2

Algorithms for Stable PCA

Necdet Serhat Aybat
Penn State University, USA

2.1 Introduction

Principal component analysis (PCA) is an essential tool in applications ranging from image and video processing, web data analysis to bioinformatics. PCA seeks the best low-dimensional approximation to high-dimensional data. In particular, let the data matrix $A \in \mathbb{R}^{m\times n}$ is of the form $A := L^o + N^o$, where L^o is a *low-rank* matrix, i.e., $\mathbf{rank}(L^o) \ll \min\{m,n\}$, and N^o is a dense small-magnitude noise matrix. PCA generates a rank-$\bar{r}$ approximation to L^o by solving $\min_L\{\|L-A\| : \ \mathbf{rank}(L) \leq \bar{r}\}$, which requires computing one singular value decomposition (SVD) of A, where $\|X\|$ denotes the spectral norm of $X \in \mathbb{R}^{m\times n}$, i.e., maximum singular value of X. However, when the observed data is corrupted by *gross errors*, classical PCA becomes impractical because even a single grossly corrupted observation can destroy the underlying low-rank structure in such a way that it cannot be recovered using PCA. To remedy this shortcoming, an extended model called robust PCA (RPCA) was considered by Wright et al. [62], Candès et al. [12] and Chandrasekaran et al. [13] when the gross errors are *sparse*. In this model it is assumed that the data matrix $A \in \mathbb{R}^{m\times n}$ is of the form $A := L^o + S^o$, where L^o is a low-rank matrix as in PCA, and S^o is a sparse gross error matrix, i.e., the number of nonzero elements is small. This model attempts to decompose the observed matrix A as a sum of two matrices: a low-rank approximation to A, and a sparse matrix that captures the gross errors. Under certain incoherence assumptions on L^o, and randomness assumption on the sparsity pattern of S^o, Candès et al. [12] showed that one can recover (L^o, S^o) exactly with very high probability as the unique solution of a convex optimization problem, called *Principal Component Pursuit* (PCP).

On the other hand, the observations in real-life applications are often corrupted by noise, possibly affecting every entry of the data matrix, which can be represented with a dense noise matrix N^o. Hence, suppose that the data matrix A can be decomposed as $A = L^o + S^o + N^o$, where L^o and S^o are low-rank and sparse matrices as in robust PCA. However, introducing

a third term in the decomposition renders PCP inapplicable as A is not a sum of low-rank and sparse components anymore. To overcome this issue and achieve stable PCA with noisy observations, Zhou et al. [65] proposed another convex model called *stable* PCP (SPCP), and showed that in the presence of dense noise N^o solving SPCP guarantees stable recovery of L^o with an error bound proportional to the overall noise magnitude $\|N^o\|_F$. In the following sections, after we briefly review the statistical recovery guarantees of SPCP and mention some other alternative models for stable recovery in the presence of entry-wise noise, we mainly focus on reviewing iterative algorithms proposed in the literature to solve SPCP problem and its variants; and discuss their their key characteristics such as convergence rate, and computational complexity per iteration. First, we go over the notation adopted throughout the chapter.

2.1.1 Notation

Let $\mathbb{R}_+ := \{t \in \mathbb{R} : t \geq 0\}$, $\mathbb{R}_{++} := \mathbb{R}_+ \setminus \{0\}$ and sgn(.) denote the signum function, i.e., given $x \in \mathbb{R}$, sgn(x) is equal to -1, 0, and 1, if $x < 0$, $x = 0$, or $x > 0$, respectively; moreover, for any $x \in \mathbb{R}^n$ and $X \in \mathbb{R}^{m\times n}$, both sgn$(x)$ and sgn(X) operate componentwise. $e_j \in \mathbb{R}^n$ denotes the unit vector with j-th entry is 1, $\mathbf{0}_n$ and $\mathbf{1}_n$ denote vectors in $\mathbb{R}^n$ with all entries equal to 0 and 1, respectively; similarly, $\mathbf{0}_{m\times n}$ and $\mathbf{1}_{m\times n}$ denote matrices in $\mathbb{R}^{m\times n}$ with all entries equal to 0 and 1, respectively. $\mathbf{I}_n$ represents $n \times n$ identity matrix, and given $\sigma \in \mathbb{R}^n$, $\mathbf{diag}(\sigma)$ represents the $n \times n$ diagonal matrix with its diagonal equal to σ. Moreover, given $\mathcal{X} = \{x^{(i)}\}_{i=1}^n \subset \mathbb{R}^m$, $[x^{(1)}, \ldots, x^{(n)}] \in \mathbb{R}^{m\times n}$ represents the matrix of which columns are the elements of $\mathcal{X}$; and $[x^{(1)}; \ldots; x^{(n)}] \in \mathbb{R}^{mn}$ represents a long vector obtained by vertically stacking the elements of $\mathcal{X}$.

Given $X \in \mathbb{R}^{m\times n}$, the nuclear norm $\|X\|_* := \sum_{i=1}^{\mathbf{rank}(X)} \sigma_i(X)$, and the spectral norm $\|X\| := \max\{\sigma_i(X) : \ i = 1, \ldots, \mathbf{rank}(X)\}$, where $\{\sigma_i(X)\}_{i=1}^{\mathbf{rank}(X)} \subset \mathbb{R}_{++}$ denotes the singular values of X; the ℓ_1-norm $\|X\|_1 := \sum_{i=1}^m \sum_{j=1}^n |X_{ij}|$; the ℓ_∞-norm $\|X\|_\infty := \max\{|X_{ij}| : i = 1, \ldots, m, \ j = 1, \ldots, n\}$; the Frobenius norm $\|X\|_F := \sqrt{\sum_{i=1}^m \sum_{j=1}^n Z_{ij}^2}$; and the ℓ_0-"norm" $\|X\|_0$ denotes the number of *non-zero* components of the matrix X. Let $\Omega \subset \{(i,j) : \ i = 1, \ldots, m, \ j = 1, \ldots, n\}$, and define the projection operator $\pi_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ as follows:

$$(\pi_\Omega(X))_{ij} = \begin{cases} X_{ij}, & (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The operator π_{Ω^c} is defined similarly for the complement set Ω^c. Note that the adjoint operator $\pi_\Omega^* = \pi_\Omega$. Given $X, \bar{X} \in \mathbb{R}^{m\times n}$, $X \odot \bar{X}$ denotes componentwise multiplication. Throughout the chapter, $\mathbb{P}(\mathcal{A})$ denotes the probability of some random event $\mathcal{A}$, and the acronym i.i.d. is used for independent and identically distributed. Given a set $\mathcal{X} \subset \mathbb{R}^{m\times n}$, define its indicator function $\mathcal{I}_\mathcal{X} : \mathbb{R}^{m\times n} \to \mathbb{R} \cup \{+\infty\}$ such that

$$\mathcal{I}_\mathcal{X}(X) := \begin{cases} 0, & \text{if } X \in \mathcal{X}; \\ +\infty, & \text{o.w.} \end{cases}$$

Moreover, given a closed convex function $\psi : \mathbb{R}^{m\times n} \to \mathbb{R} \cup \{+\infty\}$, $\mathbf{prox}_\psi : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ denotes the proximal map of ψ, also called prox map of ψ, i.e., $\mathbf{prox}_\psi(X) := \min_Z\{\psi(Z) + \frac{1}{2}\|Z - X\|_F^2\}$.

2.1.2 Recovery Guarantees

Suppose that the data matrix $A \in \mathbb{R}^{m\times n}$ can be decomposed as

$$A = L^o + S^o + N^o, \tag{2.2}$$

where L^o is a *low-rank* matrix, i.e., $r = \mathbf{rank}(L^o) \ll \min\{m,n\}$, S^o is a *sparse* gross "error" matrix, i.e., $s := \|S^o\|_0 \leq mn$, and N^o is a dense noise matrix such that $\|N^o\|_F \leq \delta$ for some $\delta > 0$. In order to stably recover L^o in the presence of the dense noise matrix N^o, Zhou et al. [65] proposed solving the following convex optimization problem called *Stable Principal Component Pursuit* (SPCP):

$$(L^*, S^*) \in \underset{L,S\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \{\|L\|_* + \lambda\|S\|_1 : \ \|L+S-A\|_F \leq \delta\}, \tag{2.3}$$

where $\lambda := \frac{1}{\sqrt{\max\{m,n\}}}$. Note that if there is no noise on observations, i.e., $A = L^o + S^o$ and $N^o = \mathbf{0}_{m\times n}$, then $\delta = 0$ and the SPCP formulation in (2.3) is equivalent to the PCP formulation

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{\|L\|_* + \lambda\|S\|_1 : \ L+S=A\}, \tag{2.4}$$

which discussed in Chapter 1; hence, PCP is a special case of SPCP.

It is clear that if either L^o is sparse, or S^o is low-rank, then recovering unknown components (L^o, S^o) through solving SPCP is hopeless. On the other hand, let $L^o = U\Sigma V^\top$ denote the SVD of L^o, i.e., $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ and $\Sigma \in \mathbb{R}^{r\times r}$ such that $U^\top U = V^\top V = \mathbf{I}_r$ and $\Sigma = \mathbf{diag}(\sigma)$, where $\sigma \in \mathbb{R}^r_{++}$ denote the singular values of L^o; and suppose that L^o and S^o satisfy the following additional assumptions:

$$\max_{1\leq i\leq r} \|U^T e_i\|_2^2 \leq \frac{\mu r}{m}, \quad \max_{1\leq i\leq r} \|V^T e_i\|_2^2 \leq \frac{\mu r}{n}, \quad \|UV^T\|_\infty \leq \sqrt{\frac{\mu r}{mn}}, \tag{2.5}$$

for some $\mu > 0$, and the set of indices corresponding to non-zero components of S^o is distributed uniformly at random among all the subsets of cardinality s. These additional assumptions make sure that *a)* singular vectors of L^o are not sparse – thus, L^o is not sparse; *b)* S^o is not low-rank with high probability.

Under these assumptions, it is shown in [12] that if $N^o = \mathbf{0}_{m\times n}$, then there exists $c > 0$ such that solving the PCP problem (2.4) *exactly* recovers L^o and S^o, i.e., $(L^*, S^*) = (L^o, S^o)$ when $\delta = 0$ in (2.3), with probability of at least $1 - c\max\{m,n\}^{-10}$, provided

$$\mathbf{rank}(L^o) \leq \rho_r \mu^{-1} \min\{m,n\} \left(\log(\max\{m,n\})\right)^{-2} \quad \text{and} \quad \|S^o\|_0 \leq \rho_s mn \tag{2.6}$$

for some constants $\rho_r,\ \rho_s > 0$. Comparable to the *exact recovery* property of PCP when $N^o = \mathbf{0}_{m\times n}$, it is shown in [65] that under the assumptions stated above if $N^o \neq \mathbf{0}_{m\times n}$ such that $\|N^o\|_F \leq \delta$, then the solution (L^*, S^*) to the SPCP problem in (2.3) satisfies $\|L^o - L^*\|_F^2 + \|S^o - S^*\|_F^2 \leq Cmn\delta^2$ for some constant C with *high probability*.

In many applications, some of the entries of A may not be available. Let $\Omega \subset \{(i,j) : i = 1,\ldots,m,\ j = 1,\ldots,n\}$ be the index set of the observable entries of A. Suppose that Ω is chosen uniformly at random among subsets of cardinality pmn for some sufficiently large $p \in (0,1]$, L^o satisfies (2.5), S^o is such that for each $(i,j) \in \Omega$, $\mathbb{P}(S^o_{ij} = 0) = 1 - q$ independently of the others for some $q \in (0,1)$, and $N^o = \mathbf{0}_{m\times n}$. Then it is also shown in [12] that there exits $c > 0$ such that with probability at least $1 - c^{-10}$, solving the following modification of PCP,

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{\|L\|_* + \lambda\|S\|_1 : \ \pi_\Omega(L+S-A) = \mathbf{0}_{m\times n}\} \tag{2.7}$$

with $\lambda = \frac{1}{\sqrt{p\max\{m,n\}}}$, exactly recovers the true low-rank component L^o provided $\mathbf{rank}(L^o) \leq \rho_r \mu^{-1} \min\{m,n\} \left(\log(\max\{m,n\})\right)^{-2}$ and $q \leq q_s$ for some constants $\rho_r,\ q_s > 0$. For applications with both missing and noisy observations, i.e., $\Omega^c \neq \emptyset$ and $N^o \neq \mathbf{0}_{m\times n}$, Tao and Yuan [59] proposed recovering the low rank and sparse components of A by solving

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{\|L\|_* + \lambda\|S\|_1 : \ \|\pi_\Omega(L+S-A)\|_F \leq \delta\}. \tag{2.8}$$

2.1.3 Overview of algorithms

When there is no noise in the observations, i.e., $N^o = \mathbf{0}_{m\times n}$ and $A = L^o + S^o$, there are many efficient algorithms for solving the PCP problems in (2.4) and (2.7); on the other hand, in the presence of dense noise, i.e., $N^o \neq \mathbf{0}_{m\times n}$, surprisingly there is only very few methods that can efficiently deal with (2.8) directly. Table 2.1 shows an overview of the Stable Principal Component Pursuit methods that will be discussed in Section 2.2 and Section 2.3.

TABLE 2.1 Stable Principal Component Pursuit: An Overview of Algorithms.

