Journal of Computational Mathematics
Vol.38, No.2, 2020, 337–354.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1809-m2018-0106

ROBUST INEXACT ALTERNATING OPTIMIZATION FOR MATRIX COMPLETION WITH OUTLIERS*

Ji Li
Beijing Computational Science Research Center, Beijing 100193, China
Email: keelee@csrc.ac.cn
Jian-Feng Cai
Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
Email: jfcai@ust.hk
Hongkai Zhao
Department of Mathematics, University of California, Irvine, CA, USA
Email: zhao@uci.edu

Abstract

We investigate the problem of robust matrix completion with a fraction of observation corrupted by sparsity outlier noise. We propose an algorithmic framework based on the ADMM algorithm for a non-convex optimization, whose objective function consists of an ℓ_1 norm data fidelity and a rank constraint. To reduce the computational cost per iteration, two inexact schemes are developed to replace the most time-consuming step in the generic ADMM algorithm. The resulting algorithms remarkably outperform the existing solvers for robust matrix completion with outlier noise. When the noise is severe and the underlying matrix is ill-conditioned, the proposed algorithms are faster and give more accurate solutions than state-of-the-art robust matrix completion approaches.

Mathematics subject classification: 65K05, 90C06, 93C41.
Key words: Matrix completion, ADMM, Outlier noise, Inexact projection.

1. Introduction

The problem of matrix completion refers to completing a matrix with many missing entries, and it arises from various applications in statistics, machine learning, and computer vision. This problem is possible to solve only if the underlying matrix is "simple", because otherwise the matrix contains too much information to infer from the limited observed entries. A commonly used notion of simplicity for matrices is *low rank* [11,13,17,32,33], which provides redundancy of matrix entries. The low rank matrix model is remarkably successful in many applications in machine learning, such as collaborative filtering [38] and the Netflix prize problem [4].

Let $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ be the underlying low-rank matrix we would like to estimate. Let $\Omega \subseteq [m] \times [n]$ be a set of indices with $|\Omega| \ll mn$, where $[m] = \{1, 2, \ldots, m\}$ and the same for $[n]$. In the problem of matrix completion, only the entries $\{M_{ij} : (i, j) \in \Omega\}$ are observed and the other entries are missing. One would like to reconstruct the underlying low-rank matrix $\boldsymbol{M}$ from $\{M_{ij} : (i, j) \in \Omega\}$. The approaches of low-rank matrix completion can be divided into two categories, namely, convex and non-convex optimization based approaches.

* Received May 31, 2018 / Revised version received August 2, 2018 / Accepted September 25, 2018 / Published online February 13, 2019 /

Convex optimization based approaches are formulated from the rank minimization. With sufficiently many observed entries and some mild assumptions, $\boldsymbol{M}$ is the only low-rank matrix in the set $\{\boldsymbol{X} \in \mathbb{R}^{m\times n} \mid X_{ij} = M_{ij}, (i,j) \in \Omega\}$ of matrices that are consistent with observed entries. In this case, the low-rank matrix completion can be reconstructed by solving the following constrained rank minimization:

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \quad & \operatorname{rank}(\boldsymbol{X}) \\ \text{s.t.} \quad & X_{ij} = M_{ij}, \quad (i,j) \in \Omega. \end{aligned} \tag{1.1}$$

However, (1.1) is non-convex and, more critically, NP-hard. Therefore, it is computationally intractable. To overcome these, a popular strategy is to replace the rank function in (1.1) by its convex relaxation, the nuclear norm [11–13], to solve the following convex optimization

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \quad & \|\boldsymbol{X}\|_* \\ \text{s.t.} \quad & X_{ij} = M_{ij}, \quad (i,j) \in \Omega. \end{aligned} \tag{1.2}$$

Problem (1.2) can be reformulated as a special case of semi-definite programming (SDP) [37], for which polynomial time solvers exist. It was proved in [11,13,22,36] that the unique solution of (1.2) is $\boldsymbol{M}$ under suitable assumptions. Thus, one can complete a low-rank matrix in polynomial time with theoretical guarantee. In real applications, the observed entries are usually corrupted by noise. Under this circumstance, it is natural to consider the nuclear norm regularized optimization

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \quad \frac{1}{2}\sum_{(i,j)\in\Omega}(X_{ij} - \tilde{M}_{ij})^2 + \lambda\|\boldsymbol{X}\|_*, \tag{1.3}$$

where $\tilde{M}_{ij}$, $(i,j) \in \Omega$, are noisy observations. When there is only a small amount of noise in the observed entries, the model (1.3) is provably accurate with the reconstruction error proportional to the noise level [11]. Though off-the-shelf SDP solvers can be applied to solve (1.2) and (1.3), numerically, they are not the most efficient, especially when the matrix size is moderately large. Customized first-order algorithms (e.g., [8,29,31,41]) are developed for solving (1.2) and (1.3). Most of them invoke the singular value thresholding (SVT) operator [8] at each iteration. The most expensive part of these algorithms is the computation of SVT in each iteration. The usual strategy is to compute the singular value decomposition (SVD) followed by the soft-thresholding on the singular values.

To improve the performance of nuclear norm optimization based matrix completion, we may consider non-convex optimization based approaches. Assume that $\operatorname{rank}(\boldsymbol{M}) = r$ is known, then the matrix completion problem can be reformulated as the following constrained least-squares problem

$$\begin{aligned} \min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \quad & \sum_{(i,j)\in\Omega}(X_{ij} - \tilde{M}_{ij})^2 \\ \text{s.t.} \quad & \operatorname{rank}(\boldsymbol{X}) = r. \end{aligned} \tag{1.4}$$

Since (1.4) is a non-convex optimization, the challenge here is how to find the global minimum with a provable guarantee. In the past a few years, there is a burst of research works on the design and analysis of provable non-convex matrix completion algorithms by solving (1.4) and its variants. There are two types of such numerical algorithms. One type of algorithms treat the

unknown matrix $\boldsymbol{X}$ as an element in the set of all rank-r matrices. Projected gradient descent algorithms are presented, termed as singular value projection (SVP) [25] or normalized iterative hard thresholding (NIHT) [40]. Furthermore, it is well known that the set of fixed rank matrices is a smooth Riemannian manifold [1]. By utilizing the geometry of smooth manifolds, more efficient algorithms are obtained, see, e.g., [5, 33, 34, 42–44]. Guarantees of finding $\boldsymbol{M}$ of these algorithms are provided in [25, 43, 44]. The other type of numerical algorithms for solving (1.4) parametrize the unknown low-rank matrix in factorization forms. This leads to non-convex unconstrained optimization problems, and various standard optimization algorithms can be applied, see, e.g., [21, 26, 30, 38, 39, 45]. Theoretical guarantees are provided based on either the local property around the underlying solution [26, 30, 39] or the global geometric landscape of the objective functions [21].

The aforementioned matrix completion algorithms work well only when there is no noise or the noise is not severe, as they are based on the formulation of least-squares data fidelity. It is well known that the least-squares formulation is not well suited for tasks where the observation is corrupted by noise with some outliers. Therefore, it is necessary to study robust matrix completion in the presence of outliers. This is the main theme of this paper. We propose a family of efficient non-convex optimization based algorithms for matrix completion from observed entries corrupted by noise with outliers. Our algorithms are derived from the acceleration of alternating direction method of multipliers (ADMM) for a rank constrained optimization, where the projection onto the set of rank-r matrices (i.e., truncated SVD) is approximated by several efficient and effective schemes. Numerical experiments demonstrate that our proposed algorithms are robust not only to the outliers in the noisy observation, but also to the fast decay of the non-zero singular values of the underlying low-rank matrix.

In the rest of this section, we give a brief review on robust low-rank matrix completion algorithms that can deal with observations corrupted by noise with outliers.

1.1. Priori Arts

The simplest model for noise with outliers might be the impulsive noise model. In this model, only a portion of the observed entries are contaminated by noise, for which the observed values are very different from the truth, and the others are clean. Therefore, the noise can be modelled by a sparse vector. Following this model, it was proposed in [10, 14, 28] to solve the optimization problems in below

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{e}} \quad & \gamma \|\boldsymbol{X}\|_* + \|\boldsymbol{e}\|_{\ell_1} \\ \text{s.t.} \quad & \mathcal{P}_\Omega \boldsymbol{X} = \mathcal{P}_\Omega \tilde{\boldsymbol{M}} + \boldsymbol{e}. \end{aligned} \tag{1.5}$$

Here $\mathcal{P}_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a projector defined as follows: for all $\boldsymbol{X} \in \mathbb{R}^{m\times n}$,

$$[\mathcal{P}_\Omega(\boldsymbol{X})]_{ij} = \begin{cases} \boldsymbol{X}_{ij}, & \text{if } (i,j) \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

The nuclear norm $\|\boldsymbol{X}\|_*$ and the ℓ_1-norm $\|\boldsymbol{e}\|_{\ell_1}$ promote the low rank of the reconstructed matrix $\boldsymbol{X}$ and the sparsity of the noise vector $\boldsymbol{e} \in \mathbb{R}^{|\Omega|}$ respectively. It is proven that the low rank matrix can be recovered by a solution of (1.5) with high probability from only a small fraction of the entries without any assumptions on the location or the amplitude of the corrupted entries in [16]. When $\Omega = [m] \times [n]$ (i.e. all the entries are observed with possible

impulsive noise), we only need to separate the low-rank matrix and the sparse error, which is known as robust principal component analysis (RPCA) [7, 10, 20, 35]. Eq. (1.5) gives the exact RPCA [10]. The impulsive model is sometimes too simple. We may assume, besides the impulsive noise, the observed entries are also polluted by a small amount of additive noise affecting all the observed entries. That is, $\mathcal{P}_\Omega \boldsymbol{X} = \mathcal{P}_\Omega \tilde{\boldsymbol{M}} + \boldsymbol{e}_1 + \boldsymbol{e}_2$, where $\boldsymbol{e}_1$ is a sparse vector modelling the impulsive noise and $\boldsymbol{e}_2$ represents the small additive noise. This gives the following optimization

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{e}_1,\boldsymbol{e}_2} \quad & \lambda \|\boldsymbol{X}\|_* + \gamma \|\boldsymbol{e}_1\|_{\ell_1} + \frac{1}{2}\|\boldsymbol{e}_2\|_{\ell_2}^2 \\ \text{s.t.} \quad & \mathcal{P}_\Omega \boldsymbol{X} = \mathcal{P}_\Omega \tilde{\boldsymbol{M}} + \boldsymbol{e}_1 + \boldsymbol{e}_2. \end{aligned} \tag{1.6}$$

To solve convex optimizations (1.5) and (1.6), SVT based algorithms such as ADMM [10] can be used. They generally require the SVD of an $m \times n$ matrix per iteration. When the rank of iteration matrices is not small compared to the matrix size, the computation could be prohibited for large scale problems.

Similar to the standard matrix completion, non-convex optimization methods generally give more efficient algorithms than convex ones. The unknown low-rank matrix is represented by either its factorization or an element in the set of all low-rank matrices. There are two different ways to deal with noise with outliers. Some of non-convex robust matrix completion still use the ℓ_1 norm or its variants [2, 3, 9, 19, 24, 47]. For example, [3, 24] aim to solving the following non-convex optimization

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{S}} \quad & \|\mathcal{P}_\Omega(\boldsymbol{S})\|_{\ell_1} \\ \text{s.t.} \quad & \mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{V} + \boldsymbol{S}) = \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) \\ & \boldsymbol{U} \in Gr(m,r), \quad \boldsymbol{V} \in \mathbb{R}^{r\times n}, \end{aligned}$$

where $Gr(m,r)$ is the Grassmannian manifold, i.e., the set of linear r-dimensional subspaces of $\mathbb{R}^m$. These non-convex optimization methods generally give more efficient algorithms than convex ones. Some other non-convex matrix completion directly uses the sparse constraint [7, 15, 35, 48]. For example, [48] proposed to solve

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{S}} \quad & \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{U}\boldsymbol{V} + \boldsymbol{S}) - \mathcal{P}_\Omega(\tilde{\boldsymbol{M}})\|_F^2 + \lambda\|\boldsymbol{U}^T\boldsymbol{U} - \boldsymbol{V}^T\boldsymbol{V}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{U} \in \mathbb{R}^{m\times r} \cap \mathcal{C}, \boldsymbol{V} \in \mathbb{R}^{n\times r} \cap \mathcal{C}, \ \|\boldsymbol{S}\|_0 \le pmn, \end{aligned}$$

where $\mathcal{C}$ is the set of matrices satisfying the so-called incoherence condition, and $\lambda > 0$ is a regularization parameter. Similar to the standard matrix completion case, these non-convex approaches usually give more efficient algorithms for robust matrix completion than convex ones.

1.2. Our contribution

In this paper, we aim at devising a computationally efficient and noise-robust algorithm for matrix completion. We first formulate the robust matrix completion problem to an optimization problem with rank constraint. Then, we propose an algorithmic framework to solve the resulting optimization, based on the alternating direction method of multipliers (ADMM). To minimize the computational cost, a critical ingredient in our proposed framework is to approximate the

truncated SVD by inexact projections. Two inexact projections are studied to get two efficient robust matrix completion algorithms. One of them is deduced by projecting onto tangent space of the low rank matrix manifold, and the other is based on power iteration. Our approach only needs an initial rank estimation, we also apply the rank adaption to reach a better recovery especially for matrix with exponentially decaying singular values. We demonstrate that two inexact projections lead to efficient robust matrix completion algorithms that are faster than the original algorithm with the exact projection. Numerical experiments are presented to demonstrate the efficiency of the proposed algorithms.

2. Algorithmic Framework

In this section, we propose our algorithmic framework for robust low-rank matrix completion. The proposed framework is based on alternating direction method of multipliers (ADMM) for some non-convex optimization problems, and another key ingredient is to approximate the projection onto the set of low-rank matrices. With different approximations of the projection, our framework will lead to several efficient robust matrix completion algorithms. In this section, we first briefly review the robust matrix completion as set feasibility problem and the deduced alternating projection method, and then the inexact projection introduced latter can be used to accelerate the computation of ADMM.

2.1. Matrix Completion as Set Feasibility

In order to present our algorithmic framework, we first reformulate the matrix completion problem as a set feasibility problem in this section. For this purpose, we introduce some additional notations. We assume that $\operatorname{rank}(\boldsymbol{M}) = r$ is known. Let $\mathcal{M}$ be the set of all matrices with rank at most r

$$\mathcal{M} = \{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \operatorname{rank}(\boldsymbol{X}) \leq r\}.$$

Let $\mathcal{N}$ be the set of all matrices whose entries related to the observations are bounded by the noise level ϵ on Ω, i.e.,

$$\mathcal{N} = \left\{\boldsymbol{X} \in \mathbb{R}^{m\times n} : \left\|\mathcal{P}_\Omega(\boldsymbol{X} - \tilde{\boldsymbol{M}})\right\|_{\ell_1} \leq \epsilon\right\}.$$

Therefore, the matrix completion is equivalent to the set feasibility problem

$$\text{find} \quad \boldsymbol{X} \in \mathcal{M} \cap \mathcal{N}. \tag{2.1}$$

We introduce the projection operation, for a given set $\mathcal{S}$, the projection of a point $\boldsymbol{Z} \notin \mathcal{S}$ onto it, is defined as

$$\mathcal{P}_{\mathcal{S}}(\boldsymbol{Z}) := \arg\min_{\boldsymbol{X}\in\mathcal{S}} \|\boldsymbol{X} - \boldsymbol{Z}\|_{\ell_2}^2 .$$

Projection onto the set $\mathcal{M}$ is straightforward, by Eckart-Young theorem, the projection $\mathcal{P}_{\mathcal{M}}(\boldsymbol{Z})$ is given by the partial SVD of $\boldsymbol{Z}$. Projection onto the set $\mathcal{N}$ is involved with the projection onto simplex ℓ_1 ball [18]. The alternating projection is just projection onto the two sets until convergence. Given $\boldsymbol{Z}_0$, the iteration scheme is

$$\boldsymbol{X}_k = \mathcal{P}_{\mathcal{M}}(\boldsymbol{Z}_k), \qquad \boldsymbol{Z}_{k+1} = \mathcal{P}_{\mathcal{N}}(\boldsymbol{X}_k).$$

Indeed, the projection $\mathcal{P}_{\mathcal{N}}$ can be expressed in the form

$$\boldsymbol{Z}_{k+1} = \boldsymbol{X}_k + \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) + \mathcal{P}_\Omega(\boldsymbol{N}_k) - \mathcal{P}_\Omega(\boldsymbol{X}_k), \tag{2.2}$$

where $\mathcal{P}_\Omega(\boldsymbol{N}_k)$ is the projection of $\mathcal{P}_\Omega(\boldsymbol{X}_k) - \mathcal{P}_\Omega(\tilde{\boldsymbol{M}})$ onto set $\{\boldsymbol{S} \in \mathbb{R}^{m\times n} : \|\mathcal{P}_\Omega(\boldsymbol{S})\|_{\ell_1} \leq \epsilon\}$. Given initial $\boldsymbol{X}_0$, the overall iteration is

$$\boldsymbol{X}_{k+1} = \mathcal{P}_{\mathcal{M}}\left(\boldsymbol{X}_k + \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) + \mathcal{P}_\Omega(\boldsymbol{N}_k) - \mathcal{P}_\Omega(\boldsymbol{X}_k)\right). \tag{2.3}$$

For the noiseless matrix completion, i.e., the entries over the sample set Ω are completely consistent with the available observations. In this case, $\mathcal{P}_\Omega(\boldsymbol{N}_k) \equiv 0$ in (2.3), and it is identical to the SVP algorithm [25] with a special choice of step size. When $\mathcal{P}_\Omega$ is replaced by a linear operator satisfying the restricted isometric condition (RIP) [37], the convergence of (2.3) to $\boldsymbol{M}$ is theoretically guaranteed [25]. However, since $\mathcal{P}_\Omega$ does not satisfy RIP, there is no theoretical guarantee of (2.3) for matrix completion. In [42–44], (2.3) is accelerated by projecting onto the tangent space of the Riemannian manifold formed by $\mathcal{M}$ before $\mathcal{P}_\Omega$, obtaining the gradient descent algorithm on Riemannian manifold. More importantly, it was shown in [43] that the Riemannian optimization algorithm converges to $\boldsymbol{M}$ for matrix completion. For the harder robust principle component analysis (RPCA), (2.3) can be viewed as adaptions of the alternating projection algorithm [35] to the case where the support of the sparse matrix is fixed in RPCA. Guarantees of such algorithms for RPCA are provided in [35]. Again, (2.3) is accelerated by Riemannian manifold tangent space projection in [7] and the theoretical guarantee is also provided there.

2.2. Non-Convex Alternating Direction Methods of Multipliers

In this section, we present our algorithmic framework for robust matrix completion based on ADMM of a non-convex objective function. Similar to existing approaches, we use ℓ_1 norm term to deal with the possible outliers in the noise. More precisely, we consider the following ℓ_1 norm based optimization with rank constraint

$$\min_{\boldsymbol{X}\in\mathcal{M}} \quad \left\|\mathcal{P}_\Omega(\tilde{\boldsymbol{M}} - \boldsymbol{X})\right\|_{\ell_1}. \tag{2.4}$$

Define the indicator function $\iota_{\mathcal{M}}$ of the set $\mathcal{M}$ by

$$\iota_{\mathcal{M}}(\boldsymbol{X}) = \begin{cases} +\infty & \text{if } \boldsymbol{X} \notin \mathcal{M}, \\ 0 & \text{otherwise.} \end{cases}$$

By introducing the auxiliary variable $\boldsymbol{Z}$ and using $\iota_{\mathcal{M}}$, (2.4) is transformed into an optimization with an equality constraint

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Z}} \quad & \left\|\mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{Z})\right\|_{\ell_1} + \iota_{\mathcal{M}}(\boldsymbol{X}) \\ \text{s.t.} \quad & \boldsymbol{X} = \boldsymbol{Z}. \end{aligned} \tag{2.5}$$

We can solve (2.5) by ADMM [6], though it is developed for convex optimization problems. The augmented Lagrangian function associated with (2.5) is

$$\mathcal{L}_\beta(\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{\Lambda}) = \left\|\mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{Z})\right\|_{\ell_1} + \iota_{\mathcal{M}}(\boldsymbol{X}) + \langle \boldsymbol{\Lambda}, \boldsymbol{X} - \boldsymbol{Z}\rangle + \frac{\beta}{2}\|\boldsymbol{X} - \boldsymbol{Z}\|_{\ell_2}^2,$$

where $\boldsymbol{\Lambda}$ is the dual variable and $\beta > 0$ is a parameter. Then applying ADMM leads to the following iteration

$$\begin{cases} \boldsymbol{X}_{k+1} = \arg\min_{\boldsymbol{X}} \mathcal{L}_\beta(\boldsymbol{X}, \boldsymbol{Z}_k, \boldsymbol{\Lambda}_k), \\ \boldsymbol{Z}_{k+1} = \arg\min_{\boldsymbol{Z}} \mathcal{L}_\beta(\boldsymbol{X}_{k+1}, \boldsymbol{Z}, \boldsymbol{\Lambda}_k), \\ \boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + \nu\beta(\boldsymbol{X}_{k+1} - \boldsymbol{Z}_{k+1}), \end{cases} \tag{2.6}$$

where $\nu > 0$ is a stepsize.

All three subproblems in (2.6) have closed form solutions. Given $\boldsymbol{Z}_0$ and $\boldsymbol{\Lambda}_0 = 0$, by simple calculation, the solution of the first subproblem is given by $\boldsymbol{X}_{k+1} = \mathcal{P}_{\mathcal{M}}(\boldsymbol{Z}_k - \frac{1}{\beta}\boldsymbol{\Lambda}_k)$. The solution of the second subproblem is

$$\begin{cases} \mathcal{P}_\Omega(\boldsymbol{Z}_{k+1}) = \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathrm{prox}_{\beta^{-1}\|\cdot\|_1}\left(\mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right) \\ \mathcal{P}_{\Omega^c}(\boldsymbol{Z}_{k+1}) = \mathcal{P}_{\Omega^c}(\boldsymbol{X}_{k+1} + \frac{1}{\beta}\boldsymbol{\Lambda}_k), \end{cases}$$

where $\mathrm{prox}_{\beta^{-1}\|\cdot\|_1}(u) = \mathrm{sgn}(u)\max(|u| - \beta^{-1}, 0)$ is the soft-thresholding operator. It can be verified that the entries over Ω^c of $\boldsymbol{\Lambda}$ is always zero at the iteration, if $\boldsymbol{\Lambda}_0 = 0$. This sparsity pattern can be integrated into the iteration scheme to get an equivalent iteration of (2.6) as follows

$$\begin{cases} \boldsymbol{X}_{k+1} = \mathcal{P}_{\mathcal{M}}\left(\boldsymbol{Z}_k - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right) \\ \mathcal{P}_\Omega(\boldsymbol{Z}_{k+1}) = \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathrm{prox}_{\beta^{-1}\|\cdot\|_1}\left(\mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right) \\ \mathcal{P}_{\Omega^c}(\boldsymbol{Z}_{k+1}) = \mathcal{P}_{\Omega^c}(\boldsymbol{X}_{k+1} + \frac{1}{\beta}\boldsymbol{\Lambda}_k) \\ \mathcal{P}_\Omega(\boldsymbol{\Lambda}_{k+1}) = \mathcal{P}_\Omega(\boldsymbol{\Lambda}_k) + \beta\left(\mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \mathcal{P}_\Omega(\boldsymbol{Z}_{k+1})\right). \end{cases} \tag{2.7}$$

Let us further simplify the iteration. We denote

$$\boldsymbol{R}_{k+1} := \underset{\beta^{-1}\|\cdot\|_1}{\mathrm{prox}}\left(\mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right).$$

Then $\boldsymbol{Z}_{k+1}$ is expressed by

$$\boldsymbol{Z}_{k+1} = \boldsymbol{X}_{k+1} + \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \boldsymbol{R}_{k+1}, \tag{2.8}$$

which implies

$$\mathcal{P}_\Omega(\boldsymbol{\Lambda}_{k+1}) = \mathcal{P}_\Omega(\boldsymbol{\Lambda}_k) + \beta\left(\boldsymbol{R}_{k+1} - \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) + \mathcal{P}_\Omega(\boldsymbol{X}_{k+1})\right). \tag{2.9}$$

Plugging (2.8) and (2.9) into the updating formula for $\boldsymbol{X}_{k+2}$ gives

$$\begin{aligned} &\boldsymbol{X}_{k+2} = \mathcal{P}_{\mathcal{M}}\left(\boldsymbol{Z}_{k+1} - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_{k+1})\right) \\ =&\mathcal{P}_{\mathcal{M}}\left(\boldsymbol{X}_{k+1} + \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \boldsymbol{R}_{k+1} - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) + \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \boldsymbol{R}_{k+1}\right) \\ =&\mathcal{P}_{\mathcal{M}}\left(\boldsymbol{X}_{k+1} + 2\left(\mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right) - 2\boldsymbol{R}_{k+1} + \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right). \end{aligned}$$

We define

$$\boldsymbol{Y}_{k+1} := \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_{k+1}) - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k). \tag{2.10}$$

We also shift the index of $\boldsymbol{X}_{k+1}$ back by 1, i.e., use $\boldsymbol{X}_k$ to denote $\boldsymbol{X}_{k+1}$ in (2.7) with little notation abuse. Thus, given $\boldsymbol{X}_0$ and $\boldsymbol{\Lambda}_0 = 0$, the ADMM iteration (2.6), (2.7) is equivalent to

$$\begin{cases} \boldsymbol{Y}_{k+1} = \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{X}_k) - \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k), \\ \boldsymbol{R}_{k+1} = \operatorname{prox}_{\beta^{-1}\|\cdot\|_{\ell_1}}(\boldsymbol{Y}_{k+1}), \\ \boldsymbol{X}_{k+1} = \mathcal{P}_{\mathcal{M}}\left(\boldsymbol{X}_k + 2\boldsymbol{Y}_{k+1} - 2\boldsymbol{R}_{k+1} + \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k)\right), \\ \mathcal{P}_\Omega(\boldsymbol{\Lambda}_{k+1}) = \mathcal{P}_\Omega(\boldsymbol{\Lambda}_k) + \beta\left(\boldsymbol{R}_{k+1} - \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) + \mathcal{P}_\Omega(\boldsymbol{X}_{k+1})\right). \end{cases} \tag{2.11}$$

From (2.11), we see that the $\boldsymbol{X}_{k+1}$ is expressed by the projection of the matrix $\boldsymbol{X}_k$ with a sparsity matrix onto the set $\mathcal{M}$. The matrices in (2.11) are either of rank-r (cf. $\boldsymbol{X}$) or sparse supported on Ω (cf. $\boldsymbol{Y}, \boldsymbol{R}, \boldsymbol{\Lambda}$). Obviously, both type of matrices can be stored with a small memory. For example, the low rank matrices are stored in their low rank factorization form. Therefore, the spatial complexity of the algorithm is very low. This structure is favorable for large-scale problems.

Let us examine the computational complexity of (2.11). We see that all operations in (2.11) except for $\mathcal{P}_{\mathcal{M}}$ are entry-wise and hence very cheap to implement. Therefore, the computational bottleneck of (2.11) is the evaluation of $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}_{k+1})$, where

$$\boldsymbol{W}_{k+1} = \boldsymbol{X}_k + 2\boldsymbol{Y}_{k+1} - 2\boldsymbol{R}_{k+1} + \frac{1}{\beta}\mathcal{P}_\Omega(\boldsymbol{\Lambda}_k).$$

We need to compute the leading r singular values and the corresponding singular vectors of the $m \times n$ matrix $\boldsymbol{W}_{k+1}$, which is a combination of a low rank matrix and a sparse matrix. This structure of the matrix $\boldsymbol{W}_{k+1}$ will facilitate the truncated SVD calculation.

Compared to the alternating projection method in literature [27], except the two parameters γ and β, which can be adapted as typically in ADMM, our non-convex ADMM framework is parameter free. Actually, our algorithmic framework is not sensitive to the two parameters. To cope with sparsity outlier noise, [27] considers the data fidelity set

$$\mathcal{N} = \left\{ \boldsymbol{X} \in \mathbb{R}^{m\times n} \middle| \left\| \mathcal{P}_\Omega(\boldsymbol{X}) - \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) \right\|_{\ell_1} \le \epsilon \right\}.$$

The algorithmic performance of alternating projection obviously depends on the a priori noise level estimation ϵ, which is not known in advance in practice. Besides, the projection onto $\mathcal{N}$ is involved the projection onto simplex ℓ_1 ball [18], it may be an extra computational cost. Every projection step onto $\mathcal{N}$ needs doing a quick sort.

Our ADMM framework for robust matrix completion can be easily applied to the matrix completion with Gaussian additive noise, the little modification of (2.11) is the update of matrix $\boldsymbol{R}_{k+1}$. In the case, it should be $\boldsymbol{R}_{k+1} = \frac{\beta}{1+\beta}\boldsymbol{Y}_{k+1}$ if we consider the ℓ_2 norm data fidelity in (2.5). It can also be extended to the robust matrix completion for the mixed Gaussian additive noise and impulsive noise. We only apply the same method for the following problem

$$\begin{aligned} \min_{\boldsymbol{X},\boldsymbol{Z},\boldsymbol{S}} \quad & \left\| \mathcal{P}_\Omega(\tilde{\boldsymbol{M}}) - \mathcal{P}_\Omega(\boldsymbol{Z}) \right\|_{\ell_1} + \frac{\gamma}{2}\left\| \mathcal{P}_\Omega(\boldsymbol{S}) \right\|_{\ell_2}^2 + \iota_{\mathcal{M}}(\boldsymbol{X}) \\ \text{s.t.} \quad & \boldsymbol{X} = \boldsymbol{Z} + \mathcal{P}_\Omega(\boldsymbol{S}). \end{aligned}$$

The parameter γ is to trade-off the two types of noise.

3. Two Inexact Projections

The most time-consuming step in our ADMM algorithm (2.11) is the evaluation of $\mathcal{P}_{\mathcal{M}}$, the projection onto $\mathcal{M}$, which is the exact global minimizer of the following problem

$$\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}) = \arg\min_{\boldsymbol{X}} \quad \|\boldsymbol{X}-\boldsymbol{W}\|_F^2, \quad \text{s.t. } \operatorname{rank}(\boldsymbol{X}) \le r. \tag{3.1}$$

By Eckart-Young theorem, the best rank-r approximation of matrix $\boldsymbol{W}$ can be obtained by the truncated SVD. Let $\boldsymbol{W} = \sum_{i=1}^{\min(m,n)} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ ($\sigma_1 \ge \sigma_2 \ge ...$) be its SVD, then $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}) = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$. To get the exact projection, we need to compute the top r singular values and their associated singular vectors of $\boldsymbol{W}$. Though there are fast SVD packages available for only the leading singular values and singular vectors, the computation of $\mathcal{P}_{\mathcal{M}}$ might be quite slow, especially for large scaled problems.

In order to accelerate the computation of the ADMM iteration (2.11), we propose to use inexact projections to approximate $\mathcal{P}_{\mathcal{M}}$. These inexact projections should be cheap to compute and able to keep the fast convergence of (2.11). We present two such inexact projections in Sections 3.1 and 3.2 respectively. The first inexact projection is motivated by the Riemannian manifold structure of the set of all low-rank matrix matrices, and the second inexact projection is obtained by a few of iterations of an SVD solver with a random initialization.

3.1. Algorithm I: Tangent Space Projection

It is well known that, when $\mathcal{M}$ is embedded into $\mathbb{R}^{m\times n}$, it forms a smooth Riemannian manifold [1]. At iteration k, the tangent space $\mathcal{L}_k$ of $\mathcal{M}$ at $\boldsymbol{X}_k$ is a good approximation to $\mathcal{M}$. Therefore, instead of projecting the iteration matrix directly onto $\mathcal{M}$, we propose to first project it onto the tangent space $\mathcal{L}_k$ and then $\mathcal{M}$. In other words, we approximate $\mathcal{P}_{\mathcal{M}}$ is approximated by $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}$. This strategy has been also used in [7, 43, 44].

At a first glance, the successive projection strategy $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}$ takes more computational cost than the direct projection $\mathcal{P}_{\mathcal{M}}$. But it is not the truth. Indeed, for the given iteration matrix $\boldsymbol{W}_{k+1}$, the computation of $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1})$ can be done without SVD of size $m \times n$, so that it can be significantly faster than $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}_{k+1})$. To see this, we break down the computation of $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1})$ into the following two substeps.

- *Project $\boldsymbol{W}_{k+1}$ onto the tangent space to get $\boldsymbol{L}_{k+1} := \mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1})$.* Let $\boldsymbol{X}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T$ be the compact SVD of $\boldsymbol{X}_k$. Then, the tangent space $\mathcal{L}_k$ has an explicit form as in the following

  $$\mathcal{L}_k = \left\{\boldsymbol{U}_k\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{V}_k^T \mid \boldsymbol{A}\in\mathbb{R}^{n\times r},\ \boldsymbol{B}\in\mathbb{R}^{m\times r}\right\}.$$

  That is, $\mathcal{L}_k$ is the direct sum of the two subspaces of matrices whose column and row spaces are the same as those of $\boldsymbol{X}_k$ respectively. Therefore, the projection onto the tangent space has a closed form

$$\begin{aligned}
\boldsymbol{L}_{k+1} &= \mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1}) = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T + \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{W}_{k+1} + \boldsymbol{W}_{k+1}\boldsymbol{V}_k\boldsymbol{V}_k^T - \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{W}_{k+1}\boldsymbol{V}_k\boldsymbol{V}_k^T \\
&= \begin{bmatrix}\boldsymbol{U}_k & (\boldsymbol{I}-\boldsymbol{U}_k\boldsymbol{U}_k^T)\boldsymbol{W}_{k+1}\boldsymbol{V}_k\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_k + \boldsymbol{U}_k^T\boldsymbol{W}_{k+1}\boldsymbol{V}_k & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_k^T \\ \boldsymbol{U}_k^T\boldsymbol{W}_{k+1}(\boldsymbol{I}-\boldsymbol{V}_k\boldsymbol{V}_k^T)\end{bmatrix} \\
&:= \begin{bmatrix}\boldsymbol{U}_k & \hat{\boldsymbol{U}}_k\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_k + \hat{\boldsymbol{M}}_k & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_k & \hat{\boldsymbol{V}}_k\end{bmatrix}^T,
\end{aligned}$$

where

$$\hat{\boldsymbol{U}}_k = (\boldsymbol{I} - \boldsymbol{U}_k\boldsymbol{U}_k^T)\boldsymbol{W}_{k+1}\boldsymbol{V}_k, \tag{3.2a}$$

$$\hat{\boldsymbol{M}}_k = \boldsymbol{U}_k^T\boldsymbol{W}_{k+1}\boldsymbol{V}_k, \tag{3.2b}$$

$$\hat{\boldsymbol{V}}_k = (\boldsymbol{I} - \boldsymbol{V}_k\boldsymbol{V}_k^T)\boldsymbol{W}_{k+1}^T\boldsymbol{U}_k. \tag{3.2c}$$

Thus, to get $\boldsymbol{L}_{k+1}$, we only need to compute $\hat{\boldsymbol{U}}_k$, $\hat{\boldsymbol{M}}_k$, and $\hat{\boldsymbol{V}}_k$ respectively. This can be done efficiently by $O(r)$ matrix-vector products.

- *Project $\boldsymbol{L}_{k+1}$ onto the low-rank manifold to get $\mathcal{P}_{\mathcal{M}}(\boldsymbol{L}_{k+1})$.* We do not need to call an SVD subroutine straightforwardly for the $m \times n$ matrix $\boldsymbol{L}_{k+1}$. Notice that $\boldsymbol{L}_{k+1}$ is of rank $2r$ and it is given in a factorization form. Therefore, its SVD can be reduced into two QR decompositions and one SVD of size $2r \times 2r$. See the details in Algorithm 3.1.

Algorithm 3.1 Inexact projection onto fixed rank matrix manifold $\mathcal{M}$

Input: Given matrices $\boldsymbol{W}_{k+1}$, and $\boldsymbol{X}_k = \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T$ in its compact SVD.
Output: $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1}) = \boldsymbol{U}_{k+1}\boldsymbol{\Sigma}_{k+1}\boldsymbol{V}_{k+1}^T$ in its compact SVD.
1: Compute $\hat{\boldsymbol{U}}_k$, $\hat{\boldsymbol{M}}_k$, and $\hat{\boldsymbol{V}}_k$ according to (3.2).
2: Compute QR decomposition: $(\boldsymbol{Q}_u, \boldsymbol{R}_u) = qr(\hat{\boldsymbol{U}}_k, 0)$ and $(\boldsymbol{Q}_v, \boldsymbol{R}_v) = qr(\hat{\boldsymbol{V}}_k, 0)$
3: Form matrix $\hat{\boldsymbol{S}} = \begin{bmatrix} \boldsymbol{\Sigma}_k + \hat{\boldsymbol{M}}_k & \boldsymbol{R}_v^T \\ \boldsymbol{R}_u & \boldsymbol{0} \end{bmatrix}$
4: Compute $(\boldsymbol{U}_s, \boldsymbol{\Sigma}_s, \boldsymbol{V}_s) = svd(\hat{\boldsymbol{S}})$, where the singular values are sorted non-increasingly.
5: $\boldsymbol{\Sigma}_{k+1} = \boldsymbol{\Sigma}_s(1:r, 1:r)$, $\boldsymbol{U}_{k+1} = [\boldsymbol{U}_k\ \boldsymbol{Q}_u]\boldsymbol{U}_s(:, 1:r)$, and $\boldsymbol{V}_{k+1} = [\boldsymbol{V}_k\ \boldsymbol{Q}_v]\boldsymbol{V}_s(:, 1:r)$

In summary, the computation of $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1})$ is reduced into $O(r)$ matrix-vector products, two QR decompositions of size $n \times r$ and $m \times r$ respectively, and one SVD of size $2r \times 2r$. The total computational cost is $18mr^2 + 16nr^2 + 8|\Omega|r + Cr^3$, where C is a constant depending on the SVD routine. Compared to $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}_{k+1})$ for which an SVD of size $m \times n$ is needed, the approximation $\mathcal{P}_{\mathcal{M}}\mathcal{P}_{\mathcal{L}_k}(\boldsymbol{W}_{k+1})$ is much more computationally efficient. Furthermore, since tangent spaces are good approximations to the manifold, this approximation strategy will not increase the number of iterations needed, as we will see in the numerical experiments.

3.2. Algorithm II: Inexact Projection via One Step of Power Iteration

We can also approximate $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}_{k+1})$ by applying a few iterations of some SVD solvers. One of the simplest yet effective algorithms for singular value problems might be the power iteration. Let $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ be an approximation of the top r left singular vectors of $\boldsymbol{W}_{k+1}$. Since the left singular vectors are also eigenvectors of $\boldsymbol{W}_{k+1}\boldsymbol{W}_{k+1}^T$, we can refine $\boldsymbol{U}$ by the following power iteration

$$\boldsymbol{U} \leftarrow orth(\boldsymbol{W}_{k+1}\boldsymbol{W}_{k+1}^T\boldsymbol{U}),$$

where $orth(\cdot)$ orthogonalize the matrix, i.e., taking the orthogonal factor in the QR decomposition. Under very mild assumptions on $\boldsymbol{W}_{k+1}$ and the initial $\boldsymbol{U}$, the power iteration will converge to the top r singular vectors of $\boldsymbol{W}_{k+1}$. By introducing a new matrix $\boldsymbol{V} = \boldsymbol{W}_{k+1}^T\boldsymbol{U} \in \mathbb{R}^{n\times r}$, we get an equivalent formulation of the power iteration

$$\boldsymbol{U} \leftarrow orth(\boldsymbol{W}_{k+1}\boldsymbol{V}), \quad \boldsymbol{V} \leftarrow \boldsymbol{W}_{k+1}^T\boldsymbol{U} \tag{3.3}$$

Since $\boldsymbol{U}$ converges to the singular vectors, it is easy to see that $\boldsymbol{U}\boldsymbol{V}^T$ converges to $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}_{k+1})$.

Thus, to approximate $\mathcal{P}_{\mathcal{M}}(\boldsymbol{W}_{k+1})$, we propose to use only one step of (3.3) with warm start or random initial guess. This one-step power iteration approximation is done in one compact QR decomposition and one matrix-vector multiplication. The computational cost is $14mr^2 + 4nr^2 + 8|\Omega|r$, which again is significantly smaller than the exact truncated SVD. Furthermore, this approximation is effective and it will not increase the number of iteration needed in ADMM.

4. Simulation Tests

We validate the performance of our approaches for synthetic low rank matrix completion. We focus on the square matrices, since the performance for nonsquare matrix completion shows the similar behavior. In our all experiments, synthetic data are created with exact rank. We do the experiments in MATLAB R2012a on a desktop computer with a 3.30 GHz CPU and 4GB of memory.

The testing problems are synthesized as follows. We first generate the underlying low-rank matrix $\boldsymbol{M}$, discussed in detail in the successive sections. Then, entries of $\boldsymbol{M}$ are sampled according to the Bernoulli model. For each index $(i, j) \in [n] \times [m]$, it is included in Ω with probability $\frac{|\Omega|}{mn}$. To simulate the outliers in the noise, we assume an observed entry is contaminated by outlier noise independently randomly with probability p. If an observed entry is corrupted by outlier noise, then the observed value is

$$\tilde{M}_{ij} = M_{ij} + \mathcal{S}_{\pm 1} \cdot \mathcal{N}(\mu, \sigma^2),$$

where $\mathcal{S}_{\pm 1}$ are independent random variables taking values $+1$ or -1 with equal probability, and $\mathcal{N}(\mu, \sigma^2)$ are independent Gaussian random variable with mean μ and variance σ^2. The parameters p, μ, and σ control how severe the noise is. Larger p, μ, σ means more severe noise.

By incorporating the two inexact projections in Sections 3.1 and 3.2 into the ADMM framework (2.11), we can get two proposed algorithms, called ADMM3 and ADMM2 respectively. We can also use them to replace the truncated SVD in the alternating projection algorithm (2.3), and the resulting algorithms are called AP3 and AP2 respectively. We will compare these algorithms with RMC [9], AOPMC [46], GRASTA [3, 24] and the AP method [27]. The algorithmic parameters of these methods are set as the default. For the RMC, the maximum number of CG iterations is set to 40 with a gradient tolerance of 10^{-8} and $\delta_0 = 1, \theta = 0.05$. For AOPMC, since we known the number of outliers, we directly provide it to the algorithm. For AP, since we know the noise level ϵ, we also provide it to the algorithm. For AOPMC and AP, the algorithmic performance depends on the number of outliers and noise level. For practical problem, AOPMC needs to run the algorithm several times to guess the number of outliers, which increases the time cost. For AP, we also need to estimate the noise level. Our ADMM algorithms are completely parameter-free. For GRASTA, it aims to perform online matrix completion, there would be some trouble running the algorithm on large-scale problem. Its cost time is proportional to the matrix size, since it operates one column at a time. For RMC and AOPMC, the quality of the solution to the inner problem is very crucial for the convergence of the algorithms toward the exact underlying low rank matrix. For this reason, the default gradient tolerance is set to 10^{-8}.

Though the proposed ADMM algorithmic framework scales well for large-scale problem, we consider the medium-scale matrix completion with size 500×500 for comparison, as the

same performance is obtained for large-scale case. We fix the penalty $\beta = 1$ in our ADMM methods. The initialization $\boldsymbol{X}_0$ for all tested algorithms is set to the sparsity matrix $\mathcal{P}_\Omega(\tilde{\boldsymbol{M}})$, and respectively we generate $\boldsymbol{U}_0, \boldsymbol{\Sigma}_0, \boldsymbol{V}_0$ from the top r left and right singular vectors and singular values of $\mathcal{P}_\Omega(\tilde{\boldsymbol{M}})$. Note that the effect of the initialization does not have a significant influence on the convergence of the algorithms. To inspect the algorithmic progress, we will monitor how the RMSE (root mean square error) deceases, where the RMSE is defined as

$$\mathrm{RMSE}(\boldsymbol{X}, \boldsymbol{M}) = \sqrt{\frac{\sum_{i,j}(X_{ij} - M_{ij})^2}{mn}}.$$

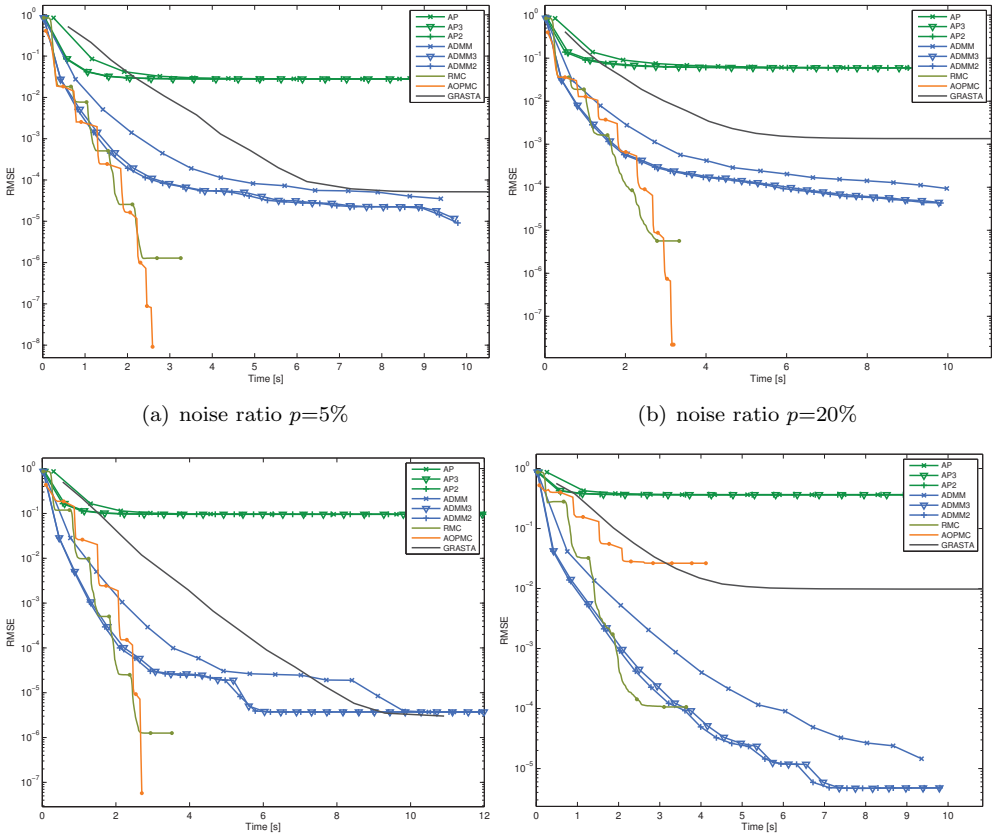
4.1. Gaussian matrices

We first illustrate the performance of the proposed algorithms for problems where the underlying low-rank matrix is generated by the product of two Gaussian matrices. Let r be the rank. We form $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ with their entries drawn from i.i.d. Gaussian, and define $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}^T$. Obviously, $\boldsymbol{M}$ is of rank r. Furthermore, from random matrix theory, $\boldsymbol{M}$ is well-conditioned, i.e., the ratio of the first and r-th singular values of $\boldsymbol{M}$ is small. These Gaussian low-rank matrices are widely used in the literature [12,13,23,26] for testing numerical performances of different algorithms. The tested matrix size is 500×500 with known rank 10. The oversampling (OS) ratio is set to 4.

We test the performance of the algorithms under different noise settings, namely, $p = 5\%$ or $p = 20\%$, and $\mu = \sigma = 0.1$ or $\mu = \sigma = 1$. Recall that p is the ratio of the outlier noise, and μ and σ relate to the severeness of the noise. Larger p, μ, σ mean more severe noise hence harder problems. We report in Fig. 4.1 the RMSE against the run time all the algorithms. We see that, in all cases, both the inexact projections can accelerate the algorithms with the exact projection, and there is no significant difference between the two inexact projections. For the hard problems where $p = 20\%$ and $\mu = \sigma = 1$, our proposed algorithms ADMM2 and ADMM3 are the fastest among all tested algorithms. Most importantly, the convergence curve of the proposed algorithms ADMM2 and ADMM3 do not change too much with respect to the severeness of the noise. As a comparison, the performance of other algorithms (e.g. AOPMC) decay drastically when the noise becomes more severe. This indicates the proposed algorithms ADMM2 and ADMM3 are more robust to the noise and more suitable for matrix completion under severe noise with outliers. They outperform other algorithms for harder problems.

To further demonstrate this, we consider two more challenging problems: $p = 5\%$, $\mu = \sigma = 5$, and oversampling ratio 4; and $p = 5\%$, $\mu = \sigma = 1$, the oversampling ratio is 2. The results are displayed in Fig. 4.2. The conclusion is still the same: our proposed algorithms ADMM2 and ADMM3 converges faster than other algorithms.

We also compare all algorithms for difference p and $\mu = \sigma$. The noise ratio p is chosen from 0 to 20% with step size 2.5%. The μ and σ are chosen from $\{0.1, 1, 2, 5, 10\}$. For each parameter setting, we generate ten random test problems, and report the average RMSE after 20 seconds run of a particular algorithm. The results are summaized in Table 4.1, where we only list the results for $\mu = \sigma = 1$ and $\mu = \sigma = 5$. For other parameters, the performance is similar. Since the two inexact projection perform similar, we report only results of AP3 and ADMM3. We can see that, for larger probability $p \geq 17.5\%$, the proposed ADMM3 algorithm outperforms all other methods. Again, contrary to other algorithms (e.g. AOPMC), the performance of the ADMM3 algorithm is insensitive to the severeness of the noise. Therefore, our ADMM3 algorithm has the advantage of no tune parameter and stability. To further verify the scalability



(a) noise ratio p=5% (b) noise ratio p=20%

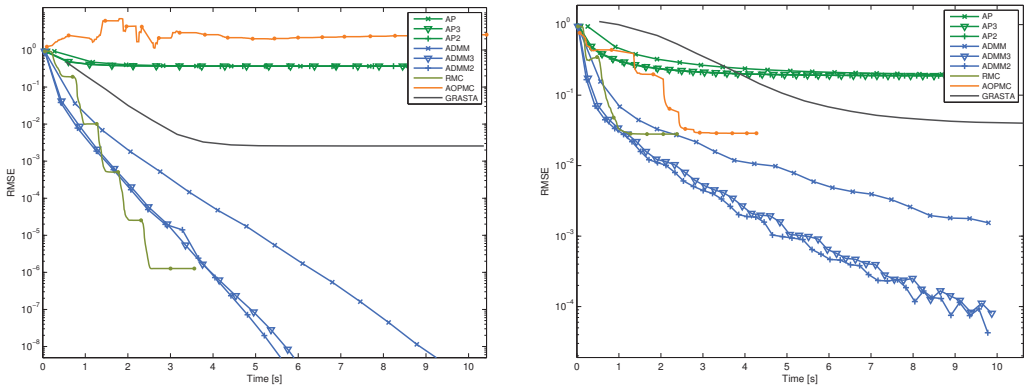
Fig. 4.1. Convergence curves for Gaussian matrix completion, outliers of top is created with $\mu = \sigma = 0.1$, The bottom is created with $\mu = \sigma = 1$.

Table 4.1: Comparison of tested methods for 500×500 matrices.

	$\mu = \sigma = 1$					$\mu = \sigma = 5$				
p	AP3	ADMM3	RMC	AOPMC	GRASTA	AP3	ADMM3	RMC	AOPMC	GRASTA
0.000	3.09e-15	1.44e-15	9.32e-07	5.98e-11	8.30e-05	3.09e-15	1.44e-15	9.32e-07	5.98e-11	6.28e-07
0.025	8.32e-04	1.88e-15	8.42e-07	2.75e-08	3.90e-05	1.45e-04	1.52e-15	8.42e-07	8.04e-01	2.45e-03
0.050	4.45e-03	3.17e-14	1.27e-06	2.47e-08	2.99e-04	5.81e-03	1.68e-07	3.22e-03	1.22e+00	4.97e-03
0.075	6.31e-03	8.29e-08	1.78e-06	2.58e-08	1.86e-03	1.50e-02	8.57e-08	2.95e-02	1.42e+00	5.55e-03
0.100	9.44e-03	5.98e-07	2.08e-06	2.55e-08	1.40e-03	1.87e-02	4.70e-07	3.01e-01	1.61e+00	8.75e-03
0.125	7.78e-03	6.93e-07	3.26e-04	2.87e-08	3.64e-03	3.63e-02	3.52e-07	8.82e-01	1.76e+00	1.52e-02
0.150	9.97e-03	1.35e-06	3.11e-04	3.92e-08	4.32e-03	3.48e-02	4.51e-07	9.89e-01	1.84e+00	2.34e-02
0.175	1.05e-02	9.16e-07	6.92e-04	8.67e-05	7.75e-03	9.67e-02	2.54e-07	1.03e+00	1.97e+00	4.49e-02
0.200	1.90e-02	1.39e-06	3.58e-03	1.62e-03	1.53e-02	1.60e-01	5.63e-07	1.03e+00	2.04e+00	4.75e-02

of our proposed methods, we also test matrix with size of 1000×1000 and with known rank of 10, 20, 30 and 50. The parameters are set to as before. As for median dimension 500×500 case, our proposed method ADMM3 performs much better than other methods when the corrupted noise is large, such as $\mu = \sigma = 5$ and corruption fraction p is large. We list the comparisons with other methods for cases $\mu = \sigma = 5$ in Tables 4.2 and 4.3. These tests demonstrate the efficiency of our method ADMM3.

DOI https://doi.org/10.4208/jcm.1809-m2018-0106 | Generated on 2024-12-19 01:44:08



(a) probability $p = 5\%$, $\mu = \sigma = 5$ (b) probability $p = 5\%$, oversