SCALABLE ROBUST MATRIX RECOVERY: FRANK-WOLFE MEETS PROXIMAL METHODS
CUN MU*, YUQIAN ZHANG†, JOHN WRIGHT†, AND DONALD GOLDFARB*

Abstract. Recovering matrices from compressive and grossly corrupted observations is a fundamental problem in robust statistics, with rich applications in computer vision and machine learning. In theory, under certain conditions, this problem can be solved in polynomial time via a natural convex relaxation, known as Compressive Principal Component Pursuit (CPCP). However, many existing provably convergent algorithms for CPCP suffer from superlinear per-iteration cost, which severely limits their applicability to large scale problems. In this paper, we propose provably convergent, scalable and efficient methods to solve CPCP with (essentially) linear per-iteration cost. Our method combines classical ideas from Frank-Wolfe and proximal methods. In each iteration, we mainly exploit Frank-Wolfe to update the low-rank component with rank-one SVD and exploit the proximal step for the sparse term. Convergence results and implementation details are discussed. We demonstrate the practicability and scalability of our approach with numerical experiments on visual data.

Key words. robust matrix recovery, compressive principal component pursuit, Frank-Wolfe, conditional gradient, proximal methods, scalability

AMS subject classifications. 90C06, 90C25, 90C52

1. Introduction. Suppose that a matrix $M_0 \in \mathbb{R}^{m \times n}$ is of the form $M_0 = L_0 + S_0 + N_0$, where $L_0$ is a low-rank matrix, $S_0$ is a sparse error matrix, and $N_0$ is a dense noise matrix. Linear measurements

$$ b = A[M_0] = (\langle A_1, M_0 \rangle, \langle A_2, M_0 \rangle, \ldots, \langle A_p, M_0 \rangle) \top \in \mathbb{R}^p $$

are collected, where $A : \mathbb{R}^{m \times n} \to \mathbb{R}^p$ is the sensing operator, $A_k$ is the sensing matrix for the $k$-th measurement and $\langle A_k, M_0 \rangle = \text{Tr}(A_k M_0)$. Can we, in a tractable way, recover $L_0$ and $S_0$ from $b$, given $A$?

One natural approach is to solve the optimization

$$ \min_{L, S} \frac{1}{2} \|b - A[L + S]\|_2^2 + \lambda_L \text{rank}(L) + \lambda_S \|S\|_0. $$

Here, $\lambda_L$ and $\lambda_S$ are regularization parameters, and $\|S\|_0$ denotes the number of nonzero entries in $S$. Unfortunately, problem (1.2) is nonconvex, and hence is not directly tractable. However, by replacing the $\ell_0$ norm $\|S\|_0$ with the $\ell_1$ norm $\|S\|_1 = \sum_{i=1}^m \sum_{j=1}^n |S_{ij}|$, and replacing the rank $\text{rank}(L)$ with the nuclear norm $\|L\|_*$ (i.e., the sum of the singular values of $L$), we obtain a natural, tractable, convex relaxation of (1.2),

$$ \min_{L, S} \frac{1}{2} \|b - A[L + S]\|_2^2 + \lambda_L \|L\|_* + \lambda_S \|S\|_1. $$

This optimization is sometimes referred to as compressive principal component pursuit (CPCP) [1]. Equivalently, since

$$ \{ M \in \mathbb{R}^{m \times n} \mid b = A[M] \} = \{ M \in \mathbb{R}^{m \times n} \mid \mathcal{P}_Q[M] = \mathcal{P}_Q[M_0] \}, $$

*Department of Industrial Engineering and Operations Research, Columbia University (cm3052@columbia.edu, goldfarb@columbia.edu)
†Department of Electrical Engineering, Columbia University, (yq2409@cs.columbia.edu, johnwright@ee.columbia.edu)
where $Q \subseteq \mathbb{R}^{m \times n}$ is a linear subspace spanned by $\{A_i\}_{i=1}^p$, and $P_Q$ denotes the projection operator onto that subspace, we can rewrite problem (1.3) in the (possibly) more compact form:

$$\min_{L,S} f(L, S) = \frac{1}{2} \|P_Q[L + S - M_0]\|^2_F + \lambda_L \|L\|_* + \lambda_S \|S\|_1.$$  

(1.4)

Recently, CPCP and close variants of it have been studied for different sensing operators $A$ (or equivalently different subspaces $Q$). Specifically, [2, 3, 4, 5, 6] consider the case where a subset $\Omega \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ of the entries of $M_0$ is observed. Then CPCP can be reduced to

$$\min_{L,S} \frac{1}{2} \|P_\Omega[L + S - M_0]\|^2_F + \lambda_L \|L\|_* + \lambda_S \|S\|_1,$$

(1.5)

where $P_\Omega[]$ denotes the orthogonal projection onto the linear space of matrices supported on $\Omega$, i.e., $P_\Omega[M_0](i, j) = (M_0)_{ij}$ if $(i, j) \in \Omega$ and $P_\Omega[M_0](i, j) = 0$ otherwise. [1] studies the case where each $A_k$ is a iid $\mathcal{N}(0, 1)$ matrix, which is equivalent (in distribution) to saying that we choose a linear subspace $Q$ uniformly at random from the set of all $p$-dimensional subspaces of $\mathbb{R}^{m \times n}$ and observe $P_Q[M_0]$. Consentanously, all the above works manage to provide theoretical guarantees for CPCP, under fairly mild conditions, to produce accurate estimates of $L_0$ and $P_\Omega[S_0]$ (or $S_0$), even when the number of measurements $p$ is much less than $mn$.

Inspired by these theoretical results, researchers from different fields have leveraged CPCP to solve many practical problems, including video background modeling [3], batch image alignment [7], face verification [8], photometric stereo [9], dynamic MRI [10], topic modeling [11], latent variable graphical model learning [12] and outlier detection and robust Principal Component Analysis [3], just to name a few.

Living in the era of big data, most of these applications involve large datasets and high dimensional data spaces. Therefore, to fully realize the benefit of the theory, we need provably convergent and scalable algorithms for CPCP. This has motivated much research into the development of first-order methods for problem (1.4) and its variants; e.g see [13, 14, 15, 16, 17, 18]. These methods all exploit a closed-form expression for the proximal operator of the nuclear norm, which involves the singular value decomposition (SVD). Hence, the dominant cost in each iteration is computing an SVD of the same size as the input data. This is substantially more scalable than off-the-shelf interior point solvers such as SDPT3 [19]. Nevertheless, the superlinear cost of each iteration has limited the practical applicability of these first-order methods to problems involving several thousands of data points and several thousands of dimensions. The need to compute a sequence of full or partial SVD’s is a serious bottleneck for truly large scale applications.

As a remedy, in this paper, we design more scalable algorithms to solve CPCP that compute only a rank-one SVD in each iteration. Our approach leverages two classical and widely studied ideas – Frank-Wolfe iterations to handle the nuclear norm, and proximal steps to handle the $\ell_1$ norm. This turns out to be an ideal combination of techniques to solve large scale CPCP problems. In particular, it yields algorithms that

*To transform problem (1.3) into problem (1.4), simple procedures like Gram-Schmidt might be invoked. Despite being equivalent, one formulation might be preferred over the other in practice, depending on the specifications of the sensing operator $A[:i]$. In this paper, we will mainly focus on solving problem (1.4) and its variants. Our methods, however, are not restrictive to (1.4) and can be easily extended to problem (1.3).
are substantially more scalable than prox-based first-order methods such as ISTA and FISTA [20], and converge much faster in practice than a straightforward application of Frank-Wolfe.

The remainder of this paper is organized as follows. Section 2 reviews the general properties of the Frank-Wolfe algorithm, and describes several basic building blocks that we will use in our algorithms. Section 3 and Section 4 respectively describe how to modify the Frank-Wolfe algorithm to solve CPCP’s norm constrained version

\begin{equation}
\min_{L, S} \langle L, S \rangle = \frac{1}{2} \|P_Q[L + S - M_0]\|_F^2 \quad \text{s.t. } \|L\|_* \leq \tau_L, \|S\|_1 \leq \tau_S,
\end{equation}

and the penalized version, i.e. problem (1.4), by incorporating proximal regularization to more effectively handle the $\ell_1$ norm. Convergence results and our implementation details are also discussed. Section 5 presents numerical experiments on large datasets that demonstrate the scalability of our proposed algorithms. After briefly discussing contributions in Section 6, we present proofs of all theorems and lemmas involved in the paper in the Appendix.

2. Preliminaries.

2.1. Frank-Wolfe method. The Frank-Wolfe (FW) method [21], also known as the conditional gradient method [22], applies to the general problem of minimizing a differentiable convex function $h$ over a compact, convex domain $D \subseteq \mathbb{R}^n$:

\begin{equation}
\text{minimize } h(x) \quad \text{subject to } x \in D \subseteq \mathbb{R}^n.
\end{equation}

Here, $\nabla h$ is assumed to be $L$-Lipschitz:

\begin{equation}
\forall x, y \in D, \quad \|\nabla h(x) - \nabla h(y)\| \leq L \|x - y\|.
\end{equation}

Throughout, we let $D = \max_{x, y \in D} \|x - y\|$ denote the diameter of the feasible set $D$.

In its simplest form, the Frank-Wolfe algorithm proceeds as follows. At each iteration $k$, we linearize the objective function $h$ about the current point $x^k$:

\begin{equation}
h(v) \approx h(x^k) + \langle \nabla h(x^k), v - x^k \rangle.
\end{equation}

We minimize the linearization over the feasible set $D$ to obtain $v^k \in \arg\min_{v \in D} \langle \nabla h(x^k), v \rangle$ and then take a step in the feasible descent direction $v^k - x^k$:

\begin{equation}
x^{k+1} = x^k + \frac{2}{k+2} (v^k - x^k).
\end{equation}

This yields a very simple procedure, which we summarize as Algorithm 1. The particular step size $\frac{2}{k+2}$, comes from the convergence analysis of the algorithm, which we discuss in more details below.

First proposed in [21], FW-type methods have been frequently revisited in different fields. Recently, they have experienced a resurgence in statistics, machine learning and signal processing, due to their ability to yield highly scalable algorithms for optimization with structure-encouraging norms such as the $\ell_1$ norm and nuclear norm. In particular, if $x$ is a matrix and $D = \{x \mid \|x\|_* \leq \beta\}$ is a nuclear norm ball, the subproblem

\begin{equation}
\min_{v \in D} \langle v, \nabla h(x) \rangle
\end{equation}
Algorithm 1 Frank-Wolfe method for problem (2.1)

1: Initialization: \( \mathbf{x}^0 \in \mathcal{D} \);
2: for \( k = 0, 1, 2, \ldots \) do
3: \( \mathbf{v}^k \in \text{argmin}_{\mathbf{v} \in \mathcal{D}} \langle \mathbf{v}, \nabla h(\mathbf{x}^k) \rangle \);
4: \( \gamma = \frac{2}{k+2} \);
5: \( \mathbf{x}^{k+1} = \mathbf{x}^k + \gamma (\mathbf{v}^k - \mathbf{x}^k) \);
6: end for

Algorithm 2 Frank-Wolfe method for problem (2.1) with general updating scheme

1: Initialization: \( \mathbf{x}^0 \in \mathcal{D} \);
2: for \( k = 0, 1, 2, \ldots \) do
3: \( \mathbf{v}^k \in \text{argmin}_{\mathbf{v} \in \mathcal{D}} \langle \mathbf{v}, \nabla h(\mathbf{x}^k) \rangle \);
4: \( \gamma = \frac{2}{k+2} \);
5: Update \( \mathbf{x}^{k+1} \) to some point in \( \mathcal{D} \) such that \( h(\mathbf{x}^{k+1}) \leq h(\mathbf{x}^k + \gamma (\mathbf{v}^k - \mathbf{x}^k)) \);
6: end for

In the past five decades, numerous variants of Algorithm 1 have been proposed and implemented. Many modify Algorithm 1 by replacing the simple updating rule (2.4) with more sophisticated schemes, e.g.,

\[
\mathbf{x}^{k+1} = \arg \min_{\mathbf{x}} h(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in \text{conv}\{\mathbf{x}^k, \mathbf{v}^k\}
\]

or

\[
\mathbf{x}^{k+1} = \arg \min_{\mathbf{x}} h(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \in \text{conv}\{\mathbf{x}^k, \mathbf{v}^k, \mathbf{v}^{k-1}, \ldots, \mathbf{v}^{k-j}\}.
\]

The convergence of these schemes can be analyzed simultaneously, using the fact that they produce iterates \( \mathbf{x}^{k+1} \) whose objective is no greater than that produced by the original Frank-Wolfe update scheme:

\[
h(\mathbf{x}^{k+1}) \leq h(\mathbf{x}^k + \gamma (\mathbf{v}^k - \mathbf{x}^k)).
\]

Algorithm 2 states a general version of Frank-Wolfe, whose update is only required to satisfy this relationship. It includes as special cases the updating rules (2.4), (2.6) and (2.7). This flexibility will be crucial for effectively handling the sparse structure in the CPCP problems (1.4) and (1.6).

The convergence of Algorithm 2 can be proved using well-established techniques [24, 25, 26, 27, 28, 29, 30, 31]. Using these ideas, we can show that it converges at a rate of \( O(1/k) \) in function value:

**Theorem 2.1.** Let \( \mathbf{x}^* \) be an optimal solution to (2.1). For \( \{\mathbf{x}^k\} \) generated by Algorithm 2, we have for \( k = 0, 1, 2, \ldots \),

\[
h(\mathbf{x}^k) - h(\mathbf{x}^*) \leq \frac{2LD^2}{k+2}.
\]
Note that the constant on the rate of convergence depends on the Lipschitz constant \( L \) of \( h \) and the diameter \( D \). This result was perhaps first derived by [26]. For completeness, we provide a proof of Theorem 2.1 in the Appendix.

While Theorem 2.1 guarantees that Algorithm 2 converges at a rate of \( O(1/k) \), in practice it is useful to have a more precise bound on the suboptimality at iterate \( k \). The surrogate duality gap

\[
\begin{align*}
d(x^k) &= \langle x^k - v^k, \nabla h(x^k) \rangle,
\end{align*}
\]

provides a useful upper bound on the suboptimality \( h(x^k) - h(x^*) \):

\[
\begin{align*}
h(x^k) - h(x^*) &\leq -\langle x^* - x^k, \nabla h(x^k) \rangle \\
&\leq \min_v \langle v - x^k, \nabla h(x^k) \rangle = \langle x^k - v^k, \nabla h(x^k) \rangle = d(x^k).
\end{align*}
\]

This was first proposed in [21] and later [25] showed that \( d(x^k) = O(1/k) \). In the Appendix, we prove the following refinement of this result, using ideas from [25, 30]:

**Theorem 2.2.** Let \( \{x^k\} \) be the sequence generated by Algorithm 2. Then for any \( K \geq 1 \), there exists \( 1 \leq k \leq K \) such that

\[
\begin{align*}
d(x^k) &\leq \frac{6LD^2}{K+2}.
\end{align*}
\]

Since this matches the worst case convergence rate for \( h(x^k) - h(x^*) \) (see (2.8)), it suggests that the upper bound \( d(x^k) \) provides a valid stopping criterion in practice.

For our problem, the main computational burden in Algorithms 1 and 2 will be solving the linear subproblem \( \min_{v \in \mathcal{D}} \langle v, \nabla h(x^k) \rangle \), i.e. minimizing linear functions over the unit balls for \( \| \cdot \|_\star \) and \( \| \cdot \|_1 \). Fortunately, both of these operations have simple closed-form solutions, which we will describe in the next section.

### 2.2. Optimization oracles.

We now describe several optimization oracles involving the \( \ell_1 \) norm and the nuclear norm, which serve as the main building blocks for our methods. These oracles have computational costs that are (essentially) linear in the size of the input.

**Minimizing a linear function over the nuclear norm ball.** Since the dual norm of the nuclear norm is the operator norm, i.e., \( \| Y \| = \max_{\| X \|_\star \leq 1} \langle Y, X \rangle \), the optimization problem

\[
\begin{align*}
\text{minimize}_{X} & \quad \langle Y, X \rangle \\
\text{subject to} & \quad \| X \|_\star \leq 1
\end{align*}
\]

has optimal value \( -\| Y \| \). One minimizer is the rank-one matrix \( X^* = -uv^\top \), where \( u \) and \( v \) are the left- and right- singular vectors corresponding to the leading singular value of \( Y \), and can be efficiently computed (e.g. using power method).

**Minimizing a linear function over the \( \ell_1 \) ball.** Since the dual norm of the \( \ell_1 \) norm is the \( \ell_\infty \) norm, i.e., \( \| Y \|_\infty = \max_{i,j} | Y_{ij} | = \max_{\| X \|_1 \leq 1} \langle Y, X \rangle \), the optimization problem

\[
\begin{align*}
\text{minimize}_{X} & \quad \langle Y, X \rangle \\
\text{subject to} & \quad \| X \|_1 \leq 1
\end{align*}
\]

has optimal value \( -\| Y \|_\infty \). One minimizer is the one-sparse matrix \( X^* = -e_i e_j^\top \), where \( (i^*, j^*) \in \arg \max_{(i,j)} | Y_{ij} | \); i.e. \( X^* \) has exactly one nonzero element.

\[1\] In some situations, we can significantly reduce this cost by solving this problem inexactly [27, 25]. Our algorithms and results can also tolerate inexact step calculations; we omit the discussion here for simplicity.
Projection onto the $\ell_1$-ball. To effectively handle the sparse term in the norm constrained problem (1.6), we will need to modify the Frank-Wolfe algorithm by incorporating additional projection steps. For any $Y \in \mathbb{R}^{m \times n}$ and $\beta > 0$, the projection onto the $\ell_1$-ball:

$$\mathcal{P}_{\|\cdot\|_1 \leq \beta}[Y] = \arg\min_{\|X\|_1 \leq \beta} \frac{1}{2} \|X - Y\|_F^2,$$

(2.14)

can be easily solved with $O(mn(\log m + \log n))$ cost [32]. Moreover, a divide and conquer algorithm, achieving linear cost in expectation to solve (2.14), has also been proposed in [32].

Proximal mapping of $\ell_1$ norm. To effectively handle the sparse term arising in problem (1.4), we will need to modify the Frank-Wolfe algorithm by incorporating additional proximal steps. For any $Y \in \mathbb{R}^{m \times n}$ and $\lambda > 0$, the proximal mapping of $\ell_1$ norm has the following closed-form expression

$$\mathcal{T}_\lambda[Y] = \arg\min_{X \in \mathbb{R}^{m \times n}} \frac{1}{2} \|X - Y\|_F^2 + \lambda \|X\|_1,$$

(2.15)

where $\mathcal{T}_\lambda : \mathbb{R} \to \mathbb{R}$ denotes the soft-thresholding operator $\mathcal{T}_\lambda(x) = \text{sgn}(x) \max(|x| - \lambda, 0)$, and extension to matrices is obtained by applying the scalar operator $\mathcal{T}_\lambda(\cdot)$ to each element.

3. FW-P Method for Norm Constrained Problem. In this section, we develop scalable algorithms for the norm-constrained compressive principal component pursuit problem,

$$\min_{L,S} l(L, S) = \frac{1}{2} \|\mathcal{P}_Q[L + S - M]\|_F^2 \ \text{s.t.} \ \|L\|_* \leq \tau_L, \|S\|_1 \leq \tau_S.$$

(3.1)

We first describe a straightforward application of the Frank-Wolfe method to this problem. We will see that although it has relatively cheap iterations, it converges very slowly on typical numerical examples, because it only makes a one-sparse update to the sparse term $S$ at a time. We will show how to remedy this problem by augmenting the FW iteration with an additional proximal step (essentially a projected gradient step) in each iteration, yielding a new algorithm which updates $S$ much more efficiently. Because it combines Frank-Wolfe and projection steps, we will call this new algorithm Frank-Wolfe-Projection (FW-P).

Properties of the objective and constraints. To apply Frank-Wolfe to (3.1), we first note that the objective $l(L, S)$ in (3.1) is differentiable, with

$$\nabla_L l(L, S) = \mathcal{P}_Q[L + S - M]$$

(3.2)

$$\nabla_S l(L, S) = \mathcal{P}_Q[L + S - M].$$

(3.3)

Moreover, the following lemma shows that the gradient map $\nabla l(L, S) = (\nabla_L l, \nabla_S l)$ is 2-Lipschitz:

Lemma 3.1. For all $(L, S)$ and $(L', S')$, we have $\|\nabla l(L, S) - \nabla l(L', S')\|_F \leq 2 \|L - L'\|_F + 2 \|S - S'\|_F$.

The feasible set in (3.1) is compact. The following lemma bounds its diameter $D$:

Lemma 3.2. The feasible set $\mathcal{D} = \{(L, S) \mid \|L\|_* \leq \tau_L, \|S\|_1 \leq \tau_S\}$ has diameter $D \leq 2\sqrt{\tau_L^2 + \tau_S^2}$. 

3.1. Frank-Wolfe for problem (3.1). Since (3.1) asks us to minimize a convex, differentiable function with Lipschitz gradient over a compact convex domain, the Frank-Wolfe method in Algorithm 1 applies. It generates a sequence of iterates $x^k = (L^k, S^k)$. Using the expression for the gradient in (3.2)-(3.3), at each iteration, the step direction $v^k = (V^k_L, V^k_S)$ is generated by solving the linearized subproblem

$$
\begin{align*}
(V^k_L, V^k_S) & \in \arg\min \left\{ \left( \frac{\partial}{\partial L} |L^k + S^k - M|, (V_L, V_S) \right) \right\} \\
\text{s.t.} \quad & \|V_L\|_* \leq \tau_L, \quad \|V_S\|_1 \leq \tau_S,
\end{align*}
$$

which decouples into two independent subproblems:

$$
\begin{align*}
V^k_L & \in \arg\min_{\|V_L\|_* \leq \tau_L} \langle P_Q(L^k + S^k - M), V_L \rangle, \\
V^k_S & \in \arg\min_{\|V_S\|_1 \leq \tau_S} \langle P_Q(L^k + S^k - M), V_S \rangle.
\end{align*}
$$

These subproblems can be easily solved by exploiting the linear optimization oracles introduced in Section 2.2. In particular,

$$
\begin{align*}
V^k_L &= -\tau_L u^k (v^k)^T, \\
V^k_S &= -\tau_S e^k_i (e^k_j)^T,
\end{align*}
$$

where $u^k$ and $v^k$ are leading left- and right- singular vectors of $P_Q(L^k + S^k - M)$ and $(u^*, v^*)$ is the of the largest element of $P_Q(L^k + S^k - M)$ in magnitude. Algorithm 3 gives the Frank-Wolfe method specialized to problem (3.1).

**Algorithm 3** Frank-Wolfe method for problem (3.1)

1. Initialization: $L^0 = S^0 = 0$;
2. for $k = 0, 1, 2, \cdots$ do
3. $D^k_L \in \arg\min_{\|D_L\|_* \leq 1} \langle P_Q(L^k + S^k - M), D_L \rangle; V^k_L = \tau_L D^k_L$;
4. $D^k_S \in \arg\min_{\|D_S\|_1 \leq 1} \langle P_Q(L^k + S^k - M), D_S \rangle; V^k_S = \tau_S D^k_S$;
5. $\gamma = \frac{2}{k+2}$;
6. $L^{k+1} = L^k + \gamma (V^k_L - L^k)$;
7. $S^{k+1} = S^k + \gamma (V^k_S - S^k)$;
8. end for

The major advantage of Algorithm 3 derives from the simplicity of the update rules (3.5)-(3.6). Both have closed form, and both can be computed in time (essentially) linear in the size of the input. Because $V^k_L$ is rank-one, the algorithm can be viewed as performing a sequence of rank one updates.

The major disadvantage of Algorithm 3 is that $S$ has only a one-sparse update at each iteration, since $V^k_S = -\tau_S e^k_i (e^k_j)^T$ has only one nonzero entry. This is a significant disadvantage in practice, as the optimal $S^*$ may have a relatively large number of nonzero entries. Indeed, in theory, the CPCP relaxation works even when a constant fraction of the entries in $S_0$ are nonzero. In applications such as foreground-background separation, the number of nonzero entries in the target sparse term can be quite large. The red curves in Figure 1 show the effect of this on the practical convergence of the algorithm, on a simulated example of size $1,000 \times 1,000$, in which about $1\%$ of the entries in the target sparse matrix $S_0$ are nonzero. As shown, the progress is quite slow.
since (3.9) It can then be easily verified that arg min (3.7)

The data are generated in Matlab as $n = 1000, r = 5$. $L_0 = \text{randn}(m, r) \times \text{randn}(r, n)$; $\Omega = \text{ones}(m, n)$; $S_0 = 100 \times \text{randn}(m, n) \times (\text{rand}(m, n) < 0.01)$; $M = L_0 + S_0 + \text{randn}(m, n)$; $\tau_1 = \text{norm}(\text{vec}(S_0), 1)$; $\tau_2 = \text{norm}(\text{vec}(S_0), 1)$; The left figure plots log(rel. err.) versus the iteration number $k$. The right figure plots log(rel. err.) versus $k$. The FW-P method is clearly more efficient than the straightforward FW method in recovering $L_0$ and $S_0$.

3.2. FW-P algorithm: combining Frank-Wolfe and projected gradient.

To overcome the drawback of the naive Frank-Wolfe algorithm described above, we propose incorporating an additional gradient projection step after each Frank-Wolfe update. This additional step updates the sparse term $S$ only, with the goal of accelerating convergence in these variables. At iteration $k$, let $(L^{k+1/2}, S^{k+1/2})$ be the result produced by Frank-Wolfe. To produce the next iterate, we retain the low rank term $L^{k+1/2}$, but set

\begin{align}
S^{k+1} &= P_{\|\cdot\|_1 \leq \tau_S} \left[ S^{k+1/2} - \nabla S l(L^{k+1/2}, S^{k+1/2}) \right] \\
&= P_{\|\cdot\|_1 \leq \tau_S} \left[ S^{k+1/2} - P_Q [L^{k+1/2} + S^{k+1/2} - M] \right],
\end{align}

i.e. we simply take an additional projected gradient step in the sparse term $S$. The resulting algorithm is presented as Algorithm 4 below. We call this method the FW-P algorithm, as it combines Frank-Wolfe steps and projections. In Figure 1, we compare Algorithms 3 and 4 on synthetic data. In this example, the FW-P method is clearly more efficient in recovering $L_0$ and $S_0$.

The convergence of Algorithm 4 can be analyzed by recognizing it as a specific instance of the generalized Frank-Wolfe iteration in Algorithm 2. This projection step (3.8) can be regarded as a proximal step to set $S^{k+1}$ as

\[ \arg \min_{\|S\|_1 \leq \tau_S} \tilde{l}^{k+1/2}(S) := l(L^{k+1/2}, S^{k+1/2}) + (\nabla S l(L^{k+1/2}, S^{k+1/2}), S - S^{k+1/2}) + \frac{1}{2} \|S - S^{k+1/2}\|_F^2. \]

It can then be easily verified that

\[ \tilde{l}^{k+1/2}(S^{k+1/2}) = l(L^{k+1/2}, S^{k+1/2}), \quad \text{and} \quad \tilde{l}^{k+1/2}(S) \geq l(L^{k+1/2}, S) \text{ for any } S, \]

since $\nabla S l(L, S)$ is 1-Lipschitz. This implies that the FW-P algorithm chooses a next iterate whose objective is no worse than that produced by the Frank-Wolfe step:

\[ l(L^{k+1}, S^{k+1}) = l(L^{k+1/2}, S^{k+1}) \leq \tilde{l}^{k+1/2}(S^{k+1}) \leq \tilde{l}^{k+1/2}(S^{k+1/2}) = l(L^{k+1/2}, S^{k+1/2}). \]
To apply the Frank-Wolfe method, we exploit a two-step reformulation to transform the
convergence guarantees.

In Section 4.1, we introduce our FW-T method, which resolves this issue. Our FW-T method
scalable algorithm for the penalized version of the CPCP problem,

Moreover, for any $K \geq 1$, there exists $1 \leq \tilde{k} \leq K$ such that the surrogate duality gap
(defined in (2.9)) satisfies

\begin{equation}
\left. d(x^\tilde{k}) = \left\langle x^\tilde{k} - v^\tilde{k}, \nabla l(x^\tilde{k}) \right\rangle \right\} \leq \frac{48(\tau_L^2 + \tau_S^2)}{K + 2}.
\end{equation}

4. FW-T Method for Penalized Problem. In this section, we develop a scalable algorithm for the penalized version of the CPCP problem,

\begin{equation}
\min_{L, S} \ f(L, S) = \frac{1}{2} ||P_Q[L + S - M]||_F^2 + \lambda_L ||L||_* + \lambda_S ||S||_1.
\end{equation}

In Section 4.1, we reformulate problem (4.1) into the form of (2.1) so that the Frank-
Wolfe method can be applied. In Section 4.2, we apply the Frank-Wolfe method
directly to the reformulated problem, achieving linear per-iteration cost and $O(1/k)$
convergence in function value. However, because it updates the sparse term one
element at a time, it converges very slowly on typical numerical examples. In Section 4,
we introduce our FW-T method, which resolves this issue. Our FW-T method
essentially exploits the Frank-Wolfe step to handle the nuclear norm and a proximal
gradient step to handle the $\ell_1$-norm, while keeping iteration cost low and retaining
convergence guarantees.

4.1. Reformulation as smooth, constrained optimization. Note that problem
(4.1) has a non-differentiable objective function and an unbounded feasible set.
To apply the Frank-Wolfe method, we exploit a two-step reformulation to transform
(4.1) into the form of (2.1). First, we borrow ideas from [24] and work with the

**Algorithm 4 FW-P method for problem (3.1)**

1: Initialization: $L^0 = S^0 = 0$
2: for $k = 0, 1, 2, \cdots$ do
3: $D^k_L \in \arg\min_{D_L : \|D_L\| \leq 1} (P_Q[L^k + S^k - M], D_L); \ V^k_L = \tau_L D^k_L$;
4: $D^k_S \in \arg\min_{D_S : \|D_S\| \leq 1} (P_Q[L^k + S^k - M], D_S); \ V^k_S = \tau_S D^k_S$;
5: $\gamma = \frac{2}{k+2}$;
6: $L^{k+\frac{1}{2}} = L^k + \gamma(V^k_L - L^k)$;
7: $S^{k+\frac{1}{2}} = S^k + \gamma(V^k_S - S^k)$;
8: $S^{k+1} = \frac{\|l\|}{\tau_S} [S^{k+\frac{1}{2}} - P_Q[L^{k+\frac{1}{2}} + S^{k+\frac{1}{2}} - M]]$;
9: $L^{k+1} = L^{k+\frac{1}{2}}$;
10: end for

This is precisely the property that is required to invoke Algorithm 2 and Theorems
2.1 and 2.2. Using Lemmas 4.1 and 4.2 to estimate the Lipschitz constant of $\nabla l$ and the diameter of $D$, we obtain the following result, which shows that FW-P retains the
$O(1/k)$ convergence rate of the original FW method:

**Theorem 3.3.** Let $l^*$ be the optimal value to problem (3.1), $x^k = (L^k, S^k)$ and
$v^k = (V^k_L, V^k_S)$ be the sequence produced by Algorithm 4. Then we have

\begin{equation}
l(L^k, S^k) - l^* \leq \frac{16(\tau_L^2 + \tau_S^2)}{k + 2}.
\end{equation}

Moreover, for any $K \geq 1$, there exists $1 \leq \tilde{k} \leq K$ such that the surrogate duality gap
(defined in (2.9)) satisfies

\begin{equation}
d(x^\tilde{k}) = \left\langle x^\tilde{k} - v^\tilde{k}, \nabla l(x^\tilde{k}) \right\rangle \leq \frac{48(\tau_L^2 + \tau_S^2)}{K + 2}.
\end{equation}
epigraph reformulation of (4.1),

\[
\begin{align*}
\min & \quad g(L, S, t_L, t_S) = \frac{1}{2} \|P_Q[L + S - M]\|^2_F + \lambda_L t_L + \lambda_S t_S \\
\text{s.t.} & \quad \|L\|_* \leq t_L, \quad \|S\|_1 \leq t_S,
\end{align*}
\]  

(4.2) obtained by introducing auxiliary variables \(t_L\) and \(t_S\). Now the objective function \(g(L, S, t_L, t_S)\) is differentiable, with

\[
\begin{align*}
\nabla_L g(L, S, t_L, t_S) &= \nabla_S g(L, S, t_L, t_S) = P_Q[L + S - M], \\
\nabla_{t_L} g(L, S, t_L, t_S) &= \lambda_L, \quad \nabla_{t_S} g(L, S, t_L, t_S) = \lambda_S.
\end{align*}
\]  

(4.3) (4.4) A calculation, which we summarize in the following lemma, shows that the gradient \(\nabla g(L, S, t_L, t_S) = (\nabla_L g, \nabla_S g, \nabla_{t_L} g, \nabla_{t_S} g)\) is 2-Lipschitz:

**Lemma 4.1.** For all \((L, S, t_L, t_S)\) and \((L', S', t'_L, t'_S)\) feasible to (4.2),

\[
\|\nabla g(L, S, t_L, t_S) - \nabla g(L', S', t'_L, t'_S)\|_F \leq 2 \|(L, S, t_L, t_S) - (L', S', t'_L, t'_S)\|_F.
\]  

(4.5) However, the Frank-Wolfe method still cannot deal with (4.2), since its feasible region is unbounded. If we could somehow obtain upper bounds on the optimal values of \(t_L\) and \(t_S\): \(U_L \geq t_L^*\) and \(U_S \geq t_S^*\), then we could solve the equivalent problem

\[
\begin{align*}
\min & \quad \frac{1}{2} \|P_Q[L + S - M]\|^2_F + \lambda_L t_L + \lambda_S t_S \\
\text{s.t.} & \quad \|L\|_* \leq t_L \leq U_L, \quad \|S\|_1 \leq t_S \leq U_S,
\end{align*}
\]  

(4.6) which now has a compact and convex feasible set. One simple way to obtain such \(U_L, U_S\) is as follows. One trivial feasible solution to problem (4.2) is \(L = 0, S = 0, t_L = 0, t_S = 0\). This solution has objective value \(\frac{1}{2} \|P_Q[M]\|^2_F\). Hence, the optimal objective value is no larger than this. This implies that for any optimal \(t_L^*, t_S^*\),

\[
t_L^* \leq \frac{1}{2\lambda_L} \|P_Q[M]\|^2_F, \quad t_S^* \leq \frac{1}{2\lambda_S} \|P_Q[M]\|^2_F.
\]  

(4.7) Hence, we can always choose

\[
U_L = \frac{1}{2\lambda_L} \|P_Q[M]\|^2_F, \quad U_S = \frac{1}{2\lambda_S} \|P_Q[M]\|^2_F,
\]  

(4.8) to produce a valid, bounded feasible region. The following lemma bounds its diameter \(D\):

**Lemma 4.2.** The feasible set \(D = \{(L, S, t_L, t_S) \mid \|L\|_* \leq t_L \leq U_L, \quad \|S\|_1 \leq t_S \leq U_S\}\) has diameter \(D \leq \sqrt{5} \cdot \sqrt{U_L^2 + U_S^2}\).

With these modifications, we can apply Frank-Wolfe directly to obtain a solution \((\hat{L}, \hat{S}, \hat{t}_L, \hat{t}_S)\) to (4.6), and hence to produce a solution \((\hat{L}, \hat{S})\) to the original problem (4.1). In subsection 4.2, we describe how to do this. Unfortunately, this straightforward solution has two main disadvantages. First, as in the norm constrained case, it produces only one-sparse updates to \(S\), which results in slow convergence. Second, the exact primal convergence rate in Theorem 2.1 depends on the diameter of the feasible set, which in turn depends on the accuracy of our (crude) upper bounds \(U_L\) and \(U_S\). In subsection 4.3, we show how to remedy both issues, yielding a Frank-Wolfe-Thresholding method that performs significantly better in practice.
4.2. Frank-Wolfe for problem (4.6). Applying the Frank-Wolfe method in Algorithm 1 generates a sequence of iterates \( x^k = (L^k, S^k, t^k_L, t^k_S) \). Using the expressions

for the gradient in (4.3) and (4.4), at each iteration, \( v^k = (V^k_L, V^k_S, V^k_{t_L}, V^k_{t_S}) \) is generated by solving the linearized subproblem

\[
\begin{align*}
    v^k &\in \arg \min_{v \in D} \langle p_Q(L^k + S^k - M), V_L + V_S \rangle + \lambda_L V_{t_L} + \lambda_S V_{t_S},
\end{align*}
\]

which can be decoupled into two independent subproblems,

\[
\begin{align*}
    (V^k_L, V^k_{t_L}) &\in \arg \min_{\|V_L\|_1 \leq V_{t_L} \leq U_L} g_L(V_L, V_{t_L}) = \langle p_Q(L^k + S^k - M), V_L \rangle + \lambda_L V_{t_L}, \\
    (V^k_S, V^k_{t_S}) &\in \arg \min_{\|V_S\|_1 \leq V_{t_S} \leq U_S} g_S(V_S, V_{t_S}) = \langle p_Q(L^k + S^k - M), V_S \rangle + \lambda_S V_{t_S}.
\end{align*}
\]

Let us consider problem (4.10) first. Set

\[
D^k_L \in \arg \min_{\|D_L\|_1 \leq 1} \hat{g}_L(D_L) = \langle p_Q[L^k + S^k - M], D_L \rangle + \lambda_L.
\]

Because \( g_L(V_L, V_{t_L}) \) is a homogeneous function, i.e., \( g_L(\alpha V_L, \alpha V_{t_L}) = \alpha g_L(V_L, V_{t_L}) \), for any \( \alpha \in \mathbb{R} \), its optimal value \( g(V^k_L, V^k_{t_L}) = V^k_{t_L} \hat{g}_L(D^k_L) \). Hence \( V^k_{t_L} = U_L \) if \( \hat{g}_L(D^k_L) < 0 \), and \( V^k_{t_L} = 0 \) if \( \hat{g}_L(D^k_L) > 0 \). From this observation, it can be easily verified (see also [24, Lemma 1] for a more general result) that

\[
\begin{align*}
    (V^k_L, V^k_{t_L}) &\in \begin{cases} 
        \{(0, 0)\} & \text{if } \hat{g}_L(D^k_L) > 0 \\
        \text{conv}\{0, 0, U_L(D^k_L, 1)\} & \text{if } \hat{g}_L(D^k_L) = 0 \\
        \{U_L(D^k_L, 1)\} & \text{if } \hat{g}_L(D^k_L) < 0.
    \end{cases}
\end{align*}
\]

In a similar manner, we can update \( (V^k_S, V^k_{t_S}) \). This leads fairly directly to the implementation of the Frank-Wolfe method for problem (4.6), described in Algorithm 5. As a direct corollary of Theorem 2.1, using parameters calculated in Lemmas 4.1 and 4.2, we have

**Corollary 4.3.** Let \( x^* = (L^*, S^*, t^*_L, t^*_S) \) be an optimal solution to (4.6). For \( \{x^k\} \) generated by Algorithm 5, we have for \( k = 0, 1, 2, \ldots \),

\[
g(x^k) - g(x^*) \leq \frac{20(U^2_L + U^2_S)}{k + 2}.
\]

In addition to the above convergence result, another major advantage of Algorithm 5 is the simplicity of the update rules (lines 3-4 in Algorithm 5). Both have closed-form solutions that can be computed in time (essentially) linearly dependent on the size of the input.

However, two clear limitations substantially hinder Algorithm 5’s efficiency. First, as in the norm constrained case, \( V^k_S \) has only one nonzero entry, so \( S \) has a one-sparse update in each iteration. Second, the exact rate of convergence relies on our (crude) guesses of \( U_L \) and \( U_S \) (Corollary 4.3). In the next subsection, we present remedies to resolve both issues.

4.3. FW-T algorithm: combining Frank-Wolfe and proximal methods.

To alleviate the difficulties faced by Algorithm 5, we propose a new algorithm called Frank-Wolfe-Thresholding (FW-T) (Algorithm 6), that combines a modified FW step with a proximal gradient step. Below we highlight the key features of FW-T.
Algorithm 5 Frank-Wolfe method for problem (4.6)

1: Initialization: \( L^0 = S^0 = 0; t_L^0 = t_S^0 = 0; \)
2: for \( k = 0, 1, 2, \cdots \) do
3: \( \bar{D}_L^k \in \arg \min_{\|D_L\| \leq 1} \langle \mathcal{P}_Q [L^k + S^k - M], D_L \rangle; \)
4: \( \bar{D}_S^k \in \arg \min_{\|D_S\| \leq 1} \langle \mathcal{P}_Q [L^k + S^k - M], D_S \rangle; \)
5: if \( \lambda_L \geq -\langle \mathcal{P}_Q [L^k + S^k - M], \bar{D}_L^k \rangle \) then
6: \( V_{L}^k = 0; V_{L}^k = 0 \)
7: else
8: \( V_{L}^k = U_L D_{L}^k, V_{L}^k = U_L; \)
9: end if
10: if \( \lambda_S \geq -\langle \mathcal{P}_Q [L^k + S^k - M], \bar{D}_S^k \rangle \) then
11: \( V_{S}^k = 0; V_{S}^k = 0; \)
12: else
13: \( V_{S}^k = U_S D_{S}^k, V_{S}^k = U_S; \)
14: end if
15: \( \gamma = \frac{2}{k+2}; \)
16: \( L^{k+1} = (1 - \gamma)L^k + \gamma V_{L}^k, t_{L}^{k+1} = (1 - \gamma)t_{L}^k + \gamma V_{L}^k; \)
17: \( S^{k+1} = (1 - \gamma)S^k + \gamma V_{S}^k, t_{S}^{k+1} = (1 - \gamma)t_{S}^k + \gamma V_{S}^k; \)
18: end for

Proximal gradient step for \( S \). To update \( S \) in a more efficient way, we incorporate an additional proximal gradient step for \( S \). At iteration \( k \), let \((L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}})\) be the result produced by Frank-Wolfe step. To produce the next iterate, we retain the low-rank term \( L^{k+\frac{1}{2}} \), but execute a proximal gradient step for the function \( f(L^{k+\frac{1}{2}}, S) \) at the point \( S^{k+\frac{1}{2}} \), i.e.

\[
S^{k+1} = \arg \min_S \left\langle \nabla f(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}), S - S^{k+\frac{1}{2}} \right\rangle + \frac{1}{2} \left\| S - S^{k+\frac{1}{2}} \right\|_F^2 + \lambda_S \left\| S \right\|_1
\]

(4.15)

\[
= \arg \min_S \left\langle \mathcal{P}_Q [L^{k+\frac{1}{2}} + S^{k+\frac{1}{2}} - M], S - S^{k+\frac{1}{2}} \right\rangle + \frac{1}{2} \left\| S - S^{k+\frac{1}{2}} \right\|_F^2 + \lambda_S \left\| S \right\|_1
\]

which can be easily computed using the soft-thresholding operator:

\[
S^{k+1} = \mathcal{T}_\lambda S \left[ S^{k+\frac{1}{2}} - \mathcal{P}_Q [L^{k+\frac{1}{2}} + S^{k+\frac{1}{2}} - M] \right].
\]

Exact line search. For the Frank-Wolfe step, instead of choosing the fixed step length \( \frac{2}{k+2} \), we implement an exact line search by solving a two-dimensional quadratic problem (4.18), as in [24]. This modification turns out to be crucial to achieve a primal convergence result that only weakly depends on the tightness of our guesses \( U_L \) and \( U_S \).

Adaptive updates of \( U_L \) and \( U_S \). We initialize \( U_L \) and \( U_S \) using the crude bound (4.8). Then, at the end of the \( k \)-iteration, we respectively update

\[
U_{L}^{k+1} = g(L^{k+1}, S^{k+1}, t_{L}^{k+1}, t_{S}^{k+1})/\lambda_L, \quad U_{S}^{k+1} = g(L^{k+1}, S^{k+1}, t_{L}^{k+1}, t_{S}^{k+1})/\lambda_S.
\]

(4.17)

This scheme maintains the property that \( U_{L}^{k+1} \geq t_{L}^* \) and \( U_{S}^{k+1} \geq t_{S}^* \). Moreover, we prove (Lemma 4.4) that \( g \) is non-increasing through our algorithm, and so this scheme produces a sequence of tighter upper bounds for \( U_{L}^* \) and \( U_{S}^* \). Although this dynamic
Algorithm 6 FW-T method for problem (4.1)

1: **Input:** data matrix $M \in \mathbb{R}^{m \times n}$; weights $\lambda_L, \lambda_S > 0$; max iteration number $T$
2: **Initialization:** $L^0 = S^0 = 0$; $t_L^0 = t_S^0 = 0$; $U_L^0 = g(L^0, S^0, t_L^0, t_S^0)/\lambda_L$; $U_S^0 = g(L^0, S^0, t_L^0, t_S^0)/\lambda_S$;
3: for $k = 0, 1, 2, \ldots, T$ do
4:  same as lines 3-14 in Algorithm 5;
5:  $(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}})$ is computed as an optimizer to

$$
\min \frac{1}{2} \|P_Q[L + S - M]\|_F^2 + \lambda_L t_L + \lambda_S t_S
$$

s.t. $(L, t_L) \in \text{conv} \left\{ \left( L_k^k, t_L^k \right), \left( V_k^k, t_L^k \right) \right\}$

$(S, t_S) \in \text{conv} \left\{ \left( S_k^k, t_S^k \right), \left( V_S^k, t_S^k \right) \right\}$;

6: $S^{k+1} = T_0 \left[ S^{k+\frac{1}{2}} - P_Q[L^{k+\frac{1}{2}} + S^{k+\frac{1}{2}} - M], \lambda_S \right]$;
7: $L^{k+1} = L^{k+\frac{1}{2}}, t_L^{k+1} = t_L^{k+\frac{1}{2}}, t_S^{k+1} = \|S^{k+1}\|_1$;
8: $U_L^{k+1} = g(L^{k+1}, S^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_L$;
9: $U_S^{k+1} = g(L^{k+1}, S^{k+1}, t_L^{k+1}, t_S^{k+1})/\lambda_S$;
10: end for

The scheme does not improve the theoretical convergence result, some acceleration is empirically exhibited.

**Convergence analysis.** Since both the FW step and the proximal gradient step do not increase the objective value, we can easily recognize FW-T method as a descent algorithm:

**Lemma 4.4.** Let $\{(L^k, S^k, t_L^k, t_S^k)\}$ be the sequence of iterates produced by the FW-T algorithm. For each $k = 0, 1, 2, \ldots$, \( (L^{k+1}, S^{k+1}, t_L^{k+1}, t_S^{k+1}) \leq (L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \leq (L^k, S^k, t_L^k, t_S^k) \).

Moreover, we can establish primal convergence (almost) independent of $U_L^0$ and $U_S^0$.

**Theorem 4.5.** Let $r_L^* \geq 0$ and $r_S^*$ be the smallest radii such that

\[
(4.20) \quad \left\{ (L, S) \mid f(L, S) \leq g(L^0, S^0, t_L^0, t_S^0) = \frac{1}{2} \|P_Q[M]\|_F^2 \right\} \subseteq B(r_L^*) \times B(r_S^*),
\]

where $
B(r) \doteq \{ X \in \mathbb{R}^{m \times n} \| X \|_F \leq r \}$ for any $r \geq 0$.\(^3\) Then for the sequence $\{(L^k, S^k, t_L^k, t_S^k)\}$ generated by Algorithm 6, we have

\[
(4.21) \quad g(L^k, S^k, t_L^k, t_S^k) - g(L^*, S^*, t_L^*, t_S^*) \leq \frac{\min \{4(t_L^0 + r_L^0)^2 + 4(t_S^0 + r_S^0)^2, 16(U_L^0)^2 + 16(U_S^0)^2 \}}{k + 2}.
\]

\(^3\)Since the objective function in problem (4.1) is coercive, i.e. $\lim_{k \to +\infty} f(L^k, S^k) = +\infty$ for any sequence $(L^k, S^k)$ such that $\lim_{k \to +\infty} \|P_Q[M]\|_F = +\infty$, clearly $r_L^* \geq 0$ and $r_S^* \geq 0$ exist.
Since $U^0_L$ and $U^0_S$ are quite crude upper bounds for $t^*_L$ and $t^*_S$, $16(U^0_L)^2 + 16(U^0_S)^2$ could be much larger than $4(t^*_L + t^*_S)^2 + 4(t^*_L + t^*_S)^2$. Therefore, this primal convergence results depend on $U^0_L$ and $U^0_S$ in a very weak manner.

However, the convergence result of the surrogate duality gap $d(x^k)$ still hinges upon the upper bounds:

**Theorem 4.6.** Let $x^k$ denote $(L^k, S^k, t^k_L, t^k_S)$ generated by Algorithm 6. Then for any $K \geq 1$, there exists $1 \leq \bar{k} \leq K$ such that

\[
g(x^\bar{k}) - g(x^*) \leq d(x^\bar{k}) \leq \frac{48 ((U^0_L)^2 + (U^0_S)^2)}{K+2}.
\]

**Stopping criterion.** Compared to the convergence of $g(x^k)$ (Theorem 4.5), the convergence result for $d(x^k)$ can be much slower (Theorem 4.6). Therefore, here the surrogate duality gap $d(\cdot)$ is not that suitable to serve as a stopping criterion. Consequently, in our implementation, we terminate Algorithm 6 if

\[
(g(x^{k+1}) - g(x^k)) / g(x^k) \leq \varepsilon,
\]

for five consecutive iterations.

**5. Numerical Experiments.** In this section, we report numerical results obtained by applying our FW-T method (Algorithm 6) to problem (1.5) with real data arising from applications considered in [3]: foreground/background separation from surveillance videos, and shadow and specularity removal from face images.

Given observations \( \{M_0(i, j) \mid (i, j) \in \Omega \} \), where $\Omega \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\}$ is the index set of the observable entries in $M_0 \in \mathbb{R}^{m \times n}$, we assigned weights

\[
\lambda_L = \delta \rho \|P_\Omega[M_0]\|_F \quad \text{and} \quad \lambda_S = \delta \sqrt{\rho \|P_\Omega[M_0]\|_F} / \sqrt{\max(m,n)}
\]

to problem (1.5),\footnote{The ratio $\lambda_L/\lambda_S = \sqrt{\rho \max(m,n)}$ follows the suggestion in [3]. For applications in computer vision at least, our choices in $\lambda_L$ and $\lambda_S$ seem to be quite robust, although it is possible to improve the performance by making slight adjustments to our current settings of $\lambda_L$ and $\lambda_S$.} where $\rho = |\Omega|/mn$ and $\delta$ is chosen as 0.001 for the surveillance problem and 0.01 for the face problem.

We compared our FW-T method with the popular first-order methods iterative soft-thresholding algorithm (ISTA) and fast iterative soft-thresholding algorithm (FISTA) [20], whose implementations both used partial singular value decomposition (SVD). For detailed descriptions and implementations of ISTA and FISTA, please refer to Appendix E. We set $\varepsilon = 10^{-3}$ as FW-T’s stopping criterion,\footnote{As discussed in [33, 34], with noisy data, solving optimization problems to high accuracy does not necessarily improve the recovery quality. Consequently, we set $\varepsilon$ to a modest value.} and terminated ISTA and FISTA whenever they reached the objective value returned by the FW-T method.\footnote{All codes are available at: https://sites.google.com/site/mucun1988/publi} All the experiments were conducted on a computer with Intel Xeon E5-2630 Processor (12 cores at 2.4 GHz), and 64GB RAM running MATLAB R2012b (64 bits).

**Foreground-background separation from surveillance video.** In surveillance videos, due to the strong correlation between frames, it is natural to model the background as low rank; while foreground objects, such as cars or pedestrians, that normally occupy only a fraction of the video, can be treated as sparse. So, if we stack each frame as a column in the data matrix $M_0$, it is reasonable to assume that

\[
M_0 = \lambda_L U^0_L V^0_L + \lambda_S U^0_S V^0_S + \Theta,
\]

\[
M_0 = \sum_{i=1}^{k} \lambda_L U^i_L V^i_L + \lambda_S U^i_S V^i_S + \Theta,
\]

\[
M_0 = \sum_{i=1}^{k} \lambda_L U^i_L V^i_L + \lambda_S U^i_S V^i_S + \Theta.
\]
Per-iteration cost vs. the number of frames in Airport and Square videos with full observation. The per-iteration cost of our FW-T method grows linearly with the size of data, in contrast with the superlinear per-iteration cost of ISTA and FISTA. That makes the FW-T method more advantageous or may even be the only feasible choice for large problems.

$M_0 \approx L_0 + S_0$, where $L_0$ captures the background and $S_0$ represents the foreground movements. Here, we solved problem (1.5) for videos introduced in [35] and [36]. The observed entries were sampled uniformly with ratio $\rho$ chosen respectively as 1, 0.8 and 0.6.

Table 1 summarizes the numerical performances of FW-T, ISTA and FISTA in terms of the iteration number and running time (in seconds). As can be observed, our FW-T method is more efficient than ISTA and FISTA, and the advantage becomes more prominent as the size of the data grows and the observations are more compressed (with smaller sampling ratio $\rho$). Even though the FW-T method takes more iterations than ISTA and FISTA, it takes less time in general due to its low per-iteration cost. To illustrate this more clearly, in Figure 2, we plot the per-iteration cost of these three methods on the Airport and Square videos as a function of the number of frames. The computational cost of FW-T scales linearly with the size of the data, whereas the cost of the other methods increases superlinearly. Another observation is that as the number of measurements decreases, the iteration numbers of both ISTA and FISTA methods grow substantially, while the FW-T method is quite stable. This explains the more favorable behavior of the FW-T method when $\rho$ is small. In Figure 3, frames of the original videos, the backgrounds and the foregrounds produced by the FW-T method are presented, and the separation achieved is quite satisfactory.

Shadow and specularity removal from face images. Images taken under varying illumination can also be modeled as the superposition of low-rank and sparse components. Here, the data matrix $M_0$ is again formed by stacking each image as a column. The low-rank term $L_0$ captures the smooth variations [37], while the sparse term $S_0$ represents cast shadows and specularities [38, 8]. CPCP can be used to remove the shadows and specularities [3, 8]. Here, we solved problem (1.4) for YaleB face images [39]. Table 2 summarizes the numerical performances of FW-T, ISTA and FISTA. Similar to the observation made regarding the above surveillance video experiment, the number of iterations required by ISTA and FISTA grows much faster than it does for the FW-T method when $\rho$ decreases. However, unlike the last experiment, where the number of frames in each dataset is at least several thousand, the number of frames is below several thousand.
Comparisons of FW-T, ISTA and FISTA on surveillance video data. The advantage of our FW-T method becomes prominent when the data are at large scale and compressed (i.e. the small $\rho$ scenarios).

<table>
<thead>
<tr>
<th>Data</th>
<th>$\rho$</th>
<th>FW-T</th>
<th>ISTA</th>
<th>FISTA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>iter.</td>
<td>time</td>
<td>iter.</td>
</tr>
<tr>
<td>Lobby</td>
<td>1.0</td>
<td>96</td>
<td>1.94e+02</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>(20480 × 1000)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>104</td>
<td>2.33e+02</td>
<td>216</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>133</td>
<td>3.12e+02</td>
<td>380</td>
</tr>
<tr>
<td>Campus</td>
<td>1.0</td>
<td>45</td>
<td>1.56e+02</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>(20480 × 1439)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>44</td>
<td>1.57e+02</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>41</td>
<td>1.39e+02</td>
<td>218</td>
</tr>
<tr>
<td>Escalator</td>
<td>1.0</td>
<td>81</td>
<td>7.40e+02</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>(20800 × 3417)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>80</td>
<td>7.35e+02</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>82</td>
<td>7.68e+02</td>
<td>162</td>
</tr>
<tr>
<td>Mall</td>
<td>1.0</td>
<td>38</td>
<td>4.70e+02</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>(81920 × 1286)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>35</td>
<td>4.58e+02</td>
<td>171</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>44</td>
<td>5.09e+02</td>
<td>308</td>
</tr>
<tr>
<td>Restaurant</td>
<td>1.0</td>
<td>70</td>
<td>5.44e+02</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>(19200 × 3055)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>74</td>
<td>5.51e+02</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>76</td>
<td>5.73e+02</td>
<td>308</td>
</tr>
<tr>
<td>Hall</td>
<td>1.0</td>
<td>60</td>
<td>6.33e+02</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>(25344 × 3584)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>62</td>
<td>6.52e+02</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>70</td>
<td>7.43e+02</td>
<td>144</td>
</tr>
<tr>
<td>Airport</td>
<td>1.0</td>
<td>130</td>
<td>6.42e+03</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>(25344 × 15730)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>136</td>
<td>6.65e+03</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>154</td>
<td>7.72e+03</td>
<td>77</td>
</tr>
<tr>
<td>Square</td>
<td>1.0</td>
<td>179</td>
<td>1.24e+04</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>(19200 × 28181)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>181</td>
<td>1.26e+04</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>191</td>
<td>1.31e+04</td>
<td>78</td>
</tr>
</tbody>
</table>
Fig. 3. Surveillance videos. The videos from top to bottom are respectively Lobby, Campus, Escalator, Mall, Restaurant, Hall, Airport and Square. The left panel presents videos with full observation ($\rho = 1$) and the right one presents videos with partial observation ($\rho = 0.6$). Visually, the low-rank component successfully recovers the background and the sparse one captures the moving objects (e.g. vehicles, pedestrians) in the foreground.

of frames here is just 65. This prevents the FW-T method from significantly benefiting from its linear per-iteration cost and consequently, while FW-T still outperforms ISTA for values of $\rho \leq 0.7$, the FISTA method is always the fastest. In Figure 4, the original images, the low-rank and the sparse parts produced by the FW-T method are presented. Visually, the recovered low-rank component is smoother and better conditioned for face recognition than the original image, while the sparse component corresponds to shadows and specularities.

6. Discussion. In this paper, we have proposed scalable algorithms called Frank-Wolfe-Projection (FW-P) and Frank-Wolfe-Thresholding (FW-T) for norm constrained
Table 2
Comparisons of FW-T, ISTA and FISTA on YaleB face data. The number of frames, 65, is relatively small for this application. This disables the FW-T method to significantly benefit from its linear per-iteration cost and consequently the FISTA method consistently has a better performance.

<table>
<thead>
<tr>
<th>Data</th>
<th>( \rho )</th>
<th>FW-T</th>
<th>ISTA</th>
<th>FISTA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>iter.</td>
<td>time</td>
<td>iter.</td>
</tr>
<tr>
<td>YaleB01</td>
<td>1.0</td>
<td>65</td>
<td>34.0</td>
<td>49</td>
</tr>
<tr>
<td>(32256 \times 65)</td>
<td>0.9</td>
<td>68</td>
<td>35.6</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>79</td>
<td>42.2</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>76</td>
<td>39.9</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>71</td>
<td>37.5</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>80</td>
<td>40.5</td>
<td>182</td>
</tr>
<tr>
<td>YaleB02</td>
<td>1.0</td>
<td>64</td>
<td>34.6</td>
<td>51</td>
</tr>
<tr>
<td>(32256 \times 65)</td>
<td>0.9</td>
<td>64</td>
<td>26.8</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>71</td>
<td>33.9</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>71</td>
<td>31.3</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>73</td>
<td>36.6</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>63</td>
<td>28.0</td>
<td>177</td>
</tr>
<tr>
<td>YaleB03</td>
<td>1.0</td>
<td>62</td>
<td>26.0</td>
<td>49</td>
</tr>
<tr>
<td>(32256 \times 65)</td>
<td>0.9</td>
<td>71</td>
<td>27.5</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>69</td>
<td>30.0</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>78</td>
<td>31.5</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>73</td>
<td>28.7</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>70</td>
<td>28.0</td>
<td>181</td>
</tr>
<tr>
<td>YaleB04</td>
<td>1.0</td>
<td>63</td>
<td>28.5</td>
<td>47</td>
</tr>
<tr>
<td>(32256 \times 65)</td>
<td>0.9</td>
<td>67</td>
<td>28.7</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>68</td>
<td>31.7</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>69</td>
<td>30.7</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>71</td>
<td>29.4</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>74</td>
<td>29.4</td>
<td>174</td>
</tr>
</tbody>
</table>

and penalized versions of CPCP. Essentially, these methods combine classical ideas in Frank-Wolfe and Proximal methods to achieve linear per-iteration cost, \( O(1/k) \) convergence in function value and practical efficiency in updating the sparse component. Extensive numerical experiments were conducted on computer vision related applications of CPCP, which demonstrated the great potential of our methods for dealing with problems of very large scale. Moreover, the general idea of leveraging different methods to deal with different functions may be valuable for other demixing problems.

We are also aware that though our algorithms are extremely efficient in the beginning iterations and quickly arrive at an approximate solution of practical significance, they become less competitive in solutions of very high accuracy, due to the nature
of Frank-Wolfe. That suggests further hybridization under our framework (e.g. using nonconvex approaches to handle the nuclear norm) might be utilized in certain applications (see [40] for research in that direction).

**Appendix A. A Useful Recurrence.**

We first present an elementary but useful fact about a real sequence, that has been often exploited in the convergence proofs for FW-type algorithms (see e.g. [25]).

**Lemma A.1.** Consider a real sequence \( \{a^k\} \). Suppose \( \{a^k\} \) satisfies the following recursive relation:

\[
\tag{A.1}
a^{k+1} \leq \frac{k}{k+2} a^k + \left( \frac{2}{k+2} \right)^2 C, \quad \text{for } k = 0, 1, 2, \ldots ,
\]

where \( C \) is a constant. Then for any \( k = 1, 2, 3, \ldots \), we have \( a^k \leq \frac{4C}{k+2} \), and hence \( \lim_{k \to \infty} a^k = 0 \).

**Proof.** The proof is by induction. Clearly, from (A.1), we have \( a^1 \leq C \leq \frac{4C}{4+2} \) as
the base case. For any fixed \( k \geq 1 \), assume that \( a^k \leq \frac{4C}{k+2} \). Then by (A.1),

\[
(A.2) \quad a^{k+1} \leq \frac{k}{k+2} a^k + \left( \frac{2}{k+2} \right)^2 \leq \frac{k}{k+2} + \frac{4C}{k+2} + \frac{4C}{(k+2)^2}
\]

Therefore, by induction, we have proved the claim. □

Appendix B. Proofs from Section 2.

B.1. Proof of Theorem 2.1. Proof. For \( k = 0, 1, 2, \ldots \), we have

\[
(2.1) \quad h(x^{k+1}) \leq h(x^k) + \gamma (v^k - x^k) \\
\leq h(x^k) + \gamma \langle \nabla h(x^k), v^k - x^k \rangle + \frac{L\gamma^2}{2} \|v^k - x^k\|^2 \\
\leq h(x^k) + \gamma \langle \nabla h(x^k), v^k - x^k \rangle + \frac{\gamma^2 LD^2}{2} \\
\leq h(x^k) + \gamma \langle \nabla h(x^k), x^* - x^k \rangle + \frac{\gamma^2 LD^2}{2} \\
\leq h(x^k) + \gamma (h(x^k) - h(x^*)) + \frac{\gamma^2 LD^2}{2},
\]

where the second inequality holds since \( \nabla h(\cdot) \) is \( L \)-Lipschitz continuous; the third line follows because \( D \) is the diameter for the feasible set \( D \); the fourth inequality follows from \( v^k \in \arg\min_{u \in D} \langle u, \nabla h(x^k) \rangle \) and \( x^* \in D \); the last one holds since \( h(\cdot) \) is convex.

Rearranging terms in (B.2), one obtains that for \( k = 0, 1, 2, \ldots \),

\[
(B.3) \quad h(x^{k+1}) - h(x^*) \leq (1 - \gamma) (h(x^k) - h(x^*)) + \frac{\gamma^2 LD^2}{2}.
\]

Therefore, by Lemma A.1,

\[
h(x^k) - h(x^*) \leq \frac{2LD^2}{k+2}, \quad \text{for } k = 1, 2, 3, \ldots
\]

□

B.2. Proof of Theorem 2.2. Proof. For notational convenience, we denote \( h^k = h(x^k) \), \( \Delta^k = h(x^k) - h(x^*) \), \( d^k = d(x^k) \), \( C = 2LD^2 \), \( B = K + 2 \), \( k = \lfloor \frac{1}{2} B \rfloor - 1 \), \( \mu = \lfloor \frac{1}{2} B \rfloor / B \).

Suppose on the contrary that

\[
(B.4) \quad d^k > \frac{3C}{B}, \quad \text{for all } k \in \left\{ \lfloor \frac{1}{2} B \rfloor - 1, \lfloor \frac{1}{2} B \rfloor, \ldots, K \right\}.
\]

From (B.1), we know that for any \( k \geq 1 \)

\[
\Delta^{k+1} \leq \Delta^k + \gamma \langle \nabla h(x^k), v^k - x^k \rangle + \frac{\gamma^2 LD^2}{2} = \Delta^k - \frac{2d^k}{k+2} + \frac{C}{(k+2)^2}.
\]
Therefore, by using (B.5) repeatedly, one has

\[
\Delta^{K+1} \leq \Delta^k - \frac{2d^k}{k+2} + \frac{C}{(k+2)^2} \leq \Delta^k - \frac{6C}{B} \sum_{k=k}^{K} \frac{1}{k+2} + \frac{C}{(k+2)^2} = \Delta^k - \frac{6C}{B} \sum_{k=k+2}^{B} \frac{1}{k} + \frac{C}{(k+2)^2} \leq \frac{C}{\mu B} \left( \frac{6C}{B} \left( 1 - \frac{1}{k} \right) + \frac{C}{(k+2)^2} \right) = \frac{C}{\mu B} \left( 2 - 6\mu(1 - \mu) \right)
\]

which implies the result.

where the second line is due to our assumption (B.4); the fourth line holds since \( \Delta^k \leq \frac{C}{k+2} \) by Theorem 1, and \( \sum_{k=a}^{b} \frac{1}{k} \leq \frac{\log(b+1)}{a} \) for any \( b \geq a > 1 \).

Now define \( \phi(x) = 2 - 6\mu(1 - x) - x \). Clearly \( \phi(\cdot) \) is convex. Since \( \phi\left( \frac{1}{2} \right) = \phi\left( \frac{2}{3} \right) = 0 \), we have \( \phi(x) \leq 0 \) for any \( x \in [\frac{1}{2}, \frac{2}{3}] \). As \( \mu = \left[ \frac{1}{2} B \right] / B \in [\frac{1}{2}, \frac{2}{3}] \), from (B.6), we have

\[
\Delta^{K+1} = h(x^{K+1}) - h(x^*) \leq \frac{C}{\mu B} \phi(\mu) \leq 0,
\]

which is a contradiction. \( \square \)

**Appendix C. Proofs from Section 3.**

**C.1. Proof of Lemma 3.1.** *Proof.* From (3.2) and (3.3), we have

\[
\| \nabla l(L, S) - \nabla l(L', S') \|_F^2 = 2 \| \mathcal{P}_Q [L + S - M] - \mathcal{P}_Q [L' + S' - M] \|_F^2 = 2 \| \mathcal{P}_Q [L + S] - \mathcal{P}_Q [L' + S'] \|_F^2 \leq 2 \| L + S - L' - S' \|_F^2 \leq 4 \| L - L' \|_F^2 + 4 \| S - S' \|_F^2 = 4 \| (L, S) - (L', S') \|_F^2,
\]

which implies the result. \( \square \)

**C.2. Proof of Lemma 3.2.** *Proof.* For any \( Z = (L, S) \) and \( Z' = (L', S') \in \mathcal{D}, \)

\[
\| Z - Z' \|_F^2 \leq \| L - L' \|_F^2 + \| S - S' \|_F^2 \leq (\| L \|_F + \| L' \|_F)^2 + (\| S \|_F + \| S' \|_F)^2 \leq 4\mathcal{R}_L^2 + 4\mathcal{R}_S^2.
\]

\( \square \)

**C.3. Proof of Theorem 3.3.** *Proof.* Substituting \( L = 2 \) (Lemma 3.1) and \( D \leq 2\sqrt{\mathcal{R}_L^2 + \mathcal{R}_S^2} \) (Lemma 3.2) into Theorems 2.1 and 2.2, we can easily obtain the results. \( \square \)

**Appendix D. Proofs from Section 4.**
D.1. Proof of Lemma 4.1. Proof. From (4.3) and (4.4), it follows as in C.1 that
\[ \| \nabla g(L, S, t_L, t_S) - \nabla g(L', S', t_L', t_S') \|_F^2 \leq 4 \| L - L' \|_F^2 + 4 \| S - S' \|_F^2 \]
\[ \leq 4 \| (L, S, t_L, t_S) - (L', S', t_L', t_S') \|_F^2, \]
which implies the result. □

D.2. Proof of Lemma 4.2. Proof. Since for any \( Z = (L, S, t_L, t_S) \), \( Z' = (L', S', t_L', t_S') \in D \), we have
\[
\| Z - Z' \|_F^2 = \| L - L' \|_F^2 + \| S - S' \|_F^2 + (t_L - t_L')^2 + (t_S - t_S')^2 \\
\leq (\| L \|_F + \| L' \|_F)^2 + (\| S \|_F + \| S' \|_F)^2 + (t_L - t_L')^2 + (t_S - t_S')^2 \\
\leq (U_L + U_L)^2 + (U_S + U_S)^2 + U_L^2 + U_S^2 \\
= 5(U_L^2 + U_S^2),
\]
which implies the result. □

D.3. Proof of Corollary 4.3. Proof. Applying Theorem 2.1 with parameters calculated in Lemmas 4.1 and 4.2, we directly have
\[
g(x^k) - g(x^*) \leq \frac{2 \cdot 2 \cdot \left( \sqrt{5(U_L^2 + U_S^2)} \right)^2}{k + 2} = \frac{20(U_L^2 + U_S^2)}{k + 2}.
\]
A more careful calculation below slightly improves the constant in (D.1).
\[
g(x^{k+1}) = g(x^k + \gamma(v^k - x^k)) \\
\leq g(x^k) + \gamma \langle \nabla g(x^k), v^k - x^k \rangle + \gamma^2 \| V^k_L - L \|_F^2 + \gamma^2 \| V^k_S - S \|_F^2 \\
\leq g(x^k) + \gamma \langle \nabla g(x^k), v^k - x^k \rangle + 4\gamma^2(U_L^2 + U_S^2),
\]
where the second line holds by noting that \( g \) is only linear in \( t_L \) and \( t_S \); the last line holds as
\[
\| V^k_L - L \|_F^2 \leq (\| V^k_L \|_F + \| L \|_F)^2 \leq (U_L + U_L)^2 = 4U_L^2, \quad \text{and} \\
\| V^k_S - S \|_F^2 \leq (\| V^k_S \|_F + \| S \|_F)^2 \leq (U_S + U_S)^2 = 4U_S^2.
\]
Following the arguments in the proof of Theorem 1 with (B.1) replaced by (D.2), we can easily obtain that
\[
g(x^k) - g(x^*) \leq \frac{16(U_L^2 + U_S^2)}{k + 2}.
\]
□

D.4. Proof of Lemma 4.4. Proof. Since \((L^k, S^k, t_L^k, t_S^k)\) is always feasible to the quadratic program (4.18),
\[
g(L^{k+1}, S^{k+1}, t_L^{k+1}, t_S^{k+1}) \leq g(L^k, S^k, t_L^k, t_S^k).
\]
Based on (4.15), the threshold step (line 6 in Algorithm 3) can be written as

\[ S^{k+1} = \arg \min_S \tilde{g}^{k+\frac{1}{2}}(S) = \frac{1}{2} \left\| P_Q(L^{k+\frac{1}{2}} + S^{k+\frac{1}{2}} - M) \right\|_F^2 + \lambda_L L^{k+\frac{1}{2}} + \lambda_S \| S \|_1 + \langle P_Q(L^{k+\frac{1}{2}} + S^{k+\frac{1}{2}} - M), S - S^{k+\frac{1}{2}} \rangle + \frac{1}{2} \left\| S - S^{k+\frac{1}{2}} \right\|_F^2. \]

The following properties of \( \tilde{g}^{k+\frac{1}{2}}(\cdot) \) can be easily verified

\[ \tilde{g}^{k+\frac{1}{2}}(S^{k+\frac{1}{2}}) = g(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, \| S^{k+\frac{1}{2}} \|_1) \leq g(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}); \]

\[ \tilde{g}^{k+\frac{1}{2}}(S) \geq g(L^{k+\frac{1}{2}}, S, t_L^{k+\frac{1}{2}}, \| S \|_1), \quad \text{for any } S. \]

Therefore, we have

\[ g(L^{k+1}, S^{k+1}, t_L^{k+1}, t_S^{k+1}) = g(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \leq g(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \]

\[ \leq \tilde{g}^{k+\frac{1}{2}}(S^{k+\frac{1}{2}}) \leq g(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \]

Combining (D.3) and (D.4), we obtain

\[ g(L^{k+1}, S^{k+1}, t_L^{k+1}, t_S^{k+1}) \leq g(L^{k+\frac{1}{2}}, S^{k+\frac{1}{2}}, t_L^{k+\frac{1}{2}}, t_S^{k+\frac{1}{2}}) \leq g(L^k, S^k, t_L^k, t_S^k). \]

\[ \square \]

**D.5. Proof of Theorem 4.5.** For notational convenience, we denote

\[ x^k = (L^k, S^k, t_L^k, t_S^k), \quad x^* = (L^*, S^*, t_L^*, t_S^*) \quad \text{and} \quad v^k = (V_L^k, V_S^k, V_L^{t_L^k}, V_S^{t_S^k}). \]

For any point \( x = (L, S, t_L, t_S) \in \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \times \mathbb{R} \), we adopt the notation that \( L[x] = L, S[x] = S, t_L[x] = t_L, \) and \( t_S[x] = t_S. \)

Since \( g(x^k) - g(x^*) \leq \frac{4(t_L^0 + t_S^0)^2}{k+2} \) can be easily established following the proof of Corollary 4.3, below we will focus on the proof of \( g(x^k) - g(x^*) \leq \frac{4(t_L^0 + t_S^0)^2}{k+2} \).

**Proof.** Let us first make two simple observations.

Since \( f(L^*, S^*) \leq g(L^k, S^k, t_L^k, t_S^k) \), we have

\[ (D.5) \quad U_L^k = g(L^k, S^k, t_L^k, t_S^k)/\lambda \geq t_L^* \quad \text{and} \quad U_S^k = g(L^k, S^k, t_L^k, t_S^k)/\lambda \geq t_S^*. \]

Therefore, our \( U_L^k \) and \( U_S^k \) always bound \( t_L^* \) and \( t_S^* \) from above.

From Lemma 4.4, \( g(L^k, S^k, t_L^k, t_S^k) \) is non-increasing,

\[ f(L^k, S^k) \leq g(L^k, S^k, t_L^k, t_S^k) \leq g(L^0, S^0, t_L^0, t_S^0), \]

which implies that \( (L^k, S^k) \subseteq \overline{B(r_L^0)} \times \overline{B(r_S^0)} \), i.e. \( \| L^k \|_F \leq r_L^* \) and \( \| S^k \|_F \leq r_S^* \).

Let us now consider the \( k \)-th iteration. Similar to the proof in [24], we introduce the auxiliary point \( v_{k}^* = (t_L^k V_L^{k}, t_S^k V_S^{k}, t_L^k V_L^{k}, t_S^k V_S^{k}) \). Then based on our argument for (4.13), it can be easily verified that

\[ (L[v_L^k], t_L[v_L^k]) = \arg \min_{\| v_L \| \leq v_L^k} g_L(V_L, v_L) \]

\[ (S[v_S^k], t_S[v_S^k]) = \arg \min_{\| v_S \| \leq v_S^k} g_S(V_S, v_S). \]

\[ \square \]
Recall $\gamma = \frac{2}{\lambda + 2}$. We have

\[
\begin{align*}
g(x^{k+1}) & \leq g(x^k + \gamma (v_k^+ - x^k)) \\
& \leq g(x^k) + \gamma (\nabla g(x^k), v_k^+ - x^k) + \gamma^2 \left( \|L[v_k^+] - L[x^k]\|_F^2 + \|S[v_k^+] - S[x^k]\|_F^2 \right) \\
& \leq g(x^k) + \gamma \left( g_L(L[v_k^+] - x^k), t_L[v_k^+] - x^k \right) + g_S(S[v_k^+] - x^k), t_S[v_k^+] - x^k)) \\
& \quad + \gamma^2 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right) \\
& \leq g(x^k) + \gamma \left( g_L(L[x^*] - x^k), t_L[x^*] - x^k \right) + g_S(S[x^*] - x^k), t_S[x^*] - x^k)) \\
& \quad + \gamma^2 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right) \\
& = g(x^k) + \gamma (\nabla g(x^k), x^* - x^k) + \gamma^2 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right) \\
& \leq g(x^k) + \gamma \left( g(x^*) - g(x^k) \right) + \gamma^2 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right),
\end{align*}
\]

where the first inequality holds since $x^k + \gamma (v_k^+ - x^k)$ is feasible to the quadratic program (4.18) while $x^{k+\frac{1}{2}}$ minimizes it; the third inequality is due to the facts that

\[
\begin{align*}
& \|L[v_k^+] - L[x^k]\|_F \leq \|L[v_k^+]\|_F + \|L[x^k]\|_F \leq \|L[v_k^+]\|_* + \|L[x^k]\|_F \leq t_L^* + r_L^* \\
& \|S[v_k^+] - S[x^k]\|_F \leq \|S[v_k^+]\|_F + \|S[x^k]\|_F \leq \|S[v_k^+]\|_1 + \|S[x^k]\|_F \leq t_S^* + r_S^*;
\end{align*}
\]

the fourth inequality holds as $(L[x^*], t_L[x^*])$ and $(S[x^*], t_S[x^*])$ are respectively feasible to (D.6) and (D.7) while $(L[v_k^+], t_L[v_k^+])$ and $(S[v_k^+], t_S[v_k^+])$ respectively minimize (D.6) and (D.7).

Therefore, we obtain

\[
g(x^{k+\frac{1}{2}}) - g(x^*) \leq (1 - \gamma) \left( g(x^k) - g(x^*) \right) + \gamma^2 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right).
\]

Moreover, by Lemma 4.4, we have

\[
g(x^{k+1}) \leq g(x^{k+\frac{1}{2}}).
\]

Thus, we obtain the recurrence

\[
g(x^{k+1}) - g(x^*) \leq (1 - \gamma) \left( g(x^k) - g(x^*) \right) + \gamma^2 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right).
\]

Applying Lemma A.1 establishes that

\[
g(L^k, S^k, t_L^k, t_S^k) - g(L^*, S^*, t_L^*, t_S^*) \leq \frac{4 \left( (t_L^* + r_L^*)^2 + (t_S^* + r_S^*)^2 \right)}{k + 2}.
\]

\[\square\]

**D.6. Proof of Theorem 4.6.** Proof. Define $\Delta^k = g(x^k) - g(x^*)$. Following (D.2), we have

\[
\Delta^{k+1} \leq \Delta^k + \gamma \left( \nabla g(x^k), v^k - x^k \right) + 4\gamma^2 \left( (U_L^0)^2 + (U_S^0)^2 \right).
\]

Then following the arguments in the proof of Theorem 2 with (B.5) replaced by (D.8), we can easily obtain the result. \[\square\]

**Appendix E. ISTA & FISTA for problem (1.5).** *Iterative soft-thresholding algorithm* (ISTA), is an efficient way to tackle unconstrained nonsmooth optimization problem especially at large scale. ISTA follows the general idea by iteratively
minimizing an upper bound of the original objective. In particular, when applied to problem (1.5) of our interest, ISTA updates \((L, S)\) for the \(k\)-th iteration by solving

\[
(E.1) \quad (L^{k+1}, S^{k+1}) = \arg \min_{L,S} \left\{ \left( \nabla_L l(L^k, S^k) \right), \left( \nabla_S l(L^k, S^k) \right) \right\}^2 + \frac{L_f}{2} \| (L - L^k) - (S - S^k) \|^2_F + \lambda_L \| L \|_* + \lambda_S \| S \|_1
\]

Here \(L_f = 2\) denotes the Lipschitz constant of \(\nabla l(L, S)\) with respect to \((L, S)\), and \(\nabla_L l(L^k, S^k) = \nabla_S l(L^k, S^k) = \mathcal{P}_\Omega [L^k + S^k - M]\). Since \(L\) and \(S\) are decoupled in (E.1), equivalently we have

\[
(E.2) \quad L^{k+1} = \arg \min_L \left\| L - \left( L^k - \frac{1}{2} \mathcal{P}_\Omega [L^k + S^k - M] \right) \right\|^2_F + \lambda_L \| L \|_*,
\]

\[
(E.3) \quad S^{k+1} = \arg \min_S \left\| S - \left( S^k - \frac{1}{2} \mathcal{P}_\Omega [L^k + S^k - M] \right) \right\|^2_F + \lambda_S \| S \|_1.
\]

The solution to problem (E.3) can be given explicitly in terms of the proximal mapping of \(\| \cdot \|_1\) as introduced in Section 2.2, i.e.,

\[
S^{k+1} = \mathcal{T}_{\lambda_S/2} \left[ S^k - \frac{1}{2} \mathcal{P}_\Omega [L^k + S^k - M] \right].
\]

For a matrix \(X\) and any \(\tau \geq 0\), let \(\mathcal{D}_\tau(X)\) denote the singular value thresholding operator \(\mathcal{D}_\tau(X) = U \mathcal{T}_\tau(\Sigma) V^*\), where \(X = U \Sigma V^*\) is the singular value decomposition of \(X\). It is not difficult to show \([41, 42]\) that the solution to problem (E.2) can be given explicitly by

\[
L^{k+1} = \mathcal{D}_{\lambda_L/2} \left[ L^k - \frac{1}{2} \mathcal{P}_\Omega [L^k + S^k - M] \right].
\]

Algorithm 7 summarizes our ISTA implementation for problem (2).

**Algorithm 7** ISTA for problem (2)

1. **Initialization**: \(L^0 = 0, S^0 = 0\);
2. for \(k = 0, 1, 2, \ldots\) do
3. \(L^{k+1} = \mathcal{D}_{\lambda_L/2} \left[ L^k - \frac{1}{2} \mathcal{P}_\Omega [L^k + S^k - M] \right];\)
4. \(S^{k+1} = \mathcal{T}_{\lambda_S/2} \left[ S^k - \frac{1}{2} \mathcal{P}_\Omega [L^k + S^k - M] \right];\)
5. end for

Regarding ISTA’s speed of convergence, it can be proved that \(f(L^k, S^k) - f^* = O(1/k)\), where \(f^*\) denotes the optimal value of problem (2).

*Fast iterative soft-thresholding algorithm* (FISTA) introduced in \([20]\), is an accelerated version of ISTA, which incorporate a momentum step borrowed from Nesterov’s optimal gradient scheme \([43]\). For FISTA, a better convergence result, \(f(L^k, S^k) - f^* = O(1/k^2)\), can be achieved with a cost per iteration that is comparable to ISTA. Algorithm 8 summarizes our FISTA implementation for problem (2).
Algorithm 8 FISTA for problem (2)

1: Initialization: $L^0 = \tilde{L}^0 = 0$, $S^0 = \tilde{S}^0 = 0$, $t_0 = 1$;
2: for $k = 0, 1, 2, \cdots$ do
3: $L^{k+1} = D_{\lambda_L/2} \left( L^k - \frac{1}{2} \mathcal{P}_\Omega [ L^k + \tilde{S}^k - M ] \right)$;
4: $S^{k+1} = T_{\lambda_S/2} \left( S^k - \frac{1}{2} \mathcal{P}_\Omega [ L^k + \tilde{S}^k - M ] \right)$;
5: $t_{k+1} = 1 + \frac{1}{2} \sqrt{1 + 4 (t_k)^2}$;
6: $\hat{L}^{k+1} = L^{k+1} + \frac{t_k - 1}{t_{k+1}} ( L^{k+1} - L^k )$;
7: $\hat{S}^{k+1} = S^{k+1} + \frac{t_k - 1}{t_{k+1}} ( S^{k+1} - S^k )$;
8: end for

Partial SVD. In each iteration of either ISTA or FISTA, we only need those singular values that are larger than $\lambda_S/2$ and their corresponding singular vectors. Therefore, a partial SVD can be utilized to reduce the computational burden of a full SVD. Since most partial SVD software packages (e.g. PROPACK [44]) require specifying in advance the number of top singular values and singular vectors to compute, we heuristically determine this number (denoted as $sv^k$ at iteration $k$). Specifically, let $d = \min \{ m, n \}$, and $svp^k$ denote the number of computed singular values that were larger than $\lambda_S/2$ in the $k$-th iteration. Similar to [17], in our implementation, we start with $sv^0 = d/10$, and adjust $sv^k$ dynamically as follows:

$$sv^{k+1} = \begin{cases} \min \{ sv^k + 1, d \} & \text{if } sv^k < sv^k \\ \min \{ sv^k + \text{round}(0.05d), d \} & \text{otherwise.} \end{cases}$$

REFERENCES