Algorithms	Problem	Authors - Dates
ASALM & VASALM: multi-block ADMM type alg.	(2.8)	Tao and Yuan (2009) [59]
FALC: augmented Lagrangian alg.	(2.8)	Aybat et al. (2010) [5]
SPG: accelerated Nesterov-type alg. + smoothing	(2.8)	Aybat et al. (2011) [3]
PSPG: accelerated proximal gradient alg. + smoothing	(2.8)	Aybat et al. (2013) [3, 4]
ADMIP: increasing penalty ADMM alg.	(2.8)	Aybat et al. (2013) [3, 6]
Quasi Newton method	(2.55)	Aravkin et al. (2014) [1]
FWP: Frank-Wolfe + projected gradient alg.	(2.57)	Mu et al. (2014) [45]
Two-block ADMM alg. for non-negative SPCP	(2.54)	Huai et al. (2015) [35]
DFC: a parallel randomized matrix approximation alg.		Mackey et al. (2011) [43]
GoDec: alternating minimization alg.	(2.60)	Zhou and Tao (2011) [63]
AMS: inexact alternating minimization alg.	(2.61)	Hintermüller and Wu (2014) [33]
LMaFit: multi-block ADMM alg.	(2.62)	Shen et al. (2011) [56]
GreBsmo: multi-block alternating minimization alg.	(2.63)	Zhou and Tao (2013) [64]
AD-MoM: multi-block ADMM alg.	(2.65)	Mardani et al. (2011) [44]

When Zhou et al. proposed the SPCP problem in [65], instead of directly solving it with a customized algorithm for (2.3), they computed a solution to the problem in Lagrangian form:

$$\min_{L,S\in\mathbb{R}^{m\times n}} \|L\|_* + \lambda\|S\|_1 + \frac{\rho}{2}\|L+S-A\|_F^2. \tag{2.9}$$

for an appropriately chosen dual variable $\rho > 0$ using the accelerated proximal gradient (APG) algorithm in [40]. It follows from convex duality that for any given $\delta > 0$, there exists $\rho(\delta) \geq 0$ such that (2.9) is equivalent to (2.3). On the other hand, it should be emphasized that while $\delta > 0$ is usually readily available, and has some natural physical meaning, corresponding $\rho(\delta)$ is usually hard to guess, and tuning ρ requires cross-validation.

The first tailor-made algorithms for the SPCP problem in (2.8) is proposed by Tao and Yuan [59]: a *multi-block* ADMM algorithm, and its variant. While the former one is significantly faster in the numerical tests conducted in [59], it does not have theoretical convergence guarantees that the latter algorithm has. The first $\mathcal{O}(1/\epsilon)$ sublinear convergence rate result for a first-order algorithm customized to (2.8) is showed in [5]. The algorithm proposed in [5] by Aybat and Iyengar is an inexact augmented Lagrangian algorithm; although it is customized for minimizing constrained composite norm problems, it does not fully exploit the special constraint structure in (2.8). To utilize the structure in constraints, Aybat, Goldfarb, and Ma [4] showed that (2.8) can be equivalently formulated as

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{\|L\|_* + \lambda\|\pi_\Omega(S)\|_1 : \ \|L+S-\pi_\Omega(A)\|_F \leq \delta\}, \tag{2.10}$$

in the sense that if (L^*, S^*) is an optimal solution to (2.10), then $(L^*, \pi_\Omega(S^*))$ is an optimal solution to the SPCP problem in (2.8); and exploited this equivalence to develop first-order algorithms for (2.10): an alternating linearization method (ALM) for $\delta = 0$, and an accelerated proximal gradient (APG) method for $\delta > 0$. Later, Aybat and Iyengar developed a variable penalty ADMM algorithm to solve (2.8), which has desirable convergence guarantees unlike the multi-block ADMM in [59]. More recently, Aravkin et al. [1] proposed a "quasi-Newton" method to solve a more general formulation which includes the SPCP problem in (2.8) as a special case; and Mu et al. [45] proposed augmenting Frank-Wolfe method [21] with additional projected gradient steps to solve a flipped formulation related

to (2.8), i.e., $\min\{\|\pi_\Omega(L+S-A)\|_F : \|L\|_* \le \tau_L,\ \|S\|_1 \le \tau_S\}$ for some $\tau_L, \tau_S > 0$. The theoretical and practical convergence properties of these algorithms will be discussed in more details in the following section. Moreover, there are also other non-convex variants of the SPCP formulation, and iterative solution methods for them; we will very briefly go over these formulations and algorithms in Section 2.3.

Finally, it should also be emphasized that in order to achieve a fast real-time implementation by exploiting modern multi-core and/or distributed computing capabilities, there are some randomized techniques that divide the stable decomposition problem into smaller decomposition subproblems, which can be solved in parallel calling one of the decomposition algorithms discussed above as a base solver, and finally the subproblem solutions are combined at the end to generate a decomposition for the whole data. In particular, Mackey et al. [43] proposed a framework of this sort, Divide-Factor-Combine (DFC). DFC *randomly* divides the original matrix factorization task into smaller subproblems, and combines the smaller decompositions using an efficient *randomized matrix approximation* technique. The inherent parallelism of DFC allows for near-linear to superlinear speedups in practice, while the theory provides high-probability recovery guarantees for DFC comparable to those possessed by the base algorithm. Moreover, Mackey et al. [43] were able to reduce recovery error of their decomposition further by using ensemble methods that are known to improve the performance of matrix approximation algorithms [38].

2.1.4 Application: Foreground extraction from noisy videos

Extracting the *almost* still background from a sequence of frames in a noisy video is an important task in video surveillance, and interestingly it can be formulated as SPCP problem if the moving objects in the foreground only occupy a small fraction of each frame. This problem is difficult due to the presence of slightly changing background in the video, e.g., changing illumination conditions such as flickering lights, and/or slowly moving objects such as escalators or tree leaves. Let $F^{(t)}\mathbb{R}^{n_1 \times n_2}$ denote the t-th video frame, and $f^{(t)} \in \mathbb{R}^R$ is obtained by stacking the columns of $F^{(t)}$, where $R = n1n_2$ is the frame resolution. For now, suppose the background is completely stationary, and there is no measurement noise. Then $f^{(t)} = b + s^{(t)}$, where $b \in \mathbb{R}^R$ denotes the background and $s^{(t)} \in \mathbb{R}^R$ denotes the sparse foreground in the t-th frame. Hence, T-frame video can be encoded as matrix $\mathbb{R}^{R \times T} \ni A = [f^{(t)}, \ldots, f^{(T)}]$ such that $A = b\mathbf{1}^\top + [s^{(1)}, \ldots, s^{(T)}]$, i.e., summation of a rank 1 matrix + sparse matrix.

In real-life surveillance videos, the background is never completely stationary, and there is always measurement noise; therefore, we expect that A can be decomposed into the sum of three matrices $A = L^o + S^o + N^o$, where L^o is low rank and S^o is sparse matrices that represent the background and the foreground, respectively, and N^o is a dense noise matrix. When the background changes periodically, e.g., flickering light, or very slowly e.g., moving escalators, the matrix L^o, which represents the backgrounds in the frames, should be of low rank due to the high correlation between the video frames. The matrix S^o, which represents the moving foregrounds in the frames, should be sparse since the foreground usually occupies a small portion of each frame. Moreover, suppose that $\Omega \subset \{(i,t) : i = 1, \ldots, R,\ t = 1, \ldots, T\}$ denotes the subset of sensor-frame pairs (i,t) such that the signal recorded by i-th sensor in frame-t is completely corrupted due to malfunction. Hence, for foreground-background separation one can solve (2.8) with an appropriately chosen $\lambda > 0$.

2.2 Algorithms for SPCP

This section focusses on algorithms for solving the SPCP formulations in (2.3), (2.8), and (2.10). In the first subsection, we describe the ADMM method and its variant proposed by Tao and Yuan [59], and discuss their convergence properties. In the second subsection, after we go over basic properties of accelerated proximal gradient (APG) methods, we describe the APG implementations for the SPCP problem proposed in [3, 4]. Next, in the third subsection, we revisit the ADMM algorithms for solving the SPCP problem and describe the increasing penalty ADMM algorithm proposed in [6]. Finally, in the fourth and fifth subsections, we briefly review the convex variational framework proposed by Aravkin et al. [1], and the Frank-Wolfe projected gradient method proposed by Mu et al. [45] for solving a convex problem related to the SPCP formulation.

2.2.1 ADMM for SPCP

In this section, we discuss the first tailor-made algorithms for the SPCP problem: the alternating splitting augmented Lagrangian method (ASALM), and its variant (VASALM) proposed by Tao and Yuan in [59] for solving an equivalent formulation of (2.8):

$$\min_{L,S,N\in\mathbb{R}^{m\times n}} \{\|L\|_* + \lambda\|S\|_1 + \mathcal{I}_{\mathcal{X}}(N) :\ L+S+N = \pi_\Omega(A)\}, \tag{2.11}$$

where $\mathcal{X} = \{N \in \mathbb{R}^{m\times n} :\ \|\pi_\Omega(N)\|_F \leq \delta\}$, and $\mathcal{I}_{\mathcal{X}}$ denote the indicator function of the set $\mathcal{X}$. Given a constant penalty parameter $\rho > 0$, and a dual variable $Y \in \mathbb{R}^{m\times n}$, let $\mathcal{L}_\rho$ denote the augmented Lagrangian of (2.11):

$$\begin{aligned}\mathcal{L}_\rho(L,S,N;Y) \quad :=\quad & \|L\|_* + \lambda\|S\|_1 + \mathcal{I}_{\mathcal{X}}(N) \\ & - \langle Y, L+S+N-\pi_\Omega(A)\rangle + \frac{\rho}{2}\|L+S+N-\pi_\Omega(A)\|_F^2.\end{aligned} \tag{2.12}$$

Convergence to optimality would be an immediate result, if one were to compute the primal-dual iterate sequence using the *method of multipliers* [31, 50]: $(L_k, S_k, N_k) \in \operatorname{argmin} \mathcal{L}_{\rho_k}(L,S,N;Y_k)$ and $Y_{k+1} = Y_k - \rho_k(L_k+S_k+N_k-\pi_\Omega(A))$, where $\{\rho_k\}_{k\in\mathbb{Z}_+} \subset \mathbb{R}_{++}$. More precisely, Tao and Yuan [59] gave a convergence proof for *method of multipliers* customized to solve (2.11) when $\{\rho_k\}_{k\in\mathbb{Z}_+} \subset \mathbb{R}_{++}$ is chosen such that $\rho_{k+1} = \kappa\rho_k$ for some $\kappa > 1$, and showed that the primal-dual iterate sequence $\{(L_k, S_k, N_k, Y_k)\}_{k\in\mathbb{Z}_+}$ is bounded, and *any* of its *limit points* is an optimal solution to (2.11). However, minimizing $\mathcal{L}_\rho$ jointly in (L,S,N) is not *practical* at all, and it is almost as hard as solving (2.11). On the other hand, in order to exploit the separability of both the objective and constraint functions, Tao and Yuan considered minimizing the augmented Lagrangian alternatingly in one of the variables: L, S, or N, while fixing the other two variables. Hence, given a constant penalty parameter $\rho > 0$, they proposed constructing the ASALM iterate sequence as follows:

$$N_{k+1} = \operatorname*{argmin}_N \mathcal{L}_\rho(L_k, S_k, N; Y_k), \tag{2.13}$$

$$S_{k+1} = \operatorname*{argmin}_S \mathcal{L}_\rho(L_k, S, N_{k+1}; Y_k), \tag{2.14}$$

$$L_{k+1} = \operatorname*{argmin}_L \mathcal{L}_\rho(L, S_{k+1}, N_{k+1}; Y_k), \tag{2.15}$$

$$Y_{k+1} = Y_k - \rho(L_{k+1} + S_{k+1} + N_{k+1} - \pi_\Omega(A)). \tag{2.16}$$

Compared to minimizing $\mathcal{L}_\rho$ jointly in (L,S,N), the minimization subproblems in (2.13), (2.14), and (2.15) can be solved very efficiently in practice. Indeed, the N-subproblem

is nothing but a Euclidean projection, the S-subproblem is the shrinkage or soft-thresholding [18] operation, which is widely used in sparse signals reconstruction algorithms [15], and the L-subproblem is the matrix-shrinkage operation [32], which is used in algorithms for recovering low-rank matrices from few linear measurements [11, 42]; and the solutions to all these subproblems, stated in the following generic forms for some given $\bar{N}, \bar{S}, \bar{L} \in \mathbb{R}^{m\times n}$, can be computed explicitly in closed forms:

$$\begin{aligned} N_+ &= \operatorname*{argmin}_{N\in\mathcal{X}} \|N-\bar{N}\|_F = \mathbf{prox}_{\mathcal{I}_{\mathcal{X}}}(\bar{N}), \\ &= \min\left\{1,\ \frac{\delta}{\|\pi_\Omega\left(\bar{N}\right)\|_F}\right\}\pi_\Omega\left(\bar{N}\right) + \pi_{\Omega^c}\left(\bar{N}\right), \end{aligned} \tag{2.17}$$

$$\begin{aligned} S_+ &= \operatorname*{argmin}_{S} \lambda\|S\|_1 + \frac{\rho}{2}\|S-\bar{S}\|_F^2 = \mathbf{prox}_{\frac{\lambda}{\rho}\|\cdot\|_1}(\bar{S}), \\ &= \operatorname{sgn}(\bar{S}) \odot \max\left\{|\bar{S}| - \tfrac{\lambda}{\rho}\mathbf{1}_{m\times n},\ \mathbf{0}_{m\times n}\right\}, \end{aligned} \tag{2.18}$$

$$\begin{aligned} L_+ &= \operatorname*{argmin}_{L} \|L\|_* + \frac{\rho}{2}\|L-\bar{L}\|_F^2 = \mathbf{prox}_{\frac{1}{\rho}\|\cdot\|_*}(\bar{L}), \\ &= U\,\mathbf{diag}\left(\max\left\{\sigma - \tfrac{1}{\rho}\mathbf{1}_r,\ \mathbf{0}_r\right\}\right)V^\top, \end{aligned} \tag{2.19}$$

where $U\,\mathbf{diag}(\sigma)V^\top$ denote the SVD of $\bar{L}$, i.e., $\sigma \in \mathbb{R}^r_{++}$ denote the singular values of $\bar{L}$, $U \in \mathbb{R}^{m\times r}$, and $V \in \mathbb{R}^{n\times r}$ such that $U^\top U = V^\top V = \mathbf{I}_r$ and $r = \mathbf{rank}(\bar{L})$.

Compared to the method of multipliers, per-iteration complexity of ASALM is considerably cheaper; moreover, according to numerical results reported in [59], ASALM iterates converge to an optimal solution on synthetic random test problems. However, currently there is no theory that can support this empirical behavior. Note that ASALM is an ADMM algorithm with *three*-blocks, and in a recent paper Chen et al. [14] have shown that direct extension of ADMM algorithm from *two*-block to *multi*-block case is not necessarily convergent – we will discuss this issue and possible remedies in more detail in Section 2.2.3.

Tao and Yuan were able to fix the convergence issue using a slightly modified variant of ASALM (VASALM) without any increase in per-iteration complexity. Given algorithm parameters $\kappa > 2$ and $\rho > 0$, the steps of VASALM are as follows:

$$N_{k+1} = \operatorname*{argmin}_{N} \mathcal{L}_\rho(L_k, S_k, N; Y_k), \tag{2.20}$$

$$S_{k+1} = \operatorname*{argmin}_{S} \mathcal{L}_{\kappa\rho}(L_k, S, N_{k+1}; Y_k) + (\kappa-1)\langle \tilde{Y}_k - Y_k, S\rangle, \tag{2.21}$$

$$L_{k+1} = \operatorname*{argmin}_{L} \mathcal{L}_{\kappa\rho}(L, S_k, N_{k+1}; Y_k) + (\kappa-1)\langle \tilde{Y}_k - Y_k, L\rangle, \tag{2.22}$$

where $\tilde{Y}_k = Y_k - \rho(L_k + S_k + N_{k+1} - \pi_\Omega(A))$ for all $k \geq 1$, and $\{Y_k\}_{k\in\mathbb{Z}_+}$ is defined as in (2.16). Note that for $\kappa = 1$, VASALM and ASALM iterations are very similar, and the only difference is using S_k in (2.22) instead of S_{k+1}. Let $\mathcal{W}^*$ denote the set of primal-dual optimal solutions to the convex problem in (2.11), i.e., $\mathcal{W}^*$ is the convex set of all saddle points of the Lagrangian $\mathcal{L}_0$; and let $\mathcal{X}^* := \{(L^*, S^*, Y^*) :\ \exists N^* \text{ s.t. } (L^*, S^*, N^*, Y^*) \in \mathcal{W}^*\}$ and $X_k := (L_k, S_k, Y_k)$ for all $k \geq 1$. In [59], it is shown that $\{X_k\}_{k\in\mathbb{Z}_+}$ is Fejèr monotone [9] with respect to the closed convex set $\mathcal{X}^*$; hence, it follows that $\{X_k\}_{k\in\mathbb{Z}_+}$ converges to a point in $\mathcal{X}^*$. Hence, although convergence of ASALM iterate sequence is not theoretically guaranteed, VASALM iterate sequence is shown to be converging with a negligible additional work. The VASALM steps in (2.21) and (2.22) are only slightly different than those in ASALM; however, it is important to note that while ASALM uses S_{k+1} in (2.15) when

computing L_{k+1}, VASALM abandons S_{k+1}, and still uses S_k in (2.22). This modification in VASALM helps proving Fejèr monotonicity of the iterate sequence; but it comes at a cost of significant performance degradation in practice – according to numerical results reported in [59], ASALM performs much better than VASALM: the number of SVD computations for ASALM is almost half of what VASALM computes.

Before concluding this section, we will discuss another ADMM implementation for *stable* PCA. Recall that Zhou et al. [65] computed a solution to the Lagrangian formulation in (2.9) using an APG algorithm proposed in [40]. In [59], Tao and Yuan proposed the extensions of ASALM and VASALM to solve the Lagrangian formulation with missing data:

$$\min_{L,S\in\mathbb{R}^{m\times n}} \left\{ \|L\|_* + \lambda\|S\|_1 + \frac{\rho}{2}\|\pi_\Omega(N)\|_F^2 :\ L+S+N = \pi_\Omega(A) \right\}. \tag{2.23}$$

Note that if all the data is observable, then (2.23) and (2.9) are equivalent. Moreover, given $\delta > 0$, it follows from convex duality that there exists $\rho(\delta)$ such that (2.23) and (2.11) are equivalent. Extension of ASALM for solving (2.23) is also an ADMM algorithm with *three*-blocks, and like ASALM for (2.11), it also lacks proof of convergence; on the other hand, the same convergence guarantees of VASALM for (2.11) also hold for VASALM for (2.23). Again this convergence assurance comes at a cost of significant practical performance degradation compared to ASALM.

2.2.2 APG for SPCP

In Section 2.2.1, we have seen two ADMM-type methods: ASALM, and VASALM. While ASALM works well in practice, it does not have any convergence guarantees; on the other hand, while VASALM iterate sequence is shown to be Fejèr monotone with respect to the primal-dual optimal solution set –hence, converging to an optimal solution, its practical performance is significantly worse than ASALM. Moreover, iteration complexity of VASALM is *not* known. In this section, we discuss how a particular class of first-order methods with *optimal* iteration complexity can be efficiently implemented to compute an ϵ-optimal solution to (2.8) within $\mathcal{O}(1/\epsilon)$ iterations, where each iteration requires computing an SVD as a bottleneck operation, similar to ASALM and VASALM. First, as a preliminary, we briefly go over the basics of *accelerated proximal gradient* (APG) method, which is the backbone of the algorithms proposed in [3, 4]; and next, we discuss the convergence properties of these algorithms to solve (2.10).

Preliminaries: Convergence of APG & Nesterov smoothing

Let $\psi : \mathbb{R}^{m\times n} \to \mathbb{R}\cup\{+\infty\}$ and $\phi : \mathbb{R}^{m\times n} \to \mathbb{R}$ be proper, closed, convex functions such that ϕ is differentiable on an open set containing $\mathbf{dom}\,\psi$, and $\nabla\phi$ is Lipschitz continuous on $\mathbf{dom}\,\psi$ with constant C_ϕ, i.e., $\|\nabla\phi(X) - \nabla\phi(\tilde{X})\|_F \le C_\phi\|X-\tilde{X}\|_F$ for all $X, \tilde{X} \in \mathbf{dom}\,\psi$. Consider the following *composite* convex optimization problem:

$$P^* := \min_{X\in\mathbb{R}^{m\times n}} P(X) := \psi(X) + \phi(X). \tag{2.24}$$

Let $\mathcal{X} \subset \mathbb{R}^{m\times n}$ be a closed convex set; for the case $\psi = \mathcal{I}_\mathcal{X}$, i.e., the indicator function of $\mathcal{X}$, Nesterov [46, 49] proposed accelerated proximal gradient algorithms such that the iterate sequence $\{X_k\}_{k\in\mathbb{Z}_+}$ satisfies: $P(X_k) - P^* \le \mathcal{O}(C_\phi/k^2)$ for all $k \ge 1$; and each iteration requires computing $\nabla\phi$ and Euclidean projection(s) onto $\mathcal{X}$. Later, in [47] Nesterov extended the APG algorithm in [46] to solve (2.24) with the same iteration complexity of $\mathcal{O}(C_\phi/k^2)$ when ψ is a general closed convex function. Each iteration of the method in [47] requires *two* proximal map computations and uses all the iterates from previous iterations,

i.e., it is an ∞-memory algorithm; on the other hand, the method in [7] proposed by Beck and Teboulle, and the one in [60] proposed by Tseng extend the algorithm for *constrained smooth minimization* in [48] by Nesterov to solve *composite* convex minimization in (2.24) with $\mathcal{O}(C_\phi/k^2)$ iteration complexity. Moreover, this method requires *one* proximal map computation per iteration and only uses the past iterate from the previous iteration, i.e., 1-memory algorithm. The steps of APG algorithm are displayed in Figure 2.1.

Algorithm APG(X_0)

1: **input:** $X_0 \in \mathbb{R}^{m\times n}$
2: $k \leftarrow 0$, $t_0 \leftarrow 1$, $\bar{X}_0 \leftarrow X_0$
3: **while** $k \geq 0$ **do**
4: $\quad Q_k \leftarrow \bar{X}_k - \frac{1}{C_\phi}\nabla\phi(\bar{X}_k)$
5: $\quad X_{k+1} \leftarrow \underset{X}{\operatorname{argmin}}\, \psi(X) + \frac{C_\phi}{2}\|X - Q_k\|_F^2$
6: $\quad t_{k+1} \leftarrow \left(1 + \sqrt{1 + 4t_k^2}\right)/2$
7: $\quad \bar{X}_{k+1} \leftarrow X_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(X_{k+1} - X_k)$
8: $\quad k \leftarrow k + 1$
9: **end while**

FIGURE 2.1 APG: Accelerated Proximal Gradient algorithm

Let $\{(X_k, \bar{X}_k)\}_{k\in\mathbb{Z}_+}$ denote the APG iterate sequence computed as in Figure 2.1, and $\mathcal{X}^*$ denote the set of optimal solutions to (2.24), i.e., $\mathcal{X}^* = \{X \in \mathbb{R}^{m\times n} : \ P(X) = P^*\}$. In [7,60], it is shown that for any initial point $X_0 \in \mathbf{dom}\,\psi$, the iterate sequence $\{X_k\}_{k\in\mathbb{Z}_+}$ satisfies

$$0 \leq P(X_k) - P^* \leq \frac{2C_\phi\|X_0 - X^*\|_F^2}{k^2}, \quad \forall\ k \geq 1,\ \text{and}\ \ \forall X^* \in \mathcal{X}^*. \tag{2.25}$$

Hence, (2.25) implies that for *any* $\epsilon > 0$, APG in Figure 2.1 can compute an ϵ-optimal solution X_ϵ to (2.24), i.e., $P^* \leq P(X_\epsilon) \leq P^* + \epsilon$, within $\mathcal{O}\left(\sqrt{\frac{C_\phi}{\epsilon}}\right)$ APG-iterations.

Consider a special case of composite convex optimization problem in (2.24):

$$P^* := \min_{L,S\in\mathbb{R}^{m\times n}} P(L,S) := \psi(L,S) + \phi(L), \tag{2.26}$$

where $\psi : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R} \cup \{+\infty\}$, $\phi : \mathbb{R}^{m\times n} \to \mathbb{R}$ are closed convex functions such that $\nabla\phi$ is Lipschitz continuous on $\mathbb{R}^{m\times n}$ with constant C_ϕ, i.e., $\|\nabla\phi(L) - \nabla\phi(\tilde{L})\|_F \leq C_\phi\|L - \tilde{L}\|_F$ for all $L, \tilde{L} \in \mathbb{R}^{m\times n}$. Suppose that Step 5 and Step 7 of the APG algorithm in Figure 2.1 are replaced with (2.27) and (2.28), respectively.

$$(L_{k+1}, S_{k+1}) \leftarrow \underset{(L,S)}{\operatorname{argmin}}\, \psi(L,S) + \frac{C_\phi}{2}\|L - \bar{L}_k + \frac{1}{C_\phi}\nabla\phi(\bar{L}_k)\|_F^2, \tag{2.27}$$

$$\bar{L}_{k+1} \leftarrow L_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(L_{k+1} - L_k). \tag{2.28}$$

With a slight modification, the proof of Theorem 4.4 in [7] still holds for this case, and implies that the APG algorithm with Steps 5 and 7 replaced with (2.27) and (2.28), respectively, generates an iterate sequence $\{(L_k, S_k)\}_{k\in\mathbb{Z}_+}$ such that

$$P(L_k, S_k) - P^* \leq \frac{2C_\phi\|L_0 - L^*\|_F^2}{k^2}, \quad \forall\ k \geq 1,\ \forall L^* :\ \exists S^* \text{ s.t. } P(L^*, S^*) = P^*. \tag{2.29}$$

Now, consider the problem in (2.10), which is equivalent to the SPCP problem in (2.8). Since the objective function in (2.10) is the sum of two *non-smooth* functions,

$$f(L) := \|L\|_*, \quad \text{and} \quad g(S) := \lambda\|\pi_\Omega(S)\|_1,$$

one cannot call the APG algorithm to solve (2.10). On the other hand, using the smoothing technique introduced in [46], one can efficiently implement the APG algorithm on an approximate problem to (2.8). First, the smooth versions of the nuclear norm and ℓ_1-norm are briefly discussed next, and then two different APG implementations using the smoothed versions of these norms are described.

For fixed parameters $\mu > 0$ and $\nu > 0$, the smooth approximations, $f_\mu(.)$ and $g_\nu(.)$, to f and g are defined as follows:

$$f_\mu(L) := \max_{W\in\mathbb{R}^{m\times n}} \left\{\langle L, W\rangle - \frac{\mu}{2}\|W\|_F^2 : \ \|W\| \le 1\right\}, \tag{2.30}$$

$$g_\nu(S) := \max_{Z\in\mathbb{R}^{m\times n}} \left\{\langle \pi_\Omega(S), Z\rangle - \frac{\nu}{2}\|Z\|_F^2 : \ \|Z\|_\infty \le \lambda\right\}. \tag{2.31}$$

Let $W_\mu(L)$ and $Z_\nu(S)$ denote the optimal solutions to (2.30) and (2.31), respectively. For any given $L, S \in \mathbb{R}^{m\times n}$, it is shown in [4] that both solutions can be written in closed form:

$$W_\mu(L) = U \ \mathbf{diag}\left(\min\left\{\frac{\sigma}{\mu}, \mathbf{1}_r\right\}\right) V^\top, \tag{2.32}$$

$$Z_\nu(S) = \mathrm{sgn}(\pi_\Omega(S)) \odot \min\left\{\tfrac{1}{\nu}|\pi_\Omega(S)|, \ \lambda\mathbf{1}_{m\times n}\right\}, \tag{2.33}$$

where $r = \mathbf{rank}(L)$, and $L = U \ \mathbf{diag}(\sigma)V^\top$ is the singular value decomposition of L with $\sigma \in \mathbb{R}^r_{++}$ denoting the vector of singular values. According to Theorem 1 in [46], both f_μ and g_ν are differentiable, and their gradients can be explicitly written as

$$\nabla f_\mu(L) = W_\mu(L), \quad \nabla g_\nu(S) = \pi^*_\Omega(Z_\nu(S)) = Z_\nu(S). \tag{2.34}$$

Moreover, both ∇f_μ and ∇g_ν are Lipschitz continuous with Lipschitz constants $C_{f_\mu} = 1/\mu$ and $C_{g_\nu} = 1/\nu$, respectively.

Smooth and partially smooth proximal gradient algorithms for SPCP

Given an optimization problem $P^* = \min\{P(X) : \ X \in \mathcal{X}\}$ for some function P and set $\mathcal{X}$, in the rest of this section, a point $X_\epsilon \in \mathcal{X}$ is called an ϵ-optimal solution, if $P(X_\epsilon) \le P^* + \epsilon$. Consider the problems approximating (2.10) with a smooth objective function

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{f_\mu(L) + g_\mu(S) : (L, S) \in \mathcal{X}\}, \tag{2.35}$$

and with a partially smooth objective function

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{f_\mu(L) + g(S) : (L, S) \in \mathcal{X}\}, \tag{2.36}$$

where

$$\mathcal{X} := \{(L, S) : \ \|L + S - \pi_\Omega(A)\|_F \le \delta\}$$

denotes the feasible region of (2.10).

The inexact solutions of (2.35) and (2.36) are closely related to (2.10). Indeed, it is shown in [4] that for $\tau = \frac{1}{2}\min\{m, n\}$, and $\lambda = \frac{1}{\sqrt{\max\{m,n\}}}$, (2.30) and (2.31) imply that

$$f_\mu(L) \le f(L) \le f_\mu(L) + \mu\tau, \quad \forall L \in \mathbb{R}^{m\times n}, \tag{2.37}$$

$$g_\mu(S) \le g(S) \le g_\mu(S) + \mu\tau, \quad \forall S \in \mathbb{R}^{m\times n}. \tag{2.38}$$

Hence, as $\mu \to 0$, f_μ and g_μ uniformly converge to f and g, respectively, i.e., $f_\mu \to f$, and $g_\mu \to g$ uniformly as $\mu \to 0$. According to Theorem 2.1 in [4], for any $\epsilon > 0$, if $(L(\mu), S(\mu))$ is an $\frac{\epsilon}{2}$-optimal solution to the *smooth* problem in (2.35) for $\mu = \frac{\epsilon}{4\tau}$, then $(L(\mu), S(\mu))$ is an ϵ-optimal solution to (2.10). Similarly, with $\mu = \frac{\epsilon}{2\tau}$, an $\frac{\epsilon}{2}$-optimal solution to the *partially smooth* problem in (2.36) is an ϵ-optimal solution to (2.10). In the rest, we state and briefly discuss the Smooth Proximal Gradient (SPG) algorithm (a slightly modified version of Algorithm 1 in [3]), and the Partially Smooth Proximal Gradient (PSPG) algorithm [4] to compute an ϵ-optimal solution to (2.35) or (2.36), respectively, with provable iteration complexity bounds.

Algorithm SPG(L_0, S_0, ϵ)

input: $L_0, S_0 \in \mathbb{R}^{m\times n}$, $\epsilon > 0$
$\mu \leftarrow \epsilon/(2\min\{m,n\})$
$k \leftarrow 0$, $t_0 \leftarrow 1$, $\bar{L}_0 \leftarrow L_0$, $\bar{S}_0 \leftarrow S_0$
while $k \geq 0$ **do**
  $[U_k, V_k, \sigma_k] \leftarrow \mathrm{SVD}(\bar{L}_k)$ **Comment :** $\bar{L}_k = U_k\ \mathbf{diag}(\sigma_k)V_k^\top$, and $r_k = \mathbf{rank}(\bar{L}_k)$
  $Q_k^L \leftarrow U_k\,\mathbf{diag}\left(\max\{\sigma_k - \mu\mathbf{1}_{r_k},\ \mathbf{0}_{r_k}\}\right)V^\top$
  $Q_k^S \leftarrow \mathrm{sgn}(\pi_\Omega\left(\bar{S}_k\right)) \odot \max\left\{|\pi_\Omega\left(\bar{S}_k\right)| - \mu\lambda\mathbf{1}_{m\times n},\ \mathbf{0}_{m\times n}\right\}$
  $(L_{k+1}, S_{k+1}) \leftarrow \underset{(L,S)}{\mathrm{argmin}}\left\{\|L - Q_k^L\|_F^2 + \|S - Q_k^L\|_F^2 :\ (L,S) \in \mathcal{X}\right\}$
  $t_{k+1} \leftarrow \left(1 + \sqrt{1 + 4t_k^2}\right)/2$
  $\bar{L}_{k+1} \leftarrow L_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(L_{k+1} - L_k)$
  $\bar{S}_{k+1} \leftarrow S_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(S_{k+1} - S_k)$
  $k \leftarrow k + 1$
end while

FIGURE 2.2 SPG: Smooth Proximal Gradient algorithm

The SPG algorithm is an implementation of the APG algorithm, displayed in Figure 2.1, to solve (2.35). In particular, let ψ and ϕ in (2.24) be set as follows: $\psi(L,S) = \mathcal{I}_\mathcal{X}(L,S)$ and $\phi(L,S) = f_\mu(L) + g_\mu(S)$, where $\mu = \frac{\epsilon}{4\tau}$ and $\tau = \frac{1}{2}\min\{m,n\}$. It follows from (2.34) that $\nabla\phi$ is Lipschitz continuous with constant $1/\mu$; hence, $C_\phi = 1/\mu$. Therefore, Step 4 of Figure 2.1 corresponds to computing $[Q_k^L, Q_k^S] = [\bar{L}_k - \mu\nabla f_\mu(\bar{L}_k),\ \bar{S}_k - \mu\nabla g_\mu(\bar{S}_k)]$. However, first computing $\nabla f_\mu(\bar{L}_k)$ according to (2.32) and (2.34), and then computing Q_k^L can cause numerical problems. It is easy to check that

$$Q_k^L = U_k\,\mathbf{diag}\left(\max\{\sigma_k - \mu\mathbf{1}_{r_k},\ \mathbf{0}_{r_k}\}\right)V^\top, \tag{2.39}$$

where $\bar{L}_k = U_k\ \mathbf{diag}(\sigma_k)V_k^\top$ denotes the SVD of $\bar{L}_k$ and $r_k = \mathbf{rank}(\bar{L}_k)$. Hence, one can compute Q_k^L directly without computing $\nabla f_\mu(\bar{L}_k)$ to improve numerical stability. Similarly, (2.33) and (2.34) imply that Q_k^S can also be directly computed without computing $\nabla g_\mu(\bar{S}_k)$ as follows

$$Q_k^S = \mathrm{sgn}(\pi_\Omega\left(\bar{S}_k\right)) \odot \max\left\{|\pi_\Omega\left(\bar{S}_k\right)| - \mu\lambda\mathbf{1}_{m\times n},\ \mathbf{0}_{m\times n}\right\}.$$

Therefore, Step 4 of Figure 2.1 can be computed as in Step 6 and Step 7 of the SPG algorithm displayed in Figure 2.2.

Given $(L_0, S_0) \in \mathcal{X} = \{(L,S) :\ \|L + S - \pi_\Omega(A)\|_F \leq \delta\}$, e.g., $L_0 = \mathbf{0}$ and $S_0 = \pi_\Omega(A)$, the SPG algorithm keeps all iterates in $\mathcal{X}$; and according to (2.25), the convergence rate in function values of the SPG iterate sequence $\{(L_k, S_k)\}$ is $\mathcal{O}\left(\frac{\mu^{-1}}{k^2}\right)$. In particular, given $\epsilon > 0$, since $\mu = \frac{\epsilon}{2\min\{m,n\}}$, according to the discussion above, SPG can compute an ϵ-optimal solution to the SPCP problem in (2.10) by computing an $\frac{\epsilon}{2}$-optimal solution to (2.35)

within at most $N_{\mathrm{SPG}} = 2\frac{\sqrt{2\min\{m,n\}}}{\epsilon}\left(\|L_0 - L^*(\mu)\|_F^2 + \|S_0 - S^*(\mu)\|_F^2\right)^{\frac{1}{2}}$ iterations, where $(L^*(\mu), S^*(\mu))$ is an arbitrary optimal solution to (2.35). However, the overall computational complexity depends on per-iteration complexity of the SPG algorithm, which is partly determined by the complexity of solving subproblem in Step 8 in Figure 2.2. Now consider the subproblem in Step 8 of Figure 2.2 in the following generic form for some given $\bar{L}, \bar{S} \in \mathbb{R}^{m\times n}$:

$$(L_+, S_+) = \underset{L,S\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \left\{\|L - \bar{L}\|_F^2 + \|S - \bar{S}\|_F^2 :\ (L,S) \in \mathcal{X}\right\}. \tag{2.40}$$

Lemma 2.1 in [3] implies that (L_+, S_+) can be computed in closed form as follows:

$$\begin{aligned}
L_+ &= \frac{\theta_+}{2(1+\theta_+)}\left(\pi_\Omega(A) - \bar{S}\right) + \frac{2+\theta_+}{2(1+\theta_+)}\bar{L},\\
S_+ &= \frac{\theta_+}{2(1+\theta_+)}\left(\pi_\Omega(A) - \bar{L}\right) + \frac{2+\theta_+}{2(1+\theta_+)}\bar{S},\\
\theta_+ &= \max\left\{0,\ \frac{\|\bar{L} + \bar{S} - \pi_\Omega(A)\|_F}{\delta} - 1\right\}.
\end{aligned}$$

The PSPG algorithm proposed in [4] is also an implementation of the modified APG algorithm, with Steps 5 and 7 in Figure 2.1 replaced by (2.27) and (2.28), to solve (2.36). In particular, let ψ and ϕ in (2.26) be set as follows: $\psi(L,S) = g(S) + \mathcal{I}_{\mathcal{X}}(L,S)$ and $\phi(L,S) = f_\mu(L)$, where $\mu = \frac{\epsilon}{2\tau}$ and $\tau = \frac{1}{2}\min\{m,n\}$. It follows from (2.34) that $\nabla\phi$ is Lipschitz continuous with constant $1/\mu$; hence, $C_\phi = 1/\mu$. Each step of the PSPG algorithm, displayed in Figure 2.3, can be derived using (2.32) and $C_\phi = 1/\mu$. Hence, using exactly the same argument when we discussed SPG, one can directly compute Q_k^L according to (2.39) without computing $\nabla f_\mu(\bar{L}_k)$ to improve numerical stability. Since PSPG is a variant of APG algorithm, given $(L_0, S_0) \in \mathcal{X}$, e.g., $L_0 = \mathbf{0}$ and $S_0 = \pi_\Omega(A)$, like the SPG algorithm, the PSPG algorithm also keeps all iterates in $\mathcal{X}$; and according to (2.29), the convergence rate in function values of the PSPG iterate sequence $\{(L_k, S_k)\}$ is $\mathcal{O}\left(\frac{\mu^{-1}}{k^2}\right)$. In particular, given $\epsilon > 0$, since $\mu = \epsilon/\min\{m,n\}$, PSPG can compute an ϵ-optimal solution to the SPCP problem in (2.10) by computing an $\frac{\epsilon}{2}$-optimal solution to (2.36) within at most $N_{\mathrm{PSPG}} = 2\frac{\sqrt{\min\{m,n\}}}{\epsilon}\|L_0 - L^*(\mu)\|_F$ iterations, where $L^*(\mu)$ is the low-rank component of an arbitrary optimal solution $(L^*(\mu), S^*(\mu))$ to (2.36).

Algorithm PSPG(L_0, ϵ)

1: **input:** $L_0 \in \mathbb{R}^{m\times n}$, $\epsilon > 0$
2: $\mu \leftarrow \epsilon/\min\{m,n\}$
3: $k \leftarrow 0$, $t_0 \leftarrow 1$, $\bar{L}_0 \leftarrow L_0$
4: **while** $k \geq 0$ **do**
5: $\quad[U_k, V_k, \sigma_k] \leftarrow \mathrm{SVD}(\bar{L}_k)$ $\quad$ **Comment** : $\bar{L}_k = U_k\ \mathbf{diag}(\sigma_k)V_k^\top$, and $r_k = \mathbf{rank}(\bar{L}_k)$
6: $\quad Q_k^L \leftarrow U_k\,\mathbf{diag}\left(\max\{\sigma_k - \mu\mathbf{1}_{r_k},\ \mathbf{0}_{r_k}\}\right)V^\top$
7: $\quad(L_{k+1}, S_{k+1}) \leftarrow \underset{(L,S)}{\operatorname{argmin}}\left\{\lambda\|\pi_\Omega(S)\|_1 + \frac{1}{2\mu}\|L - Q_k^L\|_F^2 :\ (L,S)\in\mathcal{X}\right\}$
8: $\quad t_{k+1} \leftarrow \left(1 + \sqrt{1 + 4t_k^2}\right)/2$
9: $\quad\bar{L}_{k+1} \leftarrow L_{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(L_{k+1} - L_k)$
10: $\quad k \leftarrow k + 1$
11: **end while**

FIGURE 2.3 PSPG: Partially Smooth Proximal Gradient algorithm

The overall computational complexity depends on per-iteration complexity, which is partly determined by the complexity of solving subproblem in Step 7 of the PSPG algorithm in Figure 2.3. Since the form of solution and the complexity of this step was not analyzed before, FISTA [7] has not been previously applied to solve (2.36) and its overall complexity for solving (2.36) has been unknown, until Aybat, Goldfarb, and Ma [4] showed that the subproblem can be efficiently solved and per-iteration complexity of PSPG is determined by the complexity of an SVD. Now consider the subproblem in Step 7 of Figure 2.3 in the following generic form for some $\rho > 0$ and $\bar{L} \in \mathbb{R}^{m\times n}$:

$$(L_+, S_+) = \underset{L,S\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \left\{ \lambda \|\pi_\Omega(S)\|_1 + \frac{\rho}{2}\|L-\bar{L}\|_F^2 : \ (L,S) \in \mathcal{X} \right\}. \tag{2.41}$$

According to Lemma 6.1 in [4], (L_+, S_+) can be computed in closed form as follows:

$$S_+ = \operatorname{sgn}(\pi_\Omega(A-\bar{L})) \odot \max\left\{ |\pi_\Omega(A-\bar{L})| - \lambda \left(\frac{1}{\theta_+}+\frac{1}{\rho}\right) \mathbf{1}_{m\times n}, \ \mathbf{0}_{m\times n} \right\} - \pi_{\Omega^c}(\bar{L}),$$

$$L_+ = \frac{\theta_+}{\rho+\theta_+}\pi_\Omega(A-S_+) + \frac{\rho}{\rho+\theta_+}\pi_\Omega(\bar{L}) + \pi_{\Omega^c}(\bar{L}),$$

where $\theta_+ = 0$ if $\|\pi_\Omega(A-\bar{L})\|_F \le \delta$; otherwise, θ_+ is the *unique positive* solution to the nonlinear equation $\Gamma(\theta) = 0$, where Γ is defined as follows

$$\Gamma(\theta) := \| \min\left\{ \frac{\lambda}{\theta}, \frac{\rho}{\rho+\theta}|\pi_\Omega(A-\bar{L})| \right\} \|_F,$$

i.e., $\{\theta > 0 : \ \Gamma(\theta) = 0\}$ is a singleton, and equal to $\{\theta_+\}$. If $\theta_+ = 0$, then $\frac{1}{\theta_+}$ is interpreted as $+\infty$; hence, $S_+ = -\pi_{\Omega^c}(\bar{L})$ and $L_+ = \bar{L}$. Moreover, the proof of Lemma 6.1 in [4] shows that θ_+ can be *efficiently* computed in $\mathcal{O}(|\Omega|\log(|\Omega|))$ time.

In [4], PSPG and ASALM are tested on both randomly generated synthetic problems and surveillance video foreground extraction problems. According to numerical results reported in [4], on the synthetic problems the number of partial SVD computations for PSPG and ASALM are on par; on the other hand, surprisingly the average runtime for ASALM is approximately *twice* the runtime for PSPG. This significant gap in the average solution times is explained by the fact that ASALM required more leading singular value computations than PSPG did per partial SVD.

It should be emphasized that in practice adopting a continuation technique for the smoothing parameter speeds up both SPG and PSPG. In particular, for a given approximation error level $\epsilon > 0$, the smoothing parameter μ is fixed in Figure 2.2 and Figure 2.3 accordingly, i.e., $\mu = \frac{\epsilon}{2\min\{m,n\}}$ for SPG, and $\mu = \frac{\epsilon}{\min\{m,n\}}$ for PSPG, so that APG algorithms can compute an ϵ-optimal solution to (2.8) within $\mathcal{O}(1/\epsilon)$ iterations. However, instead of fixing μ throughout the algorithm, choosing $\{\mu_k\}$ such that $\mu_{k+1} < \mu_k$ and $\mu_k \to \mu$, and setting $\mu = \mu_k$ in the k-th iteration of SPG and PSPG works much better in practice. For instance, given $\kappa \in (0,1)$ and $\mu_1 > \mu$, one continuation strategy, similar to the one adopted in [4], sets $\mu_{k+1} = \kappa\mu_k$ if $k \le \bar{K}$, and $\mu_k = \mu$ for $k > \bar{K}$, where $\bar{K} = \left\lfloor \log_{\frac{1}{\kappa}}\left(\frac{\mu_1}{\mu}\right) \right\rfloor$.

2.2.3 ADMM revisited

Recall that ASALM, discussed in Section 2.2.1, is a three-block ADMM algorithm proposed by Tao and Yuan [59] to solve (2.8), and although it does not have any convergence guarantees, it works well in practice; and slightly changing the update rule in ASALM leads to VASALM, of which iterate sequence converges to an optimal solution; but this comes at

the cost of degradation in practical convergence speed when compared to ASALM. Next, in Section 2.2.2, we have reviewed two APG algorithms, SPG and PSPG, proposed in [3, 4] that have stronger convergence guarantees: an ϵ-optimal solution to the SPCP problem in (2.8) can be computed within $\mathcal{O}(1/\epsilon)$ iterations, and computational complexities per iteration of SPG and PSPG are comparable to the work per iteration required by ASALM and VASALM, which is mainly determined by an SVD computation. On the other hand, it is also important to emphasize that SPG and PSPG iterate sequences do not converge to an optimal solution to the SPCP problem in (2.8). In particular, since within SPG and PSPG the smoothing parameter μ is fixed, depending on the approximation parameter ϵ for solving (2.35) and (2.36), respectively, further iterations after reaching an ϵ-optimal solution in $\mathcal{O}(1/\epsilon)$ iterations do not necessarily improve the solution quality.

In this section, we revisit ADMM algorithms for solving the SPCP problem, and discuss the *variable penalty* ADMM algorithm ADMIP [1] (Alternating Direction Method with Increasing Penalty) proposed in [6] as it possesses all the advantages of the methods discussed in the previous two sections:

a) its iterate sequence converges to an optimal solution of (2.8), like VASALM iterate sequence, while it is much faster than ASALM in practice,

b) when constant penalty parameter is used as a special case, it can compute an ϵ-optimal solution within $\mathcal{O}(1/\epsilon)$ iterations, of which complexity is determined by an SVD.

Let

$$\mathcal{X} := \{(L, S) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} : \ \|\pi_\Omega (L + S - A)\|_F \leq \delta\} \tag{2.42}$$

denote the feasible set of the SPCP problem in (2.8). Using *partial* variable splitting, i.e., only split the L variables in (2.8), to arrive at the following equivalent problem

$$\min_{X,L,S\in\mathbb{R}^{m\times n}} \{\|X\|_* + \lambda \|S\|_1 + \mathbf{1}_{\mathcal{X}}(L, S) : \ X - L = \mathbf{0}_{m\times n}\}. \tag{2.43}$$

Let $\psi_1(L) := \|L\|_*$ and $\psi_2(L, S) := \lambda \|S\|_1 + \mathbf{1}_{\mathcal{X}}(L, S)$. In this section, we briefly review ADMIP proposed in [6] to solve the following ADMM formulation of the SPCP problem:

$$\min_{X,L,S\in\mathbb{R}^{m\times n}} \{\psi_1(X) + \psi_2(L, S) : \ X - L = \mathbf{0}_{m\times n}\}. \tag{2.44}$$

Variable Penalty Alternating Direction Method of Multipliers

First as a preliminary to this section, we briefly review variable penalty ADMM and its connections to other convex optimization methods in general. Let $\psi_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$ be a proper, closed, convex function, and $A_i : \mathbb{R}^{m\times n_i}$ for all $i = 1, \ldots, M$ and $b \in \mathbb{R}^m$. Let $n := \sum_{i=1}^M n_i$ and $\mathbb{R}^n \ni x = [x^{(1)}; \ldots; x^{(M)}]$. Consider the following convex optimization problem:

$$P^* = \min_{x\in\mathbb{R}^n} P(x) := \sum_{i=1}^M \psi_i(x^{(i)}) \quad \text{s.t.} \quad \sum_{i=1}^M A_i x^{(i)} = b. \tag{2.45}$$

Given $\rho \geq 0$ and a dual variable $y \in \mathbb{R}^m$, the augmented Lagrangian of (2.45) is given by

$$\mathcal{L}_\rho(x^{(1)}, \ldots, x^{(M)}; y) := \sum_{i=1}^M \psi_i(x^{(i)}) - \langle y, \sum_{i=1}^M A_i x^{(i)} - b\rangle + \frac{\rho}{2}\|\sum_{i=1}^M A_i x^{(i)} - b\|_2^2,$$

[1] In an earlier preprint [3], it was named as `NSA` algorithm.

and $g_\rho(y) := \inf_x \mathcal{L}_\rho(x,y)$ denotes the dual function – note that g_0 is the Lagrangian dual function of (2.45).

The *dual ascent* method [57] sets $\rho = 0$, and starting from arbitrary $y_0 \in \mathbb{R}^m$ computes the iterate $x_k = [x_k^{(1)}; \ldots; x_k^{(M)}]$ as follows: $x_k \in \operatorname{argmin} \mathcal{L}_0(x; y_k)$, $y_{k+1} = y_k - \alpha_k \left(\sum_{i=1}^M A_i x_k^{(i)} - b\right)$, where $\alpha_k > 0$ is an appropriately chosen step size. Since $\rho = 0$, $\{x_k^{(i)}\}_{i=1}^M$ can be computed in parallel. This is simply a sub-gradient method in dual space, and it is guaranteed to converge to an optimal solution under certain assumptions [57]; however, it suffers from slow convergence rate $\mathcal{O}(1/\sqrt{k})$. Unless $\sum_{i=1}^M A_i x_k^{(i)} - b$ is the same for all $x_k \in \operatorname{argmin} \mathcal{L}_0(x; y_k)$ – this is trivially true if $\min \mathcal{L}_0(x; y_k)$ has a unique solution, g_0 is not differentiable at y_k; on the other hand, for any $\rho > 0$, it can be shown that g_ρ is the Moreau-Yosida regularization of g_0, i.e., $g_\rho(y) = \max_w g_0(w) - \frac{1}{2\rho}\|w - y\|_2^2$; hence, g_ρ is differentiable with a Lipschitz continuous gradient, of which Lipschitz constant is $\frac{1}{\rho}$, and $y^* \in \operatorname{argmax}_y g_0(y)$ if and only if $y^* \in \operatorname{argmax}_y g_\rho(y)$ – see [52,54] for details. Using this relation Rockafellar [53,55] established the connection between the *method of multipliers* [31,50] and the proximal-point algorithm. Therefore, for $\{\rho_k\} \subset \mathbb{R}_{++}$, the *method of multipliers* iterate sequence, generated as $x_k \in \operatorname{argmin}_x \mathcal{L}_{\rho_k}(x; y_k)$ and $y_{k+1} = y_k - \rho_k \left(\sum_{i=1}^M A_i x_k^{(i)} - b\right)$, is shown in [24] to converge with the following rate

$$P^* - g_0(y_k) \leq \frac{\|y_0 - y^*\|_2^2}{2\sum_{i=1}^k \rho_i}, \tag{2.46}$$

where $y^* \in \operatorname{argmax} g_0(y)$. Hence, when $\rho_k = \rho$ for some $\rho > 0$, or $\sup_k \rho_k < \infty$, the method of multipliers iterate sequence converges with $\mathcal{O}(1/k)$ rate, and when $\sup_k \rho_k = +\infty$, it converges with $o(1/k)$ rate. Similarly, it is shown in [53,55] that under assumptions related to *strong* second-order optimality conditions, the primal and dual iterates converge to an optimal pair *superlinearly* when the penalty parameters $\rho_k \nearrow \infty$, while the rate is only *linear* when $\sup_k \rho_k < \infty$. However, this result has not been extended to *variable penalty* ADMM, which will be described next. It should also be emphasized that since $\rho_k > 0$, the *separability* is lost in the augmented Lagrangian function, and x_k computation in *method of multipliers* cannot be parallelized as in *dual ascent*.

Given $y_1 \in \mathbb{R}^m$ and $\{\rho_k\}_{k \in \mathbb{Z}_+} \subset \mathbb{R}_{++}$, according to Gauss-Seidel update rule ADMM iterates $x_k = [x_k^{(1)}; \ldots; x_k^{(M)}]$ are computed as follows:

$$x_{k+1}^{(i)} \in \underset{x^{(i)} \in \mathbb{R}^{n_i}}{\operatorname{argmin}} \mathcal{L}_{\rho_k}(x_{k+1}^{(1)}, \ldots, x_{k+1}^{(i-1)}, x^{(i)}, x_k^{(i+1)}, \ldots, x_k^{(M)}; y_k), \quad i = 1, \ldots, M, \tag{2.47}$$

$$y_{k+1} = y_k - \rho_k \left(\sum_{i=1}^M A_i x_{k+1}^{(i)} - b\right). \tag{2.48}$$

First, we consider the *two-block* case, i.e., $M = 2$. When the penalty parameter sequence is constant, i.e., $\rho_k = \rho$ for some $\rho > 0$, the iterate sequence $\{x_k\}$ converges to an optimal solution to (2.45) – see the recent surveys [10, 19] and [20] for more details; moreover, for this setting $\mathcal{O}(1/k)$ rate of convergence has also been shown in [29]. Although convergence is guaranteed for all $\rho > 0$, the empirical performance is critically dependent on the choice of *constant* penalty parameter ρ – it deteriorates very rapidly if the penalty is set too large or too small [22, 23, 37]. Moreover, it is discussed in [41] that under certain assumptions on the objective function, there exists ρ^* which optimizes the convergence bounds for the constant penalty ADMM scheme; however, estimating ρ^* is difficult in practice [30]. Therefore, adopting a variable penalty sequence within ADMM can be a fruitful approach to fix the issues caused by picking ρ either too large or too small; however, it is difficult to prove the

convergence when penalty parameters change in every iteration, which is also remarked by Boyd et al. [10] (see Section 3.4.1), and the existing results on the convergence properties of variable penalty ADMM is very *limited*. The convergence results of He et al. [26, 27, 30] on variable penalty ADMM algorithms implicitly assume that both ψ_1 and ψ_2 in the objective function are *differentiable*; – see Assumption A and the following discussion on page 107 in [27]; therefore, these results do not extend to *non-smooth* optimization problem in (2.44), i.e., to the *two-block* ADMM formulation of (2.8). On the other hand, the ADMM algorithm in [37] can solve (2.45) with $M = 2$ when both ψ_1 and ψ_2 are *non-smooth* convex functions; however, the convergence proof requires that the penalty sequence $\{\rho_k\}$ increases only *finitely* many times; i.e., $\{\rho_k\}$ is *bounded above* – [27, 30] also assume bounded $\{\rho_k\}$.

Next, consider the *multi-block* case, i.e., $M \geq 3$. Through variable splitting (2.45) can be equivalently written using only *two-blocks* – see [8, 10] for details; however, the number of variables and constraints in the equivalent problem increase significantly. As a remedy, one can consider the multi-block ADMM as described in (2.47) and (2.48); however, it has been recently shown that the direct extension of two-block ADMM to three-blocks is not necessarily convergent [14] even for constant penalty, i.e., $\rho_k = \rho$ for some $\rho > 0$. Convergence for multi-block ADMM with Gauss-Seidel updates as in (2.47) and (2.48) with $\rho_k = \rho$ has only been established for very special cases of (2.45), e.g., [34,61]; in certain cases it works well even though lacking theoretical convergence guarantees, e.g., ASALM [59] for solving (2.11). Note that (2.47) cannot be parallelized; however, replacing (2.47) with

$$x_{k+1}^{(i)} \in \underset{x^{(i)} \in \mathbb{R}^{n_i}}{\operatorname{argmin}} \mathcal{L}_\rho(x_k^{(1)}, \ldots, x_k^{(i-1)}, x^{(i)}, x_k^{(i+1)}, \ldots, x_k^{(M)}; y_k), \quad i = 1, \ldots, M, \tag{2.49}$$

one obtains the multi-block ADMM with Jacobian updates. Although this modification is fully parallelizable, there are examples for which the iterate sequence of ADMM with Jacobian updates diverges even for the *two-block* case, e.g., [28]. Very recently, Deng et al. [17] have shown that appending a proximal term $\tau_i \|x^{(i)} - x_k^{(i)}\|_2^2$ to (2.49) generates a convergent sequence for particular choice of $\{\tau_i\}_{i=1}^M \subset \mathbb{R}_{++}$. Moreover, under certain conditions on $\{A_i\}_{i=1}^M$, $o(1/k)$ convergence rate has been established for the proximal Jacobian ADMM in [17]. On the other hand, for variable penalty multi-block ADMM algorithm even the convergence of the iterate sequence is not known.

ADMIP: alternating direction method with increasing penalty

The ADMIP [2] algorithm [6], displayed in Figure 2.4, is an implementation of the ADMM algorithm with Gauss-Seidel updates, i.e., (2.47) and (2.48), to solve (2.43), which is equivalent to (2.8). In particular, for given penalty parameter $\rho > 0$, and dual variable $Y \in \mathbb{R}^{m \times n}$, the augmented Lagrangian function corresponding to (2.44) has the following form:

$$\mathcal{L}_\rho(X, L, S; Y) = \psi_1(X) + \psi_2(L, S) - \langle Y, X - L \rangle + \frac{\rho}{2} \|X - L\|_F^2, \tag{2.50}$$

where $\psi_1(L) := \|L\|_*$, $\psi_2(L, S) := \lambda \|S\|_1 + \mathbf{1}_{\mathcal{X}}(L, S)$, and $\mathcal{X}$ is defined in (2.42). Hence, the two subproblems in (2.47) can be written equivalently as:

$$X_{k+1} = \underset{X \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \|X\|_* + \frac{\rho_k}{2} \|X - (L_k + Y_k/\rho_k)\|_F^2, \tag{2.51}$$

$$(L_{k+1}, S_{k+1}) = \underset{L, S \in \mathbb{R}^{m \times n}}{\operatorname{argmin}} \{\lambda \|S\|_1 + \frac{\rho_k}{2} \|L - (X_{k+1} - Y_k/\rho_k)\|_F^2 : \ (L, S) \in \mathcal{X}\}. \tag{2.52}$$

[2] In an earlier preprint [3], it was named as `NSA` algorithm.

Note that X-subproblem in (2.51) is the matrix-shrinkage operation [32], and can be computed as in (2.19). Now consider (2.52) in the following generic form for some $\rho > 0$ and $\bar{L} \in \mathbb{R}^{m\times n}$:

$$\min_{L,S\in\mathbb{R}^{m\times n}} \{\lambda\|S\|_1 + \frac{\rho}{2}\|L-\bar{L}\|_F^2 :\ (L,S)\in\mathcal{X}\}. \tag{2.53}$$

It can be easily shown that (2.53) is equivalent to the PSPG subproblem in (2.41); in the sense that if (L_+, S_+) denotes the optimal solution to (2.41), then $(L_+, \pi_\Omega(S_+))$ is the optimal solution to (2.53). Hence, Step 8 in Figure 2.4 can be computed efficiently in at most $\mathcal{O}(|\Omega|\log(|\Omega|))$ time.

Algorithm ADMIP$(L_0, Y_0, \{\rho_k\})$

1: **input:** $L_0, Y_0 \in \mathbb{R}^{m\times n}$, $\{\rho_k\} \subset \mathbb{R}_{++}$
2: $k \leftarrow 0$
3: **while** $k \geq 0$ **do**
4: $\quad Q_k^X \leftarrow L_k + \frac{1}{\rho_k} Y_k$
5: $\quad [U_k, V_k, \sigma_k] \leftarrow \mathrm{SVD}(Q_k^X)$ $\quad$ **Comment** : $Q_k^X = U_k\ \mathbf{diag}(\sigma_k) V_k^\top$, and $r_k = \mathbf{rank}(Q_k^X)$
6: $\quad X_{k+1} \leftarrow U_k\, \mathbf{diag}\left(\max\{\sigma_k - \frac{1}{\rho_k}\mathbf{1}_{r_k},\ \mathbf{0}_{r_k}\}\right) V^\top$
7: $\quad Q_k^L \leftarrow X_{k+1} - \frac{1}{\rho_k} Y_k$
8: $\quad (L_{k+1}, S_{k+1}) \leftarrow \underset{(L,S)}{\mathrm{argmin}} \left\{\lambda\|S\|_1 + \frac{\rho_k}{2}\|L - Q_k^L\|_F^2 :\ (L,S)\in\chi\right\}$
9: $\quad Y_{k+1} \leftarrow Y_k - \rho_k(X_{k+1} - L_{k+1})$
10: $\quad k \leftarrow k+1$
11: **end while**

FIGURE 2.4 ADMIP: Alternating Direction Method with Increasing Penalty

Since ADMIP is a two-block ADMM algorithm, if a *constant penalty* scheme is adopted, i.e., $\rho_k = \rho$ for all k for some $\rho > 0$, then according to [29], (X_k, L_k, S_k, Y_k) converges to an optimal solution with $\mathcal{O}(1/k)$ rate of convergence. Moreover, for the case penalty sequence is not constant, Aybat and Iyengar [6] show that both primal and dual ADMIP iterates converge to an optimal primal-dual solution to the SPCP problem in (2.8) under mild conditions on the penalty parameter sequence. In particular, if $\{\rho_k\}_{k\in\mathbb{Z}_+}$ is a non-decreasing sequence such that $\sum_k \rho_k^{-1} = +\infty$, then $L^* := \lim_k X_k = \lim_k L_k$ and $S^* := \lim_k S_k$ exist; and (L^*, S^*) is an optimal solution to (2.8); and if $\{\rho_k\}_{k\in\mathbb{Z}_+}$ also satisfies $\sum_{k\in\mathbb{Z}_+} \rho_k^{-2} = +\infty$, then $Y^* := \lim_{k\in\mathbb{Z}_+} Y_k$ exists whenever $\|\pi_\Omega(A - L^*)\|_F \neq \delta$, and (L^*, L^*, S^*, Y^*) is a saddle point of the Lagrangian function $\mathcal{L}_0$ in (2.50). Although convergence rate is not known for ADMIP, it should be emphasized that even the convergence result in [6] is the first one for a variable penalty ADMM when penalties are *not bounded*, the objective function is *non-smooth* and its subdifferential is *not* uniformly bounded.

The main advantages of adopting an increasing sequence of penalties are as follows:

1. The algorithm is robust in the sense that there is no need to search for ρ^*, depending on problem parameters, that works well in practice, or that optimizes some convergence bounds as in [41].
2. The algorithm is likely to achieve primal feasibility faster.
3. The complexity of initial (transient) iterations can be controlled through controlling $\{\rho_k\}$. The main computational bottleneck in ADMIP (see Figure 2.4) is the SVD computation in Step 5. Since the optimal L^* is of low-rank, and $L_k \to L^*$, eventually the SVD computations are likely to be very efficient. However, since the initial iterates in the transient phase of the algorithm may have large rank,

the complexity of the SVD in the initial iterations can be quite large. From Step 7 in Figure 2.4, it follows that one does not need to compute singular values smaller than $1/\rho_k$; hence, starting ADMIP with a *small* $\rho_0 > 0$ will significantly decrease the complexity of initial iterations via adopting *partial* SVD computations.

Furthermore, the numerical experiment results reported in [6] clearly demonstrate that adopting an increasing penalty parameter sequence leads to faster convergence in practice; in fact, the performance of ADMIP dominates the performance of constant penalty ADMM for any fixed penalty term. The numerical experiments also confirm that ADMIP is significantly more robust to changes in problem parameters. In [6], Aybat and Iyengar also compared ADMIP against ASALM on both randomly generated synthetic problems and surveillance video foreground extraction problems. According to numerical results reported in [6], on the synthetic problems ASALM requires about *twice* as many iterations for convergence, while the total runtime for ASALM is considerably larger; on surveillance video foreground extraction problem, both ADMIP and ASALM were able to recover the foreground and the background fairly accurately with only 60% of the pixels functioning, i.e., $\frac{|\Omega|}{mn} = 0.6$ in (2.8); both iteration count and runtime of ADMIP are smaller than those of ASALM.

Before concluding this section, we would like to give an example of alternative *two-block* ADMM formulation for SPCP. Recently, Huai et al [35] proposed a *two-block* ADMM algorithm to solve the SPCP problem with non-negativity constraints:

$$\min\{\|L\|_* + \lambda\|S\|_1 : \ \|L + S - A\|_F \leq \delta, \ L \geq \mathbf{0}_{m\times n}\}. \tag{2.54}$$

Let $\chi_1 := \{N \in \mathbb{R}^{m\times n} : \ \|N\|_F \leq \delta\}$, and $\chi_2 := \{X \in \mathbb{R}^{m\times n} : \ X \geq \mathbf{0}_{m\times n}\}$. Then Huai et al. [35] equivalently formulated as follows:

$$\begin{aligned} &\min_{L,S,N,X\in\mathbb{R}^{m\times n}} \|L\|_* + \lambda\|S\|_1 + \mathcal{I}_{\chi_1}(N) + \mathcal{I}_{\chi_2}(X) \\ &\quad \text{s.t.} \quad \begin{pmatrix} \mathbf{I}_m & \mathbf{I}_m \\ -\mathbf{I}_m & \mathbf{0}_{m\times m} \end{pmatrix} \begin{pmatrix} L \\ S \end{pmatrix} + \begin{pmatrix} \mathbf{I}_m & \mathbf{0}_{m\times m} \\ \mathbf{0}_{m\times m} & \mathbf{I}_m \end{pmatrix} \begin{pmatrix} N \\ X \end{pmatrix} = \begin{pmatrix} A \\ \mathbf{0}_{m\times n} \end{pmatrix}. \end{aligned}$$

Hence, one can solve this formulation using *two-block* ADMM alternatingly minimizing in (L, S) and (N, X) blocks. It is important to note that for large-scale SPCP problems, the memory requirement can be a critical issue for this formulation.

2.2.4 Quasi-Newton method for SPCP

Let $A = L^o + S^o + N^o$ such that $\|N^o\|_F \leq \delta$, for stable decomposition of A, Aravkin et al. [1] proposed a convex variational framework, which is accelerated with a "quasi-Newton" method. Proposed model is given in (2.55), and it includes the SPCP problem in (2.8) as a special case:

$$\min \psi(L, S) \quad \text{s.t.} \quad \phi(\mathcal{A}(L, S) - A) \leq \delta, \tag{2.55}$$

where $\mathcal{A} : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a linear operator, $\phi : \mathbb{R}^{m\times n} \to \mathbb{R}$ is a smooth convex function such as $\|.\|_F^2$ or the robust Huber penalty, and ψ can be set to either one of the following functions

$$\begin{aligned} \psi_{\text{sum}}(L, S) &:= \|L\|_* + \lambda\|S\|_1, \\ \psi_{\max}(L, S) &:= \max\{\|L\|_*, \lambda_{\max}\|S\|_1\}, \end{aligned}$$

$\lambda, \lambda_{\max} > 0$ are some given function parameters. Note that setting $\psi = \psi_{\text{sum}}$, $\rho(.) = \|.\|_F^2$, and $\mathcal{A}(L, S) = \pi_\Omega(L + S)$ in (2.55), one obtains (2.8).

This approach offers advantages over the original SPCP formulation in terms of practical parameter selection. Indeed, the authors argue that even though setting $\lambda = \frac{1}{\sqrt{\max\{m,n\}}}$

in (2.3) has theoretical justification, which is discussed in Section 2.1.2, many practical problems may violate the underlying assumptions in (2.5) and (2.6); in those cases one needs to tune λ via cross validation, and selecting $\lambda_{\max}$ in $\psi_{\max}$ might be easier than selecting λ in ψ_{sum}.

Instead of directly solving (2.55), Aravkin et al. [1] used Newton's method to find a root of the value function:

$$\upsilon(\tau) := \min_{L,S\in\mathbb{R}^{m\times n}} \phi(\mathcal{A}(L,S)-A) - \delta \quad \text{s.t.} \quad \psi(L,S) \le \tau, \tag{2.56}$$

i.e., given $\delta > 0$ compute $\tau^* = \tau(\delta)$ such that $\upsilon(\tau^*) = 0$. According to results in [2], if the constraint in (2.55) is tight at an optimal solution, then there exists $\tau^* = \tau(\delta)$ such that $\upsilon(\tau^*) = 0$ and the corresponding optimal solution to (2.56) is also optimal to (2.55).

For the sake of simplicity, suppose that $\phi(.) = \|.\|_F^2$, which ensures that υ is differentiable, and let (L_k, S_k) denote the optimal solution to (2.56) when $\tau = \tau_k$. The next point τ_{k+1} is generated using Newton's method: $\tau_{k+1} = \tau_k - \frac{\upsilon(\tau_k)}{\upsilon'(\tau_k)}$. Aravkin et al. show that υ' can be computed efficiently: $\upsilon'(\tau_k) = -\psi^\circ(\mathcal{A}^\top \nabla\phi(\mathcal{A}(L_k,S_k)-A))$, and ψ° denotes the polar gauge to ψ. Indeed, according to Corollary 4.2 in [2], ψ°_{sum} and $\psi^\circ_{\max}$ can be written in the following closed forms: $\psi^\circ_{\text{sum}}(L,S) = \max\{\|L\|, \ \frac{1}{\lambda}\|S\|_\infty\}$, and $\psi^\circ_{\max}(L,S) = \|L\| + \frac{1}{\lambda_{\max}}\|S\|_\infty$. Therefore, given $\tau = \tau_k$ if the optimal solution (L_k, S_k) to (2.56) can be computed efficiently, one can minimize (2.55) efficiently using Newton's method as discussed above. Since Euclidean projections onto $\{(L,S) : \psi(L,S) \le \tau\}$ can be computed efficiently for both ψ_{sum} and $\psi_{\max}$, in order to solve (2.56) Aravkin et al. adopted projected "quasi-Newton" method which uses Hessian approximations to compute the next iterate.

According to numerical tests reported in [2], QN-max, the quasi-Newton method running on (2.55) with $\psi = \psi_{\max}$ and $\phi(.) = \|.\|_F^2$, is competitive with the state-of-the-art codes, ASALM [59], PSPG [4], and ADMIP [6]; and performs better on the test setting where an exponential noise matrix E^o is directly added to low rank component L^o to generate the observation matrix, i.e., $A = L^o + E^o$, instead of perturbing L^o with $S^o + N^o$ – note that this experimental setup violates model assumptions discussed in Section 2.1.2. Indeed, since exponential noise has longer tail than the Gaussian noise, during recovery SPCP algorithms artificially decomposed E^o into sparse component, S^o, and small magnitude, dense noise component, N^o. The authors also used the synthetic experimental setting of Aybat & Iyengar in [6]; and in those experiments the performance of ADMIP is significantly better than QN-max when high accuracy solutions are targeted.

2.2.5 Frank-Wolfe for SPCP

Let Ω be the subset of observable indices of the data matrix $A = L^o + S^o + N^o \in \mathbb{R}^{m\times n}$, Mu et al. [45] proposed stably decomposing A through solving

$$\phi^* := \min_{L,S\in\mathbb{R}^{m\times n}} \{\phi(L,S) = \tfrac{1}{2}\|\pi_\Omega(L+S-A)\|_F^2 : \ \|L\|_* \le \tau_L, \ \|S\|_1 \le \tau_S\} \tag{2.57}$$

for some $\tau_L, \tau_S > 0$. Recall that full or partial SVD computation is the main bottleneck for many of the algorithms discussed in Section 2.2; to avoid this expensive computation, Mu et al. [45] adopted Frank-Wolfe algorithm [21] to solve (2.57), which only requires computing the largest singular value and corresponding left and right singular vectors as opposed to full SVD computation. It is important to note that according to discussion in Section 2.2.4, given $\delta > 0$, there exists $\tau(\delta)$ such that solving (2.57) with $\tau_L = \tau_S = \tau(\delta)$ is equivalent to solving $\min\{\psi_{\max}(L,S) : \ \phi(L,S) \le \delta\}$ with $\lambda_{\max} = 1$. Before discussing further details of the method proposed in [45], as a preliminary we briefly review the Frank-Wolfe method [21] next.

Frank-Wolfe Method

Let $\phi : \mathbb{R}^n \to \mathbb{R}$ be a convex function such that $\nabla\phi$ is Lipschitz continuous with constant L over an open set containing convex, compact set $\mathcal{X}$. Consider the constrained problem

$$\phi^* := \min_{x\in\mathbb{R}^n} \phi(x) \quad \text{s.t.} \quad x \in \mathcal{X}, \tag{2.58}$$

where $D := \max_{x,y\in\mathcal{X}} \|x - y\|_2$. Starting from $x_0 \in \mathcal{X}$, the original Frank-Wolfe (FW) method generates the iterate sequence $\{x_k\}$ as follows: $\bar{x}_k \in \operatorname{argmin}_x \langle\nabla\phi(x_k), x\rangle$, and $x_{k+1} = x_k + \alpha_k(\bar{x}_k - x_k)$, where $\alpha_k = \frac{2}{k+2}$. In [45], the authors considered a more general update rule, which includes the original update scheme as its special case. In particular, any $x_k \in \mathcal{X}$ satisfying $\phi(x_k) \le \phi(x_k + \alpha_k(\bar{x}_k - x_k))$ can be chosen as the next iterate.

Algorithm FW(x_0)

1: **input:** $x_0 \in \mathcal{X}$
2: $k \leftarrow 0$
3: **while** $k \ge 0$ **do**
4: $\quad \bar{x}_k \in \operatorname{argmin}\{\langle\nabla\phi(x_k), x\rangle : \ x \in \mathcal{X}\}$
5: $\quad \alpha_k \leftarrow \frac{2}{k+2}$
6: $\quad$ Choose $x_k \in \mathcal{X}$ such that $\phi(x_k) \le \phi(x_k + \alpha_k(\bar{x}_k - x_k))$
7: $\quad k \leftarrow k + 1$
8: **end while**

FIGURE 2.5 FW: Frank-Wolfe method

The $\mathcal{O}(1/k)$ convergence rate in function values for the algorithm FW, displayed in Figure 2.5, is first given in [16] – see also [45]. In particular, it can be shown that

$$\phi(x_k) - \min_{x\in\mathcal{X}} \phi(x) \le \frac{2LD^2}{k+2}. \tag{2.59}$$

However, note that this error bound depends on some problem parameters, L, D, which may be unknown in practice. As a remedy, Jaggi [36] proposed the surrogate measure $d_k := \langle\nabla\phi(x_k), x_k - \bar{x}_k\rangle$, which can be computed easily without knowing D and L, and it satisfies $\phi(x_k) - \phi^* \le d_k = \mathcal{O}(1/k)$ – see Theorem 2 in [36]. Moreover, Mu et al. [45] have refined this result and shown that for any $K \ge 1$, there exists $k' \le K$ such that $d_{k'} \le \frac{6LD^2}{K+2}$. Hence, given an $\epsilon > 0$, $d_k \le \epsilon$ is a practical stopping criterion.

Combining Frank-Wolfe and projected gradient for SPCP

Note that when FW method in Figure 2.5 is implemented on (2.57), the bottleneck steps are to minimize linear functions over norm-balls, which can be written in the following generic form for some $\bar{L}, \bar{S} \in \mathbb{R}^{m\times n}$:

$$L_+ := -\tau_L \bar{u}\bar{v}^\top \in \operatorname{argmin}\{\langle\bar{L}, L\rangle : \ \|L\|_* \le \tau_L\},$$
$$S_+ := -\tau_S e_i e_j^\top \in \operatorname{argmin}\{\langle\bar{S}, S\rangle : \ \|S\|_1 \le \tau_S\},$$

where $\bar{u} \in \mathbb{R}^m$ and $\bar{v} \in \mathbb{R}^n$ are the left and right singular vectors corresponding to the largest singular value of $\bar{L}$; $e_i \in \mathbb{R}^m$ and $e_j \in \mathbb{R}^n$ are unit vectors with i-th and j-th components are equal to 1, respectively, such that $|\bar{S}_{ij}| = \|\bar{S}\|_\infty$. Hence, computing L_+ is much cheaper than computing the full SVD of $\bar{L}$. On the other hand, as pointed out in [45], one clear disadvantage of FW method on (2.57) is that at every iteration only one entry of the sparse component is updated. This leads to very slow convergence in practice. Hence, Mu et al. [45] proposed combining FW iterations with an additional projected gradient step in S-block.

In particular, they proposed FWP method displayed in Figure 2.5, where after FW iterate $(L_{k+1}, \tilde{S}_{k+1})$ is computed, an extra projected gradient step is computed in Step 9 – note that $\nabla_S \phi(L_{k+1}, \tilde{S}_{k+1}) = \pi_\Omega \left(L_{k+1} + \tilde{S}_{k+1} - A \right)$. According to numerical results in [45], where FWP is tested on video and image processing problems such as foreground extraction from noisy surveillance videos, and shadow and specularity removal from face images, the additional projected gradient step significantly improves the convergence rate observed in practice; moreover, Mu et al. also showed that the FWP iterate sequence, generated as shown in Figure 2.6, satisfies $\phi(L_k, S_k) - \phi^* \leq \frac{16(\tau_L^2 + \tau_S^2)}{k+2}$.

Algorithm FWP

1: **input:** $L_0 = S_0 = \mathbf{0}_{m \times n}$
2: $k \leftarrow 0$
3: **while** $k \geq 0$ **do**
4: $\quad \bar{L}_k \in \operatorname{argmin}\{\langle \pi_\Omega (L_k + S_k - A), L \rangle : \ \|L\|_* \leq \tau_L\}$
5: $\quad \bar{S}_k \in \operatorname{argmin}\{\langle \pi_\Omega (L_k + S_k - A), S \rangle : \ \|S\|_1 \leq \tau_S\}$
6: $\quad \alpha_k \leftarrow \frac{2}{k+2}$
7: $\quad L_{k+1} \leftarrow L_k + \alpha_k(\bar{L}_k - L_k)$
8: $\quad \tilde{S}_{k+1} \leftarrow S_k + \alpha_k(\bar{S}_k - S_k)$
9: $\quad S_{k+1} \leftarrow \operatorname{argmin}\{\|S - (\tilde{S}_{k+1} - \pi_\Omega \left(L_{k+1} + \tilde{S}_{k+1} - A \right))\|_F : \ \|S\|_1 \leq \tau_S\}$
10: $\quad k \leftarrow k + 1$
11: **end while**

FIGURE 2.6 FWP: Frank-Wolfe method with projected gradient step

2.3 Algorithms for non-convex formulations

In Section 2.2, we briefly discuss the algorithms for solving convex optimization problem in (2.8), and related formulations. In the literature, there are several *non-convex* models related to the original SPCP problem. Throughout this section, suppose that the data matrix $A \in \mathbb{R}^{m \times n}$ satisfies (2.2).

In [63] Zhou and Tao proposed enforcing the structure on the desired solution via constraints instead of using $\|.\|_*$ and $\|.\|_1$ as in the SPCP formulation. Indeed, their model ensures low-rank structure on the variable L corresponding to low-rank component, and the sparsity of the variable S corresponding to sparse component by using *rank* and *cardinality constraints*, respectively:

$$\min_{L,S \in \mathbb{R}^{m \times n}} \|L + S - A\|_F^2 \quad \text{s.t.} \quad \mathbf{rank}(L) \leq \bar{r}, \quad \mathbf{card}(S) \leq \bar{s}, \tag{2.60}$$

where $\bar{r}, \bar{s} > 0$ are given model parameters such that $\bar{r} \geq \mathbf{rank}(L^o)$ and $\bar{s} \geq \mathbf{card}(S^o)$. Zhou and Tao [63] proposed GoDec algorithm to solve this model. GoDec is an *alternating minimization* method which alternatingly minimizes the objective in one variable while fixing the other one. They showed that GoDec iterate sequence converges to a local minima, which may be far away from the global optimum. In the implementation, they speed up the algorithm using bilateral random projections instead of computing an SVD for the L-subproblem – see [63] for details. After generating data matrices according to (2.2), the authors compared GoDec with IALM [40], and they report that GoDec is three times faster than IALM according to numerical results in [63] – IALM [40] is an variable penalty ADMM algorithm for the PCP problem in (2.4).

In [33], Hintermüller and Wu considered a slightly modified version of (2.60):

$$\min_{L,S\in\mathbb{R}^{m\times n}} \|L+S-A\|_F^2 + \frac{\rho}{2}\|L\|_F^2 \quad \text{s.t.} \quad \mathbf{rank}(L)\leq \bar{r}, \quad \mathbf{card}(S)\leq \bar{s}, \tag{2.61}$$

where $\bar{r},\bar{s}>0$ are given model parameters as in (2.60), and $0\leq\rho\ll 1$ is a given regularization parameter. The authors proposed an *inexact alternating minimization method* AMS to solve (2.61), where the iterates L_{k+1} and S_{k+1} are computed as "inexact" solutions to subproblems $\min_L\{\|L+S_k-A\|_F^2+\rho\|L\|_F^2 :\ \mathbf{rank}(L)\leq\bar{r}\}$ and $\min_S\{\|L_{k+1}+S-A\|_F^2 :\ \mathbf{card}(S)\leq\bar{s}\}$, respectively. In particular, the authors show that when the iterates (L_k,S_k) satisfy certain inexact optimality conditions for subproblems, if the the iterate sequence has a limit point such that the constraints are tight at the point, then the limit point satisfies the necessary first-order optimality conditions. In the numerical tests, Hintermüller and Wu compared AMS against *constant penalty* ADMM algorithm in [12], of which steps are the same as those of IALM [40] with a constant penalty parameter sequence. It is reported that AMS is approximately 3 times faster than the ADMM in [12] for foreground extraction from surveillance videos.

For the case $N^o=\mathbf{0}_{m\times n}$, given $\bar{r}$ such that $\bar{r}\geq\mathbf{rank}(L^o)$ and the set of observable indices Ω, Shen et al. [56] considered using bilateral factorization $L=UV^\top$ to guarantee low-rank structure

$$\min_{U\in\mathbb{R}^{m\times\bar{r}},V\in\mathbb{R}^{n\times\bar{r}},L\in\mathbb{R}^{m\times n}} \|\pi_\Omega\left(L-A\right)\|_1 \quad \text{s.t.} \quad UV^\top=L. \tag{2.62}$$

Shen et al. proposed three-block ADMM algorithm to solve (2.62). Note that (2.62) is a non-convex problem; and the ADMM algorithms in general even with two-blocks is not necessarily convergent for non-convex problems. In [56], it is shown that if the iterate sequence converges, then it converges to a KKT point of (2.62). According to numerical test results reported in [56], the proposed algorithm is around one order of magnitude faster than IALM [40], and the authors argue that this speedup is a result of solving the factorization model free of any SVD computations. However, the effectiveness of the proposed methods is limited to the cases where the sparse matrix does not dominate the low-rank one in magnitude.

Later in [64], Zhou and Tao proposed the following model for stable decomposition when $N^o\neq\mathbf{0}_{m\times n}$:

$$\min_{U,V,S} \lambda\|S\|_1+\|UV^\top+S-A\|_F^2 \quad \text{s.t.} \quad \mathbf{rank}(U)=\mathbf{rank}(V)\leq\bar{r}, \tag{2.63}$$

where $\lambda>0$ and $\bar{r}$ such that $\bar{r}\geq\mathbf{rank}(L^o)$. Zhou and Tao [64] proposed solving (2.63) using a Greedy Bilateral Smoothing algorithm (GreBsmo), which is based on alternating minimization in three-bloks: U, V and S. The iterate sequence generated by alternating minimization methods may converge to a point which is not optimal, even for convex problems when the objective is *non-smooth*; therefore, GreBsmo does not have strong convergence properties as convex methods discussed in Section 2.2. On the other hand, in [64] the authors report that GreBsmo considerably speed up the decomposition performing 30-100 times faster than both Godec [63] and IALM [40] when applied to foreground extraction problems as in Section 2.1.4.

Suppose that $A=R(L^o+S^o)+N^o$, where R is a given matrix, and L^o, S^o, N^o are low-rank, sparse, and dense noise components. Mardani et al. [44] developed a distributed algorithm for stably decomposing A for detecting traffic flow anomalies within internet protocol networks with a fixed topology, represented as a directed graph. The Lagrangian form of the SPCP problem in (2.9) is a special case of the proposed formulation in [44] when R is the *identity matrix*, and the network is consist of a *single* node. Let $R=\mathbf{I}_m$ and

$A \in \mathbb{R}^{m\times n}$ such that $\bar{r} \geq \mathbf{rank}(L^o)$ for some $\bar{r} > 0$, Mardani et al. exploited the following characterization of $\|.\|_*$ due to Recht et al. [51]:

$$\|L\|_* = \min_{U\in\mathbb{R}^{m\times\bar{r}}, V\in\mathbb{R}^{n\times\bar{r}}} \{\tfrac{1}{2}\|U\|_F^2 + \tfrac{1}{2}\|V\|_F^2 :\ UV^\top = L\}, \tag{2.64}$$

and proposed the following non-convex problem, which is equivalent to the Lagrangian form of the SPCP problem in (2.9):

$$\min_{U\in\mathbb{R}^{m\times\bar{r}}, V\in\mathbb{R}^{n\times\bar{r}}, S\in\mathbb{R}^{m\times n}} \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) + \lambda\|S\|_1 + \frac{\rho}{2}\|UV^\top + S - A\|_F^2. \tag{2.65}$$

To solve (2.65), the authors proposed using a *three-block* ADMM algorithm after variable splitting. In particular, on a single node network, the ADMM formulation considered in [44] is given as

$$\min_{U\in\mathbb{R}^{m\times\bar{r}}, V\in\mathbb{R}^{n\times\bar{r}}, S\in\mathbb{R}^{m\times n}} \left\{\|U\|_F^2 + \|V\|_F^2 + 2\lambda\|S\|_1 + \rho\|UV^\top + \bar{S} - A\|_F^2 :\ S = \bar{S}\right\}. \tag{2.66}$$

Here, (V, S), U, and $\bar{S}$ denotes the three ADMM blocks in which alternatingly the augmented Lagrangian is minimized while the other two blocks are fixed. Unfortunately, there is no convergence guarantees on the ADMM iterate sequence for non-convex problems. Indeed, the iterate sequence may not even have a limit point.

Inspired by the formulation in (2.65), Sprechman et al. [58] proposed combining robust decomposition with dictionary learning. In particular, suppose that $U^o \in \mathbb{R}^{m\times\bar{r}}$ is a known or previously learnt *under-complete dictionary* with $\bar{r} \ll m$ *atoms*, and every random observation $a \in \mathbb{R}^m$ has the following representation: $a = U^o v + s + \eta$, where $v \in \mathbb{R}^{\bar{r}}$, $s \in \mathbb{R}^m$ is a sparse vector, and $\eta \in \mathbb{R}^m$ is dense noise vector such that $\|\eta\|_2 \leq \delta$ for some $\delta > 0$. This interpretation brings the SPCP problem close to the context of dictionary learning in the sparse modeling domain. In particular, if U^o was known, then given an observation $a \in \mathbb{R}^m$, Sprechman et al. [58] proposed solving $\min_{v\in\mathbb{R}^{\bar{r}}, s\in\mathbb{R}^m} \|v\|_2^2 + 2\lambda\|s\|_1 + \rho\|U^o v + s - a\|_F^2$. To make this model more realistic, Sprechman et al. [58] also developed an *online* version of their algorithm that learns U^o while approximately decomposing the i.i.d. data samples $\{a_t\}_{t\in\mathbb{Z}_+}$ arriving sequentially.

Recently, Yuan et al. [39] considered a special case where $L^o = u^o\mathbf{1}_n^\top$, i.e., $\mathbb{R}^{m\times n} \ni A = u^o\mathbf{1}^\top + S^o + N^o$. Recall that as discussed in Section 2.1.4, this model has applications in foreground extraction from noisy surveillance videos when the background is constant. Since this model forces the low-rank component to have rank-1, it cannot capture subtle movements in the background such as moving escalators, flickering lights, etc. and uses sparse, and noise components to capture these changes in the background. Under constant background assumption, Yuan et al. [39] considered the Lagrangian form of the SPCP problem for an appropriately chosen $\rho > 0$:

$$\min_{S\in\mathbb{R}^{m\times n}, u\in\mathbb{R}^m} \|S\|_1 + \frac{\rho}{2}\|u\mathbf{1}_n^\top + S - A\|_F^2.$$

Even under the simplifying rank-1 assumption, this model has no closed-form solution and need to be solved iteratively. Moreover, considering that there might be a blur in a noisy video surveillance video, Yuan et al. [39] also proposed the following extension:

$$\min_{S\in\mathbb{R}^{m\times n}, u\in\mathbb{R}^m} \|S\|_1 + \frac{\rho}{2}\|H(u\mathbf{1}_n^\top + S) - A\|_F^2, \tag{2.67}$$

where $H : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is the regular blurring operator – the blur is assumed to be frame-wise [25]. Yuan et al. [39] proposed three-block ADMM algorithm, which is not necessarily

convergent – see Section 2.2.3, to solve (2.67). On the noisy videos tested in [39], the proposed method performed well in terms of runtime compared to methods solving (2.9), which is mainly because the proposed method does not require computing SVD as $\|.\|_*$ term is dropped from the formulation.

2.4 Conclusion

The data observations in real-life applications are often corrupted by noise, possibly affecting every entry of the data matrix. In this chapter, we reviewed optimization algorithms for decomposing data matrices A that can be decomposed as $A = L^o + S^o + N^o$, where L^o and S^o are low-rank and sparse matrices as in robust PCA, and N^o is a dense noise matrix. When there is no noise in the observations, i.e., $N^o = \mathbf{0}_{m\times n}$ and $A = L^o + S^o$, there are many efficient algorithms for solving the PCP problems in (2.4) and (2.7); on the other hand, in the presence of dense noise, i.e., $N^o \neq \mathbf{0}_{m\times n}$, PCP model becomes inapplicable as A is not a sum of low-rank and sparse components anymore, and for stable decomposition one should consider the SPCP formulation in (2.8), which has uses in numerous applications arising from background separation from surveillance video, shadow and specularity removal from face images and video denoising from heavily corrupted data. Surprisingly there is only few methods that can efficiently deal with (2.8) directly. In the first part of this chapter, we briefly describe ADMM type methods proposed by Tao & Yuan [59], and Aybat & Iyengar [6]; efficient APG implementations proposed by Aybat et al. [3, 4]; and review the convex variational framework proposed by Aravkin et al. [1], and the Frank-Wolfe projected gradient method proposed by Mu et al. [45] for solving a convex problem related to the SPCP formulation. In the second part, we very briefly go over some non-convex variants of the SPCP formulation, and iterative solution methods for them.

References

1. Aleksandr Aravkin, Stephen Becker, Volkan Cevher, and Peder Olsen. A variational approach to stable principal component pursuit. In *30th Conference on Uncertainty in Artificial Intelligence (UAI) 2014*, number EPFL-CONF-199542, 2014. available at arXiv:1406.1089 [math.OC], June, 2014.
2. Aleksandr Y Aravkin, James V Burke, and Michael P Friedlander. Variational properties of value functions. *SIAM Journal on optimization*, 23(3):1689–1717, 2013.
3. Necdet Serhat Aybat, Donald Goldfarb, and Garud Iyengar. Fast first-order methods for stable principal component pursuit. *Published online at arXiv:1105.2126 [math.OC], May, 2011*, 2011.
4. Necdet Serhat Aybat, Donald Goldfarb, and Shiqian Ma. Efficient algorithms for robust and stable principal component pursuit problems. *Computational Optimization and Applications*, 58(1):1–29, 2014. Published online at arXiv:1105.2126 [math.OC], September, 2013.
5. Necdet Serhat Aybat and Garud Iyengar. A unified approach for minimizing composite norms. *Mathematical Programming*, 144(1-2):181–226, 2014. Published online at arXiv:1005.4733 [math.OC], May, 2010.
6. Necdet Serhat Aybat and Garud Iyengar. An alternating direction method with increasing penalty for stable principal component pursuit. *Computational Optimization and Applications*, pages 1–34, 2015. Published online at arXiv:1309.6553 [math.OC], September, 2013.
7. Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

8. Dimitri P Bertsekas, John N Tsitsiklis, and Athena Scientific. Partial solutions manual parallel and distributed computation: Numerical methods. 2003.
9. J. M. Borwein and Q. J. Zhu. *Techniques Of Variational Analysis*. Springer, 2005.
10. Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
11. Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.
12. Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011.
13. Venkat Chandrasekaran, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.
14. Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, pages 1–23, 2014.
15. Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61, 1998.
16. V. F. Demyanov and A. M. Rubinov. *Approximate Methods in Optimization Problems*. Modern Analytic and Computational Methods in Science and Mathematics. American Elsevier Publishing Company, Inc., 1970.
17. Wei Deng, Ming-Jun Lai, Zhimin Peng, and Wotao Yin. Parallel multi-block admm with $o(1/k)$ convergence. *arXiv preprint arXiv:1312.3040*, 2013.
18. David L Donoho and Iain M Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995.
19. J Eckstein and W Yao. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 32, 2012.
20. Jonathan Eckstein and Dimitri P Bertsekas. On the douglasrachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.
21. Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.
22. M. Fukushima. Application of the alternating direction method of multipliers to separable convex programming problems. *Computational Optimization and Applications*, 1(1):93–111, 1992.
23. R. Glowinski. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*. Studies in Mathematics and its Applications. Elsevier Science, 2000.
24. Osman Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM Journal on Control and Optimization*, 29(2):403–419, 1991.
25. P.C. Hansen, J.G. Nagy, and D.P. O'Leary. *Deblurring Images: Matrices, Spectra, and Filtering*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, 2006.
26. B. He and H. Yang. Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities. *Operations Research Letters*, 23:151–161, 1998.
27. B. S. He, L. Z. Liao, D. R. Han, and H. Yang. A new inexact alternating directions method for monontone variational inequalities. *Mathematical Programming, Se-*

ries A, 92:103–118, 2002.

28. Bingsheng He, Liusheng Hou, and Xiaoming Yuan. On full jacobian decomposition of the augmented lagrangian method for separable convex programming. *SIAM J. Optim., under revision*, 2013.
29. Bingsheng He and Xiaoming Yuan. On the **cone**$(1/n)$ convergence rate of the douglas-rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.
30. B.S. He, H. Yang, and S.L. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications*, 106(2):337–356, 2000.
31. Magnus R Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.
32. Nicholas J Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103:103–118, 1988.
33. Michael Hintermüller and Tao Wu. Robust principal component pursuit via inexact alternating minimization on matrix manifolds. *Journal of Mathematical Imaging and Vision*, 51(3):361–377, 2015.
34. Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*, 2012.
35. Kaizhan Huai, Mingfang Ni, Feng Ma, and Zhanke Yu. A customized proximal point algorithm for stable principal component pursuit with nonnegative constraint. *Journal of Inequalities and Applications*, 2015(1), 2015.
36. Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 427–435, 2013.
37. S. Kontogiorgis and R. R. Meyer. A variable-penalty alternating direction method for convex optimization. *Mathematical Programming*, 83:29–53, 1998.
38. Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Ensemble nystrom method. In *Advances in Neural Information Processing Systems*, pages 1060–1068, 2009.
39. Xinxin Li, Michael K. Ng, and Xiaoming Yuan. Median filtering-based methods for static background extraction from surveillance video. *Numerical Linear Algebra with Applications*, 2015.
40. Zhouchen Lin, Arvind Ganesh, John Wright, Leqin Wu, Minming Chen, and Yi Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 61, 2009.
41. P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979.
42. Shiqian Ma, Donald Goldfarb, and Lifeng Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.
43. Lester W Mackey, Michael I Jordan, and Ameet Talwalkar. Divide-and-conquer matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1134–1142, 2011.
44. Morteza Mardani, Gonzalo Mateos, and Georgios B Giannakis. Unveiling anomalies in large-scale networks via sparsity and low rank. In *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*, pages 403–407. IEEE, 2011.
45. Cun Mu, Yuqian Zhang, John Wright, and Donald Goldfarb. Scalable robust matrix recovery: Frank-wolfe meets proximal methods. *arXiv preprint*, 2014. Published online at arXiv:1403.7588 [math.OC], March, 2014.

46. Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.
47. Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.
48. Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $\mathcal{O}(1/k^2)$. In *Doklady AN SSSR*, volume 269, pages 543–547, 1983.
49. Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.
50. M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, 1969.
51. Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.
52. R Tyrell Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and applications*, 12(6):555–562, 1973.
53. R Tyrell Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of operations research*, 1(2):97–116, 1976.
54. R Tyrrell Rockafellar. A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical Programming*, 5(1):354–373, 1973.
55. R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.
56. Yuan Shen, Zaiwen Wen, and Yin Zhang. Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29(2):239–263, 2014.
57. Naum Zuselevich Shor, Krzysztof C Kiwiel, and Andrzej Ruszcayǹski. *Minimization methods for non-differentiable functions*. Springer-Verlag New York, Inc., 1985.
58. Pablo Sprechmann, Alex M Bronstein, and Guillermo Sapiro. Learning robust low-rank representations. *arXiv preprint arXiv:1209.6393*, 2012.
59. Min Tao and Xiaoming Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011. Published online at `http://www.optimization-online.org/`, December, 2009.
60. Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.
61. Ermin Wei and Asuman Ozdaglar. On the $\mathbf{cone}(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. *arXiv preprint arXiv:1307.8254*, 2013.
62. John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009.
63. Tianyi Zhou and Dacheng Tao. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *The Proceedings of the* 28^{th} *International Conference on Machine Learning, Bellevue, WA, USA*.
64. Tianyi Zhou and Dacheng Tao. Greedy bilateral sketch, completion & smoothing. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 650–658, 2013.
65. Zihan Zhou, Xiaodong Li, John Wright, Emmanuel Candes, and Yi Ma. Stable principal component pursuit. In *Information Theory Proceedings (ISIT), 2010 IEEE*

International Symposium on, pages 1518–1522. IEEE, 2010.