e-06</td>
<td>15</td>
<td>1.2</td>
<td>2739.5</td>
<td>4e-16</td>
<td>3e-08</td>
<td>17</td>
<td>1.5</td>
<td>25.5</td>
</tr>
<tr>
<td>1PHT</td>
<td>666</td>
<td>2e-13</td>
<td>8e-08</td>
<td>1e-06</td>
<td>15</td>
<td>1.1</td>
<td>2091.9</td>
<td>8e-16</td>
<td>4e-08</td>
<td>18</td>
<td>1.4</td>
<td>36.3</td>
</tr>
<tr>
<td>1F39</td>
<td>767</td>
<td>4e-12</td>
<td>2e-08</td>
<td>3e-06</td>
<td>17</td>
<td>1.3</td>
<td>3230.1</td>
<td>8e-16</td>
<td>2e-08</td>
<td>20</td>
<td>0.8</td>
<td>44.0</td>
</tr>
<tr>
<td>1DCH</td>
<td>806</td>
<td>7e-13</td>
<td>2e-08</td>
<td>4e-06</td>
<td>18</td>
<td>1.6</td>
<td>3897.0</td>
<td>3e-16</td>
<td>4e-09</td>
<td>18</td>
<td>1.1</td>
<td>41.8</td>
</tr>
<tr>
<td>1HQQ</td>
<td>891</td>
<td>6e-13</td>
<td>2e-08</td>
<td>4e-06</td>
<td>17</td>
<td>2.7</td>
<td>5089.2</td>
<td>7e-16</td>
<td>1e-08</td>
<td>21</td>
<td>1.0</td>
<td>43.5</td>
</tr>
<tr>
<td>1POA</td>
<td>914</td>
<td>6e-12</td>
<td>2e-08</td>
<td>4e-06</td>
<td>16</td>
<td>2.1</td>
<td>4734.1</td>
<td>1e-15</td>
<td>4e-09</td>
<td>19</td>
<td>1.1</td>
<td>65.4</td>
</tr>
<tr>
<td>1AX8</td>
<td>1003</td>
<td>6e-13</td>
<td>2e-08</td>
<td>4e-06</td>
<td>17</td>
<td>3.0</td>
<td>6397.0</td>
<td>2e-16</td>
<td>4e-09</td>
<td>18</td>
<td>1.7</td>
<td>61.1</td>
</tr>
<tr>
<td>1TJO</td>
<td>1394</td>
<td>5e-13</td>
<td>2e-08</td>
<td>5e-06</td>
<td>21</td>
<td>12.3</td>
<td>-</td>
<td>2e-15</td>
<td>9e-10</td>
<td>29</td>
<td>2.3</td>
<td>77.4</td>
</tr>
<tr>
<td>1RGS</td>
<td>2015</td>
<td>3e-12</td>
<td>8e-08</td>
<td>6e-06</td>
<td>39</td>
<td>16.1</td>
<td>-</td>
<td>1e-15</td>
<td>4e-09</td>
<td>31</td>
<td>2.3</td>
<td>168.8</td>
</tr>
<tr>
<td>1TOA</td>
<td>2138</td>
<td>9e-12</td>
<td>2e-07</td>
<td>8e-06</td>
<td>49</td>
<td>16.8</td>
<td>-</td>
<td>2e-15</td>
<td>2e-09</td>
<td>31</td>
<td>1.0</td>
<td>142.4</td>
</tr>
<tr>
<td>1KDH</td>
<td>2846</td>
<td>3e-11</td>
<td>2e-06</td>
<td>2e-05</td>
<td>150</td>
<td>21.9</td>
<td>-</td>
<td>5e-16</td>
<td>2e-09</td>
<td>40</td>
<td>2.1</td>
<td>199.1</td>
</tr>
<tr>
<td>1NFG</td>
<td>3501</td>
<td>2e-12</td>
<td>3e-05</td>
<td>1e-04</td>
<td>325</td>
<td>21.2</td>
<td>-</td>
<td>3e-15</td>
<td>3e-09</td>
<td>43</td>
<td>1.0</td>
<td>275.8</td>
</tr>
<tr>
<td>1BPM</td>
<td>3672</td>
<td>1e-12</td>
<td>5e-05</td>
<td>2e-04</td>
<td>396</td>
<td>23.8</td>
<td>-</td>
<td>5e-17</td>
<td>8e-10</td>
<td>40</td>
<td>1.4</td>
<td>438.6</td>
</tr>
<tr>
<td>1MQQ</td>
<td>5510</td>
<td>1e-14</td>
<td>4e-04</td>
<td>1e-03</td>
<td>911</td>
<td>26.0</td>
<td>-</td>
<td>2e-15</td>
<td>1e-09</td>
<td>61</td>
<td>1.3</td>
<td>947.1</td>
</tr>
</tbody>
</table>

Table 3: Computational results on regularized molecular conformation problems. “-” indicates that the solver is terminated because the maximum running time of four hours is reached.

7. Conclusions

In this work, we proposed an efficient algorithm BM-Global for solving the low-rank matrix optimization problem. We utilized both the efficiency from a smooth objective of the Burer-Monteiro decomposition approach and the convexity and partial smoothness of the nuclear-norm-regularized convex form to obtain a highly efficient algorithm with sound theoretical guarantees. Extensive numerical experiments showed that our proposed algorithm outperforms state-of-the-art for low-rank matrix optimization.

Acknowledgement

This work was partially done when Lee was visiting the Department of Mathematics at the National University of Singapore. Lee’s research was supported in part by NSTC of Taiwan grants 109-2222-E-001-003-MY3 and 111-2628-E-001-003-, and Academia Sinica grant AS-GCS-111-M05. Toh’s research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 3 grant call (MOE-2019-T3-1-010).

References


Supplementary Materials

Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Implementation Details</td>
<td>36</td>
</tr>
<tr>
<td>I.1 Matrix Completion</td>
<td>36</td>
</tr>
<tr>
<td>I.2 Quadratic SDP</td>
<td>43</td>
</tr>
<tr>
<td>II Additional Experimental Details and More Experiments</td>
<td>44</td>
</tr>
<tr>
<td>II.1 The SpaRSA variant for Matrix Completion</td>
<td>44</td>
</tr>
<tr>
<td>II.2 Experimental Details of Section 6.1.1</td>
<td>45</td>
</tr>
<tr>
<td>II.3 Comparison with the SpaRSA variant for Matrix Completion</td>
<td>45</td>
</tr>
<tr>
<td>II.4 Parallelism</td>
<td>45</td>
</tr>
<tr>
<td>II.5 Numerical results for the APG method</td>
<td>48</td>
</tr>
</tbody>
</table>

I. Implementation Details

I.1 Matrix Completion

We now describe our implementation details of BM-Global that are tailored for the matrix completion problem. In particular, we will discuss the mechanism for deciding $\alpha_t$ in (12), the algorithm for obtaining the approximate eigendecomposition using only matrix-vector products, details of the safeguard to ensure $\epsilon_t \downarrow 0$ in (12), initialization strategy for $X_0$, the solver for (BM-nuclear) and (BM-PSD), and the degree of parallelism of our algorithm.

To avoid redundancy, we focus on the case of (BM-nuclear) in this section, and keep in our mind that it can be easily adapted to the case of (BM-PSD) by straightforward changes from SVDs to eigendecompositions.

I.1.1 Approximate SVD

Let us denote

$$Z_t := \tilde{X}_t - \alpha_t \nabla f(\tilde{X}_t).$$

For the proximal operation (4), since $m \leq n$ by our assumption, it is cheaper to compute an approximate eigendecomposition of $Z_t Z_t^\top$ (instead of $Z_t^\top Z_t$) to get

$$Z_t Z_t^\top \approx \hat{U}_t \text{diag} \left((\hat{\Sigma}_t)^2\right) \hat{U}_t^\top$$

for some $\Sigma_t \in \mathbb{R}^{kt}$ with $\Sigma_t > 0$ and some orthogonal $\hat{U}_t \in \mathbb{R}^{m \times kt}$ for a given rank $k_t$. The notation $(\hat{\Sigma}_t)^2$ denotes the element-wise square. Note that here we have removed the eigenvectors corresponding to the eigenvalue 0 as it does not affect the product matrix.
We then conduct an exact SVD on $\hat{U}_t^T Z_t \in \mathbb{R}^{k_t \times n}$ (whose calculation can be done without forming $Z_t$ or $\hat{X}_t$ explicitly) to obtain

$$\hat{U}_t^T Z_t = \hat{U}_t \text{diag} \left( \hat{\Sigma}_t \right) V_t^T,$$

with cost $O(k_t^3 + nk_t^2)$, which is much cheaper than SVD for $Z_t$ when $k_t \ll m$. The approximate SVD of $Z_t$ is then obtained through

$$Z_t \approx U_t \text{diag} \left( \hat{\Sigma}_t \right) V_t^T =: \tilde{Z}_t,$$

where $\tilde{Z}_t$ coincides with the one we used in (10) and (11). Clearly, $U_t$ and $V_t$ are both orthonormal, so this is indeed a valid SVD for the matrix

$$\tilde{Z}_t = \hat{U}_t \hat{U}_t^T Z_t = P_{\text{ran}(\hat{U}_t)}(Z_t).$$

The inexact proximal gradient step is then finished as

$$X_{t+1}^+ (\alpha_t) = U_t \text{diag} (\Sigma_t) V_t^T, \quad \Sigma_t := \left[ \hat{\Sigma}_t - \lambda \alpha_t e \right]_+.$$

Note that we only store $U_t, \Sigma_t, V_t$ but do not explicitly form $X_{t+1}$. Here we slightly abuse the notation to let $\Sigma_t$ denote only the coordinates of the thresholded vector that have a nonzero value, and let $U_t$ and $V_t$ contain only the columns corresponding to these values to save spatial and computational cost.

### I.1.2 An Efficient Algorithm for Approximate Eigendecompositions Using Only Matrix-Matrix Products

As mentioned before, forming $X_t$ and $\hat{X}_t$ is prohibitively expensive. Therefore, for computing (I.2), we need to rely on iterative methods that only require evaluating matrix multiplications involving $Z_t$ and $Z_t^T$, so that we can utilize the decomposition of $\hat{X}_t = W_t H_t^T$ as well as the structured assumption of $\nabla f(X)$ made in Section 1. A highly efficient and robust approach is the limited memory block Krylov subspace method (LmSVD) proposed by Liu et al. (2013); see also the references therein for other popular algorithms. LmSVD is an extension of the classic simple subspace iteration (SSI) method for computing the extremal eigenvalues and eigenvectors that extends from the renowned power method (i.e., $k_t = 1$).

For any initial guess for right-singular vectors $U_{t,0} \in \mathbb{R}^{m \times k_t}$, the SSI method computes the new iterates $U_{t,i}$ via

$$U_{t,i} \leftarrow \text{orth} \left( Z_t Z_t^T U_{t,i-1} \right), \quad \forall i \geq 1,$$

where $\text{orth}(M)$ extracts an orthonormal basis for the range space of the given matrix $M$, and $i$ is the iteration counter for SSI. For the ease of description, we abstract the operation $Z_t Z_t^T(U)$ for an input $U$ as a self-adjoint semidefinite operator $L_t()$. Note that in each iteration of SSI, one needs to perform two matrix multiplications (one for $Z_t^T$ and the other for $Z_t$) and one orthonormalization that cost $O(mn k_t)$ and $O(m k_t^2)$ flops, respectively. In our case, suppose that $k_t \ll m \leq n$, then the main computational bottleneck is the matrix products. Therefore, LmSVD tries to accelerate the practical convergence via cutting down the total number of iterations to reduce such matrix products without incurring additional
heavy computation. To achieve this goal, LmSVD finds the next iterate via replacing $U_{t,i}$ in (I.6) with an improved candidate $\hat{U}_{t,i}$ via solving the following constrained optimization problem:

$$\hat{U}_{t,i} = \text{argmax} \left\{ \langle U, Z_t Z_t^\top U \rangle \mid U^T U = I_{k_t}, \text{ran}(U) \subseteq S_{t,i} \right\},$$

(I.7)

where the subspace $S_{t,i} \subseteq \mathbb{R}^m$ is selected as

$$S_{t,i} = \text{ran}(U_{t,i}, U_{t,i-1}, \ldots, U_{t,i-p_t}), \quad U_{t,i} := L_t(\hat{U}_{t,i-1}), \quad j = i, i-1, \ldots, i-p_t$$

for some pre-specified $p_t \geq 0$. Let $q_t := k_t(p_t + 1)$ and

$$P_{t,i} := [U_{t,i}, U_{t,i-1}, \ldots, U_{t,i-p_t}] \in \mathbb{R}^{m \times q_t},$$

then $U \in S_{t,i}$ if and only if there exists $V \in \mathbb{R}^{q_t \times k_t}$ such that

$$U = P_{t,i} V.$$  

(I.8)

Direct computation then shows that (I.7) is equivalent to

$$\max_{V \in \mathbb{R}^{q_t \times k_t}} \langle V, ((P_{t,i})^\top L_t(P_{t,i} V)) \rangle \quad \text{s.t.} \quad V^\top ((P_{t,i})^\top P_{t,i}) V = I_{k_t}. \tag{I.9}$$

However, the matrix $P_{t,i}$ may be rank deficient to cause numerical issues in solving (I.9). To resolve this issue, LmSVD replaces $P_{t,i}$ with an orthonormal basis of ran($P_{t,i}$). To extract such a basis, since $U_{t,i}$ (i.e., the first block in matrix $P_{t,i}$) always has a full column rank, LmSVD first projects the remaining blocks in $P_{t,i}$ to ker($(U_{t,i})^\top$) to form

$$P_{t,i} := \left( I_m - U_{t,i}(U_{t,i})^\top \right) [U_{t,i-1}, \ldots, U_{t,i-p_t}] \in \mathbb{R}^{m \times p_t k_t}, \tag{I.10}$$

and then consider its orthonormalization. We denote the eigendecomposition of $P_{t,i}^\top P_{t,i} \in \mathbb{R}^{pk_t \times pk_t}$ by

$$P_{t,i}^\top P_{t,i} = \tilde{U}_{P_{t,i}} \tilde{\Lambda}_{P_{t,i}} \tilde{U}_{P_{t,i}}^\top,$$

for matrices $\tilde{U}_{P_{t,i}}, \tilde{\Lambda}_{P_{t,i}} \in \mathbb{R}^{pk_t \times pk_t}$ with $\tilde{U}_{P_{t,i}}$ orthonormal and $\tilde{\Lambda}_{P_{t,i}}$ diagonal. Clearly, if $\tilde{\Lambda}_{P_{t,i}}$ is nonsingular,

$$\hat{P}_{t,i} := \left[ U_{t,i}, U_{t,i} \tilde{U}_{P_{t,i}} \tilde{\Lambda}_{P_{t,i}}^{-1/2} \right] \tag{I.11}$$

is an orthonormal basis of ran($P_{t,i}$). In practice, we can drop those columns of $\hat{P}_{t,i}$ that correspond to nearly zero eigenvalues of $P_{t,i}^\top P_{t,i}$. Moreover, there may exist columns of $\hat{P}_{t,i}$ whose norms are nearly zero. To stabilize the numerical computation, one might also want to drop these columns. From here on, we always assume that $\hat{P}_{t,i} \in \mathbb{R}^{m \times q_{t,i}}$, for some $q_{t,i} > 0$, forms an orthonormal basis of ran($P_{t,i}$) and it is obtained via performing the above two trimming procedures to the matrix in (I.11).

After knowing an orthonormal basis $\hat{P}_{t,i}$ of $P_{t,i}$, we then express any $U \in S_{t,i}$ as

$$U = \hat{P}_{t,i} V$$
Algorithm 2: LmSVD\((L_t, U_t^{0}, p_t)\)

input : A self-adjoint semidefinite operator \(L_t\) over \(\mathbb{R}^m\), an initial guess \(U_t^{0} \in \mathbb{R}^{m \times k_t}\), a threshold \(\epsilon > 0\) for ruling out small eigenvalues, and an integer \(p_t > 0\)

1 \(U_t^{0} \leftarrow \text{orth}(U_t^{0})\)

2 for \(i = 0, \ldots,\) do

3 \(p \leftarrow \min\{p_t, i\}\)

4 \(S_{t,i} \leftarrow \text{ran}\left(U_t^{i}, U_t^{i-1}, \ldots, U_t^{i-p}\right)\) *Block Krylov subspace selection

5 \(\hat{U}_t^{i,i} \leftarrow \text{argmax}\{\langle U, L_t(U) \rangle \mid U^\top U = I_{k_t}, U \in S_{t,i}\}\) *Block subspace optimization

6 \(U_t^{i,i+1} \leftarrow \text{orth}\left(L_t(\hat{U}_t^{i,i})\right)\) *Orthonormalization

7 \(s \leftarrow \left(\text{diag}\left((U_t^{i,i+1})^\top L_t(U_t^{i,i+1})\right)\right)^{1/2}\)

8 \(J_s \leftarrow \text{find}(s > \epsilon)\) *Rule out small eigenvalues and associated eigenvectors

9 \(\hat{U}_t \leftarrow (U_t^{i,i+1})_{:,J_s}\)

output: \(\hat{U}_t\)

for some \(V \in \mathbb{R}^{q_t,i \times k_t}\). The above expression then yields the following optimization problem to be solved at each iteration of LmSVD.

\[
\max_{V \in \mathbb{R}^{q_t,i \times k_t}} \langle V, L_t V \rangle \quad \text{s.t.} \quad V^\top V = I_{k_t}, \tag{I.12}
\]

where \(L_t^{i,i} := (\hat{P}_t^{i,i})^\top L_t(\hat{P}_t^{i,i}) \in \mathbb{R}^{q_t,i \times q_t,i}\). The solution \(V_t^{j,i}\) for problem (I.12) is nothing but the \(k_t\) leading eigenvectors of the matrix \(L_t^{i,i}\). Therefore, we can compute the full spectral decomposition of \(L_t^{i,i}\) to get \(V_t^{j,i}\) and the computational cost is acceptable provided that \(p_t\) and \(k_t\) are small. The overall algorithm of LmSVD is summarized in Algorithm 2.

We emphasize that LmSVD has an efficient and robust official implementation by Liu et al. (2013). In the present work, we borrow most parts of the implementation of Liu et al. (2013) but impose some minor modifications to adapt for our purposes. First, as we shall see in the following subsection, instead of using a randomly generated initial point \(U_t^{0}\), we use a more sophisticated initialization scheme. Second, the implementation of Liu et al. (2013) terminates by following a two-level strategy, but in our implementation, we simply terminate the algorithm as long as the difference between the eigenvalues of \(L_t^{i,i}\) and \(L_t^{i,i-1}\) is small.

I.1.3 Ensuring Sufficient Precision in the Proximal Operation

We notice that (I.5) suggests that all entries in \(\hat{\Sigma}_t\) smaller than the threshold as well as their corresponding columns of \(V_t\) and \(U_t\) do not contribute to the calculation of \(X_t^+ (\alpha_t)\), so we just need to compute the entries not truncated by the proximal operation. Therefore, if \(\text{rank}(X_{t+1}) = k_{t+1}\), ideally we just need to compute the first \(k_{t+1}\) eigenvalues in our approximate eigendecomposition at the \(t\)th iteration of proximal gradient. On the other

hand, to ensure that we are recovering a global solution \( X^* \) of (CVX), it is necessary to check that the ranks of the iterates are large enough so that we do not get stuck at an approximation of \( X^* \) with an insufficient rank. To safeguard that our algorithm converges to a global optimum, or more explicitly, to make \( \epsilon_t \) in (12) decrease to 0 fast enough (see Section 4), we want to ensure that eventually the smallest eigenvalue we obtain will be truncated out, so that we can be certain that all eigenvalues/eigenvectors that contribute to the computation of \( X_t^+ (\alpha_t) \) have already been obtained. Therefore, we will need a mechanism to adaptively adjust the rank of \( X_t \).

As noted in Lemma 1, we know that the output of approximately solving (BM-nuclear) should be close to the singular vectors of \( \tilde{X}_t \) (up to column-wise scaling). Moreover, when \( X_t \) and \( \tilde{X}_t \) are close to a global optimum \( X^* \), we expect that \( X_{t+1} \) will be close to \( X^* \) and thus also to \( X_t \) and \( \tilde{X}_t \), so the SVD of \( X_{t+1} \) is also expected to be close to that of \( X_t \) and \( \tilde{X}_t \). We therefore use \( U_t \) from (I.4) and \( W_t \) from the output of the BM phase to form

\[
\hat{U}_{t,0} := \text{orth} ([U_t, W_t])
\]

(I.13)
as the base of our the warmstart input to the approximate eigendecomposition in obtaining \( X_{t+1} \) from \( Z_t \). If the BM phase is not entered or does not produce any change in the iterate, we use

\[
\hat{U}_{t,0} := \text{orth} ([U_t, U_{t-1}])
\]

(I.14)

instead. To further guarantee that \( \epsilon_t \to 0 \) in (12), we need to ensure that the rank of \( U_{t,0} \) is sufficiently large, and that \( \hat{U}_{t,0} \) will approach the singular vectors corresponding to the singular values not truncated out. Ideally, we hope that the output of our approximate eigendecomposition will be exactly all the eigenvalues or singular values that are retained nonzero, plus the largest one that is truncated out in (4). Therefore, we add in one column with randomness to \( R_t \) whenever the rank of \( X_t \) and \( X_{t-1} \) are the same (namely, the rank has stopped increasing) and there is at most one eigenvalue truncated out in the inexact proximal gradient step at the \( (t - 1) \)th iteration. The idea is that the case of truncating only one eigenvalue is the ideal scenario we want and we want the next iteration to still have one eigenvalue to truncate as the safeguard, while if no truncation happened, then we should continue increasing the rank. Utilizing this idea, we retain the singular vector \( u \) that corresponds to the largest truncated singular value in the latest iteration where such a truncation took place, and compute its projection \( u_t \) to \( \ker((\hat{U}_{t,0})^\top) \). (It is possible and acceptable that \( u_t = 0 \).) The warmstart input is finally formed by

\[
U_{t,0} := \text{orth} \left( \left[ \hat{U}_{t,0}, u_t + \xi_t \right] \right), \quad \xi_t \in \ker \left( [\hat{U}_{t,0}, u_t]^\top \right), \quad \|\xi_t\| \leq \psi_t,
\]

(I.15)

where \( \xi_t \) is a random vector and \( \{\psi_t\} \) is a sequence such that \( \psi_t \downarrow 0 \).

When the rank of \( X_t \) and \( U_{t-1,0} \) are the same, it means no truncation took place in the proximal operation, and we view this as that the maintained \( u \) has been added to \( U_{t,0} \) as a column in the next iteration when we call LmSVD to compute an approximate eigendecomposition. In this situation, we then seek the eigenvector that corresponds to the
next eigenvalue truncated as our new $u$. When there is no more such vectors available, we simply add in a unit random vector that is orthogonal to the columns of $\hat{U}_t^0$.

Here we provide further explanations to our design above. Assume that the eigenvalues in the exact eigendecomposition of $Z_tZ_t^\top$ are $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m$, $S_1 := \{\sigma_1, \ldots, \sigma_k\}$ is the set of those eigenvalues that will not become zero after the truncation in (I.5), and $S_2 := \{\sigma_{k+1}, \ldots, \sigma_m\}$ are those that will become zero after the truncation in (I.5). To cope with pathological cases in which some eigenvectors corresponding to some $S_3 \subset S_1$ and some corresponding to $S_4 \subset S_2$ are obtained, we inject noise to $u$ so that during the procedure of Algorithm 2, it will approach an eigenvector that corresponds to some value in $S_1 \setminus S_3$ instead of getting stuck at an eigenvector that will be truncated out. (Analysis of the classical SSI suggests that $U_t^i$ approaches the leading eigenvectors as long as no column is exactly a multiple of an eigenvector that corresponds to an eigenvalue with a smaller absolute value.) On the other hand, when we are close to an optimal solution $X^*$ of (CVX), and the eigenvectors corresponding to $S_1$ are all identified or well-approximated, it is natural that we do not want to add in much noise in the initialization of Algorithm 2, as such noise will decelerate the convergence of Algorithm 2. Therefore, in our design, we only add $u$ with noise to $U_t^0$ when $\text{rank}(X_t) = \text{rank}(X_{t-1})$ or $\text{rank}(U_t^{t-1,0}) \leq \text{rank}(X_t) + 1$, namely when no truncation happened or when only one vector is truncated. In the latter case, adding $u$ to $U_t^0$ is for the purpose of making $U_t^0$ contain one eigenvector that is likely to be truncated, so that we can still have the safeguard for ensuring that we have found the correct rank in the approximate eigendecomposition. We then decrease the level of noise by a certain factor whenever $\text{rank}(X_t) = \text{rank}(R_t)$, that is, when exactly only one eigenvalue is truncated. This corresponds to (I.15). For the noise level $\psi_t$, in our implementation, we start with $\psi_0 = 0.1$ so that the noise will not dominate $u_t + \xi_t$, and whenever we need to decrease the significance of the noise, we just let $\psi_t = \psi_{t-1}/2$, and otherwise we assign $\psi_t = \psi_{t-1}$.

I.1.4 Initialization

It can be clearly seen that the point of origin is a saddle point of (BM-nuclear), and this is also the case for many other popular problems that has the form of (BM). Therefore, it is essential to have an effective way to initialize $X_0$, or equivalently $(\hat{W}_0, H_0)$, in Algorithm 1. Existing methods for (BM) usually take a random initialization, but such approaches often lead to an unideal initial objective even worse than using $X_0 = 0$. Moreover, it is hard to decide what is an appropriate value for the initial rank – too large the rank takes longer running time, but too small the rank might lead to slow convergence at the early stage. The most straightforward idea would be to conduct one (inexact) proximal gradient step from the origin, but the difficulty is that we will be in lack of a warmstart matrix for the approximate eigendecomposition in Algorithm 2, and we still need to decide the rank of this matrix.

To get a good initialization for $X_0$, we follow the recent developments in randomized numerical linear algebra by Halko et al. (2011); Martinsson (2019) to combine the HMT method (Halko et al., 2011) with the Nyström method (Nyström, 1930). We imagine that our starting point is actually $X_{-1} = 0$, and then conduct one step of inexact proximal gradient from there with the fixed stepsize $\alpha_{-1} = L^{-1}$ to initialize $X_{-1}$. Regarding the
approximate SVD for
\[ Z_{-1} := X_{-1} - \alpha_{-1} \nabla f(X_{-1}) = -L^{-1} \nabla f(0), \]
we describe how to obtain the eigendecomposition of \( A_{-1} := Z_{-1}Z_{-1}^\top \). Given the initial rank \( k_{-1} \), we start with a random matrix \( \tilde{U}^{-1,0} \in \mathbb{R}^{m \times k_{-1}} \) whose entries are independently and identically distributed as the standard normal distribution. Then we conduct
\[ Q_{-1} = \text{orth} \left( A_{-1} \tilde{U}^{-1,0} \right) \]

as in the HMT method, and use this \( Q_{-1} \) as the sketching matrix in the Nyström method to consider the approximation
\[
A_{-1} \approx A_{-1} Q_{-1} \left( Q_{-1}^\top A_{-1} Q_{-1} \right)^\dagger \left( A_{-1} Q_{-1} \right)^\top := \hat{A}_{-1}, \tag{I.16}
\]
\[ \hat{Q}_{-1} \hat{R}_{-1} = A_{-1} Q_{-1}, \]
where \( \hat{Q}_{-1} \) is an orthonormal matrix and \( \hat{R}_{-1} \) is upper triangular. We then obtain the eigendecomposition of
\[ \hat{R}_{-1}(Q_{-1}^\top A_{-1} Q_{-1})^\dagger \hat{R}_{-1}^\top = \tilde{U}_{-1} \text{diag} \left( (\hat{\Sigma}_{-1})^2 \right) \tilde{U}_{-1}^\top \]
which costs only \( O(k_{-1}^3) \) in both forming the matrix on the left-hand side and calculating its eigendecomposition, as the matrices involved are all \( k_{-1} \) by \( k_{-1} \). When \( k_{-1} \ll m \), this cost is negligible in comparison to other steps. The eigendecomposition of \( \hat{A}_{-1} \) is then obtained by setting \( \hat{U}_0 = \hat{Q}_{-1} \tilde{U}_{-1} \) to get \( \hat{U}_0 \) in the right-hand side of (I.2). The next steps for initializing \( X_0 \) then directly follow the procedure of (I.3)-(I.5).

The remaining issue is to decide the rank \( k_{-1} \) for \( \hat{R}_{-1} \). The most naive idea is to set \( k_{-1} = 1 \) so that all computation in the initialization is of the lowest possible overhead. However, in this case, we will not be able to fully exploit the multicore advantage of modern computing devices. We therefore set the initial \( k_{-1} \) to be the number of cores we can use, so that all the matrix-matrix computation of this step can be fully parallelized without increasing the overhead for initialization.
I.1.5 Solver for (BM-nuclear)

The step of optimizing (BM-nuclear) relies on an off-the-shore solver, and to obtain the best efficiency, the choice should be application-dependent. As we aim for large-scale problems and wish to fully exploit multicore parallelization ubiquitous in modern computers, we will use methods that efficiently utilize multiple computational cores. For (BM-nuclear), as the objective is smooth, many algorithms are available for this choice, ranging from asynchronous, low-order ones to synchronous, high-order approaches. For the low-order ones, we note that since each row of $W$ affects different entries in $WH^\top$, one can update multiple rows simultaneously if $f(X)$ is separable, and the same argument applies to the update of $H$ as well. Therefore, as long as $k$ is no fewer than the number of cores, we should be able to enjoy full parallelism. For problems like (MF), many efficient algorithms such as those by Yu et al. (2014); Zhuang et al. (2013); Wang et al. (2017) are readily available. On the other hand, for (BM-PSD), the property of the constraint set $C$ will limit our choices for the solver. In most applications, however, the constraint $WW^\top \in C$ can be formulated as a smooth manifold $\mathcal{M}$ for $W$, and one can utilize efficient manifold optimization approaches for such problems; see, for example, Absil et al. (2009); Boumal et al. (2014). The major computation in such manifold optimization approaches is usually the computation of the Riemannian gradient and the Hessian-vector products, which are by nature parallelizable.

I.1.6 Parallelism

A key to cope with large-scale problems in modern computing environments is to utilize multicore parallelization. Although inherently the bottleneck of Algorithm 2, (I.6), can be highly parallelized, in practice usually the parallelism is low as this operation is data-heavy but not computation-heavy, so higher parallelism is hindered by the memory bandwidth. On the other hand, solvers for (BM-nuclear) tend to be more computationally intensive, so they can often achieve better parallelism, by utilizing all cores available, than solvers for (CVX). By switching to the BM phase, our algorithm hence exploits better parallelism than state of the art for (CVX). The initial rank $k$ in the previous subsection is also selected to fully exploit the advantage of multiple cores from the beginning on.

I.2 Quadratic SDP

I.2.1 The BM formulations for the RKE problem and the molecular conformation problem.

For the RKE problem and the molecular conformation problem, the corresponding factorized problem is given as

$$
\min_{W \in \mathbb{R}^{n \times k}} \left( g(W) := \frac{1}{2} \sum_{(i,j) \in \Omega} w_{ij} \left( \langle E_{ij}, WW^\top \rangle - d_{ij}^2 \right)^2 + \lambda \|W\|_F^2 \right), \quad \text{s.t.} \quad W^\top e = 0. \quad (I.17)
$$

The gradient of this function $g : \mathbb{R}^{n \times k} \to \mathbb{R}$ is

$$
\nabla g(W) = 2 \left( \sum_{(i,j) \in \Omega} w_{ij} \left( \langle E_{ij}, WW^\top \rangle - d_{ij}^2 \right) E_{ij} + \lambda I \right) W.
$$
Algorithm 3: The APG method for solving (QSDP)

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{input}: Initial point: \( Y_0 = X_0 \in S^n \), Lipschitz constant \( L > 0 \) and \( \beta_0 = 1 \)
\For {\( t = 0, \ldots \), do}
\State \( X_{t+1} \leftarrow P_X(Y_t - L^{-1}\nabla f(Y_t)) \)
\State \( \beta_{t+1} \leftarrow \frac{1 + \sqrt{1 + 4\beta_t^2}}{2} \)
\State \( Y_{t+1} \leftarrow X_{t+1} + \frac{\beta_{t-1}}{\beta_{t+1}}(X_{t+1} - X_t) \)
\EndFor
\State \textbf{output}: \( X_{t+1} \)
\end{algorithmic}
\end{algorithm}

For any given \( D \in \mathbb{R}^{n \times k} \), the Hessian operator of \( g \) performed on \( H \) can be computed as

\[ \nabla^2 g(W)[D] = 2 \sum_{(i,j) \in \Omega} w_{ij} \langle E_{ij}, WD^T + DW^T \rangle E_{ij} W + 2 \sum_{(i,j) \in \Omega} w_{ij} \left( \langle E_{ij}, WW^T \rangle - d_{ij}^2 \right) E_{ij} D + 2\lambda D. \]

\textbf{I.2.2 Implementation Details}

Recall that for a given problem size \( n \), Algorithm 1 iterates on a low-rank matrix \( W_t \in \mathbb{R}^{n \times k_t} \) with \( k_t < n \) dynamically adjusted. In our experiment, we choose the initial \( k_0 \) as

\[ k_0 = \min\{100, \lfloor 0.15n \rfloor \}. \]

Given \( k_0 \), we then run APG for 100 iterations with this rank constraint to generate an initial point \( W_0 \). When applying the APG method, we never form the matrix \( X := WW^T \) and use only eigendecompositions with rank \( k_0 \). In particular, we use MATLAB’s built-in \texttt{eigs} subroutine to compute the largest \( k \) eigenpairs. After the initialization stage, at the \( t \)-th iteration of Algorithm 1, we first apply the Manopt solver to solve the factorized problem (I.17) with the initial point \( W_t \) to compute a matrix \( \tilde{W}_t \). Using \( \tilde{W}_t \), we then perform one inexact PG step with an approximate eigendecomposition with fixed step size \( \alpha_t = 1/L \). If all the computed eigenvalues are positive, then it is highly possible that the current rank is too small. In this case, we set \( k_{t+1} \leftarrow k_t + \lfloor \frac{n}{20} \rfloor \). Otherwise, we set \( k_{t+1} \) as the number of positive eigenvalues returned by \texttt{eigs}.

\textbf{I.2.3 The accelerated proximal gradient method}

Since we utilize the APG method for the initialization of BM-Global for (QSDP), for completeness, we provide a concise description of the APG method in Algorithm 3. To perform the projection \( P_X \) for (QSDP) through Lemma 7, we use the \texttt{eig} subroutine in MATLAB. (Note that the PG method is equivalent to setting \( \beta_t \equiv 1 \) in Algorithm 3)

\textbf{II. Additional Experimental Details and More Experiments}

\textbf{II.1 The SpaRSA variant for Matrix Completion}

It is known that the convergence speed of standard PG could be slow, thus we also considered following Wright et al. (2009) to use the Barzilai-Borwein (BB) initialization (Barzilai and
Supp: Nuclear-norm regularized low-rank optimization through BM decomposition

Borwein, 1988) with linesearch to decide the step size to accelerate the practical convergence of the convex lifting step. Given \( \alpha_{\text{max}} \geq \alpha_{\text{min}} > 0 \), we first compute

\[
\alpha_{\text{BB}}^t := \max \left\{ \alpha_{\text{min}}, \min \left\{ \alpha_{\text{max}}, \frac{\langle X_t - X_{t-1}, \nabla f(X_t) - \nabla f(X_{t-1}) \rangle}{\|X_t - X_{t-1}\|_F^2} \right\} \right\} .
\]

(II.1)

But different from Wright et al. (2009), our backtracking linesearch does not search for an \( \alpha_t \) that gives sufficient descent. Instead, given \( \beta, \delta \in (0, 1) \), we find the smallest nonnegative integer \( i \) such that \( \alpha_t = \alpha_{\text{BB}}^t \beta^i \) satisfies (13) or (14).

In our scenario, we only have the factorized form \( X_t = W_t H_t^\top \) for each \( t \) but not the iterate \( X_t \) itself, so we use the following formula to calculate the numerator of (II.1), which is also used as the last term in (13)

\[
\|X_t - X_{t-1}\|_F^2 = \langle X_t - X_{t-1}, X_t - X_{t-1} \rangle = \langle X_t, X_t \rangle + \langle X_{t-1}, X_{t-1} \rangle - 2\langle X_t, X_{t-1} \rangle
\]

\[
= \text{tr} \left( (W_t^\top W_t)(H_t^\top H_t) \right) + \text{tr} \left( (W_{t-1}^\top W_{t-1})(H_{t-1}^\top H_{t-1}) \right) + \text{tr} \left( (W_{t-1}^\top W_t)(H_{t-1}^\top H_t) \right).
\]

(II.2)

Similar calculation is also applied to compute the denominator of (II.1). If rank(\( X_t \)) = \( k_t \) and rank(\( X_{t-1} \)) = \( k_{t-1} \), the cost of computing (II.2) is \( O((m + n)(k_t^2 + k_{t-1}^2 + k_t k_{t-1})) \).

II.2 Experimental Details of Section 6.1.1

We report the required running time for making the relative objective smaller than \( \epsilon \in \{10^{-4}, 10^{-8}, 10^{-12}\} \) for both the fixed-step variant and the SpaRSA variant of our algorithm. The algorithms are terminated either when they have reached \( \epsilon = 10^{-12} \) or when they have conducted 100 inexact proximal gradient steps. The results are shown in Tables (IV) and (V).

Aside from the selection of \( x = 1 \) and \( y = 3 \) made in the main paper for the fixed-step variant, for SpaRSA, we also see that \( x = 1 \) and \( y = 3 \) is still robust and we thus use this setting for SpaRSA from now on.

II.3 Comparison with the SpaRSA variant for Matrix Completion

In this subsection, we present experimental results with the SpaRSA variant. The results are shown in Fig. (IV). As expected, this variant is faster than the fixed-step variant in terms of iterations, although the difference is minor. On the other hand, it is significantly slower in terms of the real running time. The likely reason is the possible additional approximate SVDs executed in the backtracking procedure, while the inexact PG step in our algorithm is not the major contributor to the objective decrease, so the improvement in the convergence speed of this part cannot counterbalance the corresponding additional cost.

II.4 Parallelism

We also examine the parallelism of our method. In particular, we run our method with 1, 2, 4, 8 cores and check how much time they respectively take to make (74) no larger than
Table (IV): Time (seconds) required for solving (MC) on toy-example using Algorithm 1 to make (74) no larger than $\epsilon$. We alternate between $x$ consecutive inexact proximal gradient steps and $y$ consecutive iterations of the BM phase solver. The fastest one for each case is labeled in red. "-" indicates that the designated $\epsilon$ is not reached within 100 inexact proximal gradient steps.

<table>
<thead>
<tr>
<th>$\epsilon = 10^{-4}$</th>
<th>$y$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x = 1$</td>
<td>5.4e+0</td>
<td>1.1e+0</td>
<td>1.2e+0</td>
<td>1.4e+0</td>
<td>1.5e+0</td>
<td>1.6e+0</td>
<td>1.7e+0</td>
<td>1.8e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 2$</td>
<td>1.4e+0</td>
<td>1.3e+0</td>
<td>1.8e+0</td>
<td>1.7e+0</td>
<td>1.9e+0</td>
<td>2.1e+0</td>
<td>2.4e+0</td>
<td>2.6e+0</td>
</tr>
<tr>
<td>SpaRSA</td>
<td>$x = 3$</td>
<td>1.1e+0</td>
<td>1.1e+0</td>
<td>1.1e+0</td>
<td>1.2e+0</td>
<td>1.2e+0</td>
<td>1.2e+0</td>
<td>1.3e+0</td>
<td>1.3e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 4$</td>
<td>1.3e+0</td>
<td>1.0e+0</td>
<td>1.1e+0</td>
<td>1.2e+0</td>
<td>1.6e+0</td>
<td>1.7e+0</td>
<td>1.8e+0</td>
<td>1.8e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 5$</td>
<td>1.0e+0</td>
<td>1.1e+0</td>
<td>1.3e+0</td>
<td>1.3e+0</td>
<td>1.4e+0</td>
<td>1.5e+0</td>
<td>1.6e+0</td>
<td>1.7e+0</td>
</tr>
<tr>
<td>$\epsilon = 10^{-8}$</td>
<td>$x = 1$</td>
<td>1.2e+0</td>
<td>1.0e+0</td>
<td>1.2e+0</td>
<td>1.3e+0</td>
<td>1.4e+0</td>
<td>1.6e+0</td>
<td>1.7e+0</td>
<td>2.0e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 2$</td>
<td>1.2e+0</td>
<td>1.5e+0</td>
<td>1.5e+0</td>
<td>1.7e+0</td>
<td>1.9e+0</td>
<td>2.1e+0</td>
<td>2.3e+0</td>
<td>2.5e+0</td>
</tr>
<tr>
<td>SpaRSA</td>
<td>$x = 3$</td>
<td>1.5e+0</td>
<td>1.2e+0</td>
<td>1.1e+0</td>
<td>1.2e+0</td>
<td>1.2e+0</td>
<td>1.3e+0</td>
<td>1.3e+0</td>
<td>1.4e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 4$</td>
<td>1.2e+0</td>
<td>1.3e+0</td>
<td>1.3e+0</td>
<td>1.4e+0</td>
<td>1.5e+0</td>
<td>1.6e+0</td>
<td>1.7e+0</td>
<td>1.7e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 5$</td>
<td>1.3e+0</td>
<td>1.2e+0</td>
<td>1.3e+0</td>
<td>1.4e+0</td>
<td>1.5e+0</td>
<td>1.6e+0</td>
<td>1.8e+0</td>
<td>1.9e+0</td>
</tr>
<tr>
<td>$\epsilon = 10^{-12}$</td>
<td>$x = 1$</td>
<td>4.4e+0</td>
<td>3.7e+0</td>
<td>4.9e+0</td>
<td>5.0e+0</td>
<td>5.5e+0</td>
<td>5.8e+0</td>
<td>7.0e+0</td>
<td>7.1e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 2$</td>
<td>4.4e+0</td>
<td>3.8e+0</td>
<td>4.2e+0</td>
<td>4.6e+0</td>
<td>5.1e+0</td>
<td>5.2e+0</td>
<td>5.6e+0</td>
<td>5.7e+0</td>
</tr>
<tr>
<td>SpaRSA</td>
<td>$x = 3$</td>
<td>4.4e+0</td>
<td>3.8e+0</td>
<td>4.1e+0</td>
<td>4.6e+0</td>
<td>4.9e+0</td>
<td>5.2e+0</td>
<td>5.6e+0</td>
<td>6.0e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 4$</td>
<td>5.0e+0</td>
<td>3.6e+0</td>
<td>3.6e+0</td>
<td>4.0e+0</td>
<td>4.3e+0</td>
<td>4.5e+0</td>
<td>4.7e+0</td>
<td>4.7e+0</td>
</tr>
<tr>
<td></td>
<td>$x = 5$</td>
<td>4.9e+0</td>
<td>4.1e+0</td>
<td>4.2e+0</td>
<td>4.5e+0</td>
<td>4.8e+0</td>
<td>5.1e+0</td>
<td>5.3e+0</td>
<td>5.3e+0</td>
</tr>
</tbody>
</table>

| $\alpha_t \equiv 1$ | $x = 1$ | 1.6e+1 | 7.4e+0 | 8.1e+0 | 7.9e+0 | 9.6e+0 | 9.1e+0 | 1.0e+1 | 1.2e+1 |
|----------------------| $x = 2$ | 1.1e+1 | 8.9e+0 | 7.7e+0 | 8.2e+0 | 8.4e+0 | 9.4e+0 | 9.2e+0 | 9.0e+0 |
| SpaRSA               | $x = 3$ | 1.0e+1 | 7.0e+0 | 7.2e+0 | 7.2e+0 | 7.1e+0 | 7.3e+0 | 7.8e+0 | 8.6e+0 |
|                      | $x = 4$ | 9.0e+0 | 7.6e+0 | 7.1e+0 | 7.0e+0 | 7.4e+0 | 7.9e+0 | 7.6e+0 | 7.5e+0 |
|                      | $x = 5$ | 1.1e+1 | 7.8e+0 | 6.2e+0 | 6.8e+0 | 6.9e+0 | 7.3e+0 | 6.9e+0 | 7.9e+0 |

| $\epsilon = 10^{-12}$| $x = 1$ | 1.2e+1 | 6.8e+0 | 8.0e+0 | 7.7e+0 | 8.8e+0 | 9.3e+0 | 1.0e+1 | 1.0e+1 |
|                      | $x = 2$ | 1.7e+1 | 6.9e+0 | 7.1e+0 | 7.1e+0 | 7.8e+0 | 7.8e+0 | 8.4e+0 | 8.7e+0 |
| SpaRSA               | $x = 3$ | 1.4e+1 | 7.2e+0 | 6.9e+0 | 7.9e+0 | 7.8e+0 | 8.0e+0 | 8.6e+0 | 8.5e+0 |
|                      | $x = 4$ | 1.3e+1 | 7.7e+0 | 6.9e+0 | 7.4e+0 | 7.7e+0 | 8.0e+0 | 8.6e+0 | 8.8e+0 |
|                      | $x = 5$ | 1.7e+1 | 7.6e+0 | 6.5e+0 | 7.1e+0 | 7.4e+0 | 7.9e+0 | 8.4e+0 | 8.3e+0 |
Table (V): Time (seconds) required for solving (MC) on movielens100k using Algorithm 1 to make (74) no larger than $\epsilon$. We alternate between $x$ consecutive inexact proximal gradient steps and $y$ consecutive iterations of the BM phase solver. The fastest one for each case is labeled in red. "-" indicates that the designated $\epsilon$ is not reached within 100 inexact proximal gradient steps.

\[
\epsilon = 10^{-4}
\]

<table>
<thead>
<tr>
<th>$y$</th>
<th>$x = 1$</th>
<th>$x = 2$</th>
<th>$x = 3$</th>
<th>$x = 4$</th>
<th>$x = 5$</th>
<th>SpaRSA $x = 1$</th>
<th>SpaRSA $x = 2$</th>
<th>SpaRSA $x = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$5.7e-1$</td>
<td>$5.2e-1$</td>
<td>$4.1e-1$</td>
<td>$4.6e-1$</td>
<td>$5.1e-1$</td>
<td>$5.4e-1$</td>
<td>$5.7e-1$</td>
<td>$6.0e-1$</td>
</tr>
<tr>
<td>$x = 2$</td>
<td>$2.0e+0$</td>
<td>$1.4e+0$</td>
<td>$7.0e-1$</td>
<td>$6.8e-1$</td>
<td>$5.7e-1$</td>
<td>$5.6e-1$</td>
<td>$4.9e-1$</td>
<td>$9.5e-1$</td>
</tr>
<tr>
<td>$x = 3$</td>
<td>$3.8e+0$</td>
<td>$2.4e+0$</td>
<td>$1.6e+0$</td>
<td>$1.6e+0$</td>
<td>$9.1e-1$</td>
<td>$9.3e-1$</td>
<td>$8.0e-1$</td>
<td>$5.8e-1$</td>
</tr>
<tr>
<td>$x = 4$</td>
<td>$4.1e+0$</td>
<td>$2.9e+0$</td>
<td>$2.3e+0$</td>
<td>$1.7e+0$</td>
<td>$1.3e+0$</td>
<td>$1.3e+0$</td>
<td>$7.7e-1$</td>
<td>$7.4e-1$</td>
</tr>
<tr>
<td>$x = 5$</td>
<td>$3.9e+0$</td>
<td>$2.6e+0$</td>
<td>$2.1e+0$</td>
<td>$1.5e+0$</td>
<td>$1.5e+0$</td>
<td>$1.4e+0$</td>
<td>$1.1e+0$</td>
<td>$9.7e-1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha_t \equiv 1$</td>
<td>$x = 1$</td>
<td>$7.8e-1$</td>
<td>$4.5e-1$</td>
<td>$4.4e-1$</td>
<td>$4.3e-1$</td>
<td>$5.3e-1$</td>
<td>$5.7e-1$</td>
<td>$6.0e-1$</td>
</tr>
<tr>
<td></td>
<td>$x = 2$</td>
<td>$1.6e+0$</td>
<td>$8.0e-1$</td>
<td>$5.7e-1$</td>
<td>$5.2e-1$</td>
<td>$4.5e-1$</td>
<td>$4.9e-1$</td>
<td>$5.2e-1$</td>
</tr>
<tr>
<td></td>
<td>$x = 3$</td>
<td>$2.2e+0$</td>
<td>$1.4e+0$</td>
<td>$1.0e+0$</td>
<td>$1.1e+0$</td>
<td>$8.1e-1$</td>
<td>$7.7e-1$</td>
<td>$5.3e-1$</td>
</tr>
<tr>
<td></td>
<td>$x = 4$</td>
<td>$2.5e+0$</td>
<td>$1.6e+0$</td>
<td>$1.3e+0$</td>
<td>$1.3e+0$</td>
<td>$9.4e-1$</td>
<td>$6.6e-1$</td>
<td>$6.3e-1$</td>
</tr>
<tr>
<td></td>
<td>$x = 5$</td>
<td>$2.8e+0$</td>
<td>$1.9e+0$</td>
<td>$1.6e+0$</td>
<td>$1.4e+0$</td>
<td>$1.1e+0$</td>
<td>$1.1e+0$</td>
<td>$8.2e-1$</td>
</tr>
</tbody>
</table>

\[
\epsilon = 10^{-8}
\]

<table>
<thead>
<tr>
<th>$y$</th>
<th>$x = 1$</th>
<th>$x = 2$</th>
<th>$x = 3$</th>
<th>$x = 4$</th>
<th>$x = 5$</th>
<th>SpaRSA $x = 1$</th>
<th>SpaRSA $x = 2$</th>
<th>SpaRSA $x = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$2.4e+0$</td>
<td>$2.2e+0$</td>
<td>$1.9e+0$</td>
<td>$1.7e+0$</td>
<td>$1.6e+0$</td>
<td>$1.6e+0$</td>
<td>$1.7e+0$</td>
<td>$1.7e+0$</td>
</tr>
<tr>
<td>$x = 2$</td>
<td>$4.6e+0$</td>
<td>$4.2e+0$</td>
<td>$3.2e+0$</td>
<td>$2.4e+0$</td>
<td>$2.1e+0$</td>
<td>$1.8e+0$</td>
<td>$1.7e+0$</td>
<td>$1.6e+0$</td>
</tr>
<tr>
<td>$x = 3$</td>
<td>$7.0e+0$</td>
<td>$5.2e+0$</td>
<td>$4.4e+0$</td>
<td>$3.5e+0$</td>
<td>$2.9e+0$</td>
<td>$2.7e+0$</td>
<td>$2.7e+0$</td>
<td>$2.4e+0$</td>
</tr>
<tr>
<td>$x = 4$</td>
<td>$7.6e+0$</td>
<td>$5.9e+0$</td>
<td>$4.8e+0$</td>
<td>$4.0e+0$</td>
<td>$3.5e+0$</td>
<td>$3.1e+0$</td>
<td>$3.0e+0$</td>
<td>$2.6e+0$</td>
</tr>
<tr>
<td>$x = 5$</td>
<td>$7.2e+0$</td>
<td>$5.8e+0$</td>
<td>$4.6e+0$</td>
<td>$3.6e+0$</td>
<td>$3.4e+0$</td>
<td>$3.3e+0$</td>
<td>$2.9e+0$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha_t \equiv 1$</td>
<td>$x = 1$</td>
<td>$2.4e+0$</td>
<td>$2.0e+0$</td>
<td>$1.7e+0$</td>
<td>$1.6e+0$</td>
<td>$1.6e+0$</td>
<td>$1.8e+0$</td>
<td>$2.0e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 2$</td>
<td>$3.4e+0$</td>
<td>$2.7e+0$</td>
<td>$2.1e+0$</td>
<td>$1.9e+0$</td>
<td>$1.8e+0$</td>
<td>$1.7e+0$</td>
<td>$1.7e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 3$</td>
<td>$4.3e+0$</td>
<td>$3.4e+0$</td>
<td>$2.8e+0$</td>
<td>$2.7e+0$</td>
<td>$2.3e+0$</td>
<td>$2.1e+0$</td>
<td>$1.9e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 4$</td>
<td>$4.9e+0$</td>
<td>$3.8e+0$</td>
<td>$3.2e+0$</td>
<td>$3.0e+0$</td>
<td>$2.6e+0$</td>
<td>$2.4e+0$</td>
<td>$2.2e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 5$</td>
<td>$-5.2e+0$</td>
<td>$4.3e+0$</td>
<td>$4.3e+0$</td>
<td>$3.7e+0$</td>
<td>$3.6e+0$</td>
<td>$3.4e+0$</td>
<td>$3.3e+0$</td>
</tr>
</tbody>
</table>

\[
\epsilon = 10^{-12}
\]

<table>
<thead>
<tr>
<th>$y$</th>
<th>$x = 1$</th>
<th>$x = 2$</th>
<th>$x = 3$</th>
<th>$x = 4$</th>
<th>$x = 5$</th>
<th>SpaRSA $x = 1$</th>
<th>SpaRSA $x = 2$</th>
<th>SpaRSA $x = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$4.6e+0$</td>
<td>$3.8e+0$</td>
<td>$3.5e+0$</td>
<td>$3.3e+0$</td>
<td>$3.4e+0$</td>
<td>$3.2e+0$</td>
<td>$2.9e+0$</td>
<td>$2.9e+0$</td>
</tr>
<tr>
<td>$x = 2$</td>
<td>$7.0e+0$</td>
<td>$6.6e+0$</td>
<td>$5.3e+0$</td>
<td>$3.3e+0$</td>
<td>$3.4e+0$</td>
<td>$3.2e+0$</td>
<td>$3.0e+0$</td>
<td>$3.0e+0$</td>
</tr>
<tr>
<td>$x = 3$</td>
<td>$9.9e+0$</td>
<td>$7.5e+0$</td>
<td>$6.8e+0$</td>
<td>$5.4e+0$</td>
<td>$4.6e+0$</td>
<td>$4.5e+0$</td>
<td>$4.3e+0$</td>
<td>$4.0e+0$</td>
</tr>
<tr>
<td>$x = 4$</td>
<td>$-5.8e+0$</td>
<td>$7.3e+0$</td>
<td>$6.2e+0$</td>
<td>$5.4e+0$</td>
<td>$5.1e+0$</td>
<td>$4.7e+0$</td>
<td>$4.2e+0$</td>
<td>$4.2e+0$</td>
</tr>
<tr>
<td>$x = 5$</td>
<td>$-8.8e+0$</td>
<td>$7.3e+0$</td>
<td>$6.2e+0$</td>
<td>$5.6e+0$</td>
<td>$5.5e+0$</td>
<td>$5.0e+0$</td>
<td>$4.6e+0$</td>
<td>$4.6e+0$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha_t \equiv 1$</td>
<td>$x = 1$</td>
<td>$5.5e+0$</td>
<td>$3.7e+0$</td>
<td>$3.1e+0$</td>
<td>$3.0e+0$</td>
<td>$3.0e+0$</td>
<td>$3.1e+0$</td>
<td>$3.4e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 2$</td>
<td>$5.7e+0$</td>
<td>$4.5e+0$</td>
<td>$3.6e+0$</td>
<td>$3.4e+0$</td>
<td>$3.2e+0$</td>
<td>$3.1e+0$</td>
<td>$3.0e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 3$</td>
<td>$-5.2e+0$</td>
<td>$4.3e+0$</td>
<td>$4.3e+0$</td>
<td>$3.7e+0$</td>
<td>$3.6e+0$</td>
<td>$3.4e+0$</td>
<td>$3.3e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 4$</td>
<td>$-5.8e+0$</td>
<td>$4.8e+0$</td>
<td>$4.5e+0$</td>
<td>$4.1e+0$</td>
<td>$3.9e+0$</td>
<td>$3.7e+0$</td>
<td>$3.5e+0$</td>
</tr>
<tr>
<td></td>
<td>$x = 5$</td>
<td>$-5.3e+0$</td>
<td>$4.9e+0$</td>
<td>$4.4e+0$</td>
<td>$4.2e+0$</td>
<td>$3.9e+0$</td>
<td>$3.7e+0$</td>
<td>$3.7e+0$</td>
</tr>
</tbody>
</table>
Figure (IV): Performance of the SpaRSA variant of BM-Global. Top row: iterations v.s. relative objective. Bottom row: running time (seconds) v.s. relative objective.

Figure (V): Comparison of different algorithms in terms of speedup with respect to different number of cores.

We can observe from Fig. (V) that except for the small dataset movielens100k, PolyMF-SS always achieves the highest speedup because most of its operations are inherently parallel and computationally heavy, and BM-Global comes the next, while AIS-Impute and Active-ALT barely exhibit any parallelism at all.

II.5 Numerical results for the APG method

This subsection presents numerical results of the APG method Algorithm 3 on testing problems considered in Section 6.2
In our experiments, the APG method is executed with $Y_0 = X_0 = 0$ and a fixed step size $\alpha = 1/L$, where $L := \|AA\|_2$ is estimated by the \texttt{eigs} subroutine in MATLAB. To perform the projection $P_X$ through Lemma 7, we use the \texttt{eig} subroutine in MATLAB. Our stopping condition for the APG method is

$$\frac{|f(X_{t+1}) - f(X_t)|}{1 + |f(X_t)|} \leq \text{tol},$$

where \texttt{tol} is the tolerance (chosen as $10^{-6}$ in our experiments). Moreover, we set the maximal number of iterations to be 10000 and the maximal computational time to be four hours. The computational results for the RKE problems and the molecular conformation problems are presented in Tables (VI) and (VII), respectively. We can observe from the presented results that the APG method usually performs worse than BM-Global and QSDPNAL. Indeed, it returns solutions with a lower accuracy and takes more computational time. Also, the numerical results suggest that the APG method could be sensitive to the sign of the parameter $\lambda$, since it performs much better in the cases of $\lambda > 0$ then in the cases of $\lambda < 0$.

<table>
<thead>
<tr>
<th>Name</th>
<th>$n$</th>
<th>$\eta_{\text{prim}}$</th>
<th>$\eta_{\text{opt}}$</th>
<th>$\text{rank}$</th>
<th>Time</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrainMRI</td>
<td>124</td>
<td>2e-16</td>
<td>1e-03</td>
<td>5</td>
<td>0.7</td>
<td>86</td>
</tr>
<tr>
<td>protein</td>
<td>213</td>
<td>3e-15</td>
<td>1e-03</td>
<td>27</td>
<td>6.7</td>
<td>285</td>
</tr>
<tr>
<td>CoilDelftDiff</td>
<td>288</td>
<td>2e-15</td>
<td>1e-03</td>
<td>32</td>
<td>9.0</td>
<td>234</td>
</tr>
<tr>
<td>coildelftsame</td>
<td>288</td>
<td>3e-15</td>
<td>2e-03</td>
<td>32</td>
<td>7.7</td>
<td>189</td>
</tr>
<tr>
<td>CoilYork</td>
<td>288</td>
<td>1e-15</td>
<td>1e-03</td>
<td>21</td>
<td>9.2</td>
<td>231</td>
</tr>
<tr>
<td>Chickenpieces-5-45</td>
<td>446</td>
<td>4e-16</td>
<td>2e-03</td>
<td>28</td>
<td>32.6</td>
<td>288</td>
</tr>
<tr>
<td>newgroups</td>
<td>600</td>
<td>3e-17</td>
<td>2e-03</td>
<td>84</td>
<td>69.7</td>
<td>341</td>
</tr>
<tr>
<td>flowcytodis</td>
<td>612</td>
<td>8e-16</td>
<td>1e-03</td>
<td>16</td>
<td>84.6</td>
<td>353</td>
</tr>
<tr>
<td>DelftPedestrians</td>
<td>689</td>
<td>2e-15</td>
<td>2e-03</td>
<td>70</td>
<td>105.0</td>
<td>377</td>
</tr>
<tr>
<td>WoodyPlants50</td>
<td>791</td>
<td>8e-16</td>
<td>2e-03</td>
<td>49</td>
<td>185.7</td>
<td>478</td>
</tr>
<tr>
<td>delftgestures</td>
<td>1500</td>
<td>1e-15</td>
<td>1e-02</td>
<td>106</td>
<td>283.3</td>
<td>191</td>
</tr>
<tr>
<td>zongker</td>
<td>2000</td>
<td>1e-15</td>
<td>4e-03</td>
<td>303</td>
<td>1236.5</td>
<td>451</td>
</tr>
<tr>
<td>polydish57</td>
<td>4000</td>
<td>2e-15</td>
<td>1e-02</td>
<td>139</td>
<td>8530.7</td>
<td>357</td>
</tr>
<tr>
<td>polydism57</td>
<td>4000</td>
<td>3e-15</td>
<td>1e-02</td>
<td>83</td>
<td>8023.2</td>
<td>337</td>
</tr>
</tbody>
</table>

Table (VI): Computational results for the APG method on RKE problems.

References


Table (VII): Computational results for the APG method on molecular conformation problems. "-" indicates that the solver is terminated because the maximum running time of four hours is reached.


<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>$\eta_{\text{prim}}$</th>
<th>$\eta_{\text{opt}}$</th>
<th>$\text{rnk}$</th>
<th>RMSD</th>
<th>Time</th>
<th>Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1PBM</td>
<td>126</td>
<td>4e-16</td>
<td>6e-06</td>
<td>14</td>
<td>7.4</td>
<td>30.1</td>
<td>5073</td>
</tr>
<tr>
<td>1AU6</td>
<td>161</td>
<td>2e-16</td>
<td>6e-06</td>
<td>15</td>
<td>6.8</td>
<td>55.9</td>
<td>5250</td>
</tr>
<tr>
<td>1PTQ</td>
<td>402</td>
<td>2e-15</td>
<td>2e-06</td>
<td>20</td>
<td>10.2</td>
<td>727.8</td>
<td>10000</td>
</tr>
<tr>
<td>1CTF</td>
<td>487</td>
<td>5e-16</td>
<td>2e-06</td>
<td>23</td>
<td>11.2</td>
<td>1250.5</td>
<td>10000</td>
</tr>
<tr>
<td>1HOE</td>
<td>558</td>
<td>1e-15</td>
<td>2e-06</td>
<td>24</td>
<td>11.6</td>
<td>1831.3</td>
<td>10000</td>
</tr>
<tr>
<td>1LFB</td>
<td>641</td>
<td>1e-15</td>
<td>2e-06</td>
<td>28</td>
<td>13.4</td>
<td>2379.5</td>
<td>10000</td>
</tr>
<tr>
<td>1PHT</td>
<td>666</td>
<td>5e-16</td>
<td>2e-06</td>
<td>28</td>
<td>12.2</td>
<td>2598.5</td>
<td>10000</td>
</tr>
<tr>
<td>1F39</td>
<td>767</td>
<td>1e-15</td>
<td>2e-06</td>
<td>33</td>
<td>13.6</td>
<td>3432.6</td>
<td>10000</td>
</tr>
<tr>
<td>1DCH</td>
<td>806</td>
<td>3e-17</td>
<td>2e-06</td>
<td>32</td>
<td>13.4</td>
<td>3776.6</td>
<td>10000</td>
</tr>
<tr>
<td>1HQQ</td>
<td>891</td>
<td>1e-15</td>
<td>2e-06</td>
<td>41</td>
<td>15.0</td>
<td>4539.2</td>
<td>10000</td>
</tr>
<tr>
<td>1POA</td>
<td>914</td>
<td>9e-16</td>
<td>2e-06</td>
<td>37</td>
<td>14.2</td>
<td>4844.0</td>
<td>10000</td>
</tr>
<tr>
<td>1AX8</td>
<td>1003</td>
<td>8e-16</td>
<td>2e-06</td>
<td>40</td>
<td>14.3</td>
<td>5969.8</td>
<td>10000</td>
</tr>
<tr>
<td>1TJO</td>
<td>1394</td>
<td>2e-15</td>
<td>2e-06</td>
<td>62</td>
<td>19.0</td>
<td>12512.3</td>
<td>10000</td>
</tr>
<tr>
<td>1RGS</td>
<td>2015</td>
<td>6e-16</td>
<td>4e-06</td>
<td>126</td>
<td>20.2</td>
<td>-</td>
<td>4952</td>
</tr>
<tr>
<td>1TOA</td>
<td>2138</td>
<td>2e-15</td>
<td>5e-06</td>
<td>136</td>
<td>19.2</td>
<td>-</td>
<td>4303</td>
</tr>
<tr>
<td>1KDH</td>
<td>2846</td>
<td>5e-16</td>
<td>1e-05</td>
<td>254</td>
<td>21.9</td>
<td>-</td>
<td>1901</td>
</tr>
<tr>
<td>1NFG</td>
<td>3501</td>
<td>3e-15</td>
<td>5e-05</td>
<td>391</td>
<td>21.2</td>
<td>-</td>
<td>903</td>
</tr>
<tr>
<td>1BPM</td>
<td>3672</td>
<td>2e-16</td>
<td>6e-05</td>
<td>436</td>
<td>23.8</td>
<td>-</td>
<td>788</td>
</tr>
<tr>
<td>1MQQ</td>
<td>5510</td>
<td>7e-16</td>
<td>3e-04</td>
<td>888</td>
<td>26.0</td>
<td>-</td>
<td>247</td>
</tr>
</tbody>
</table>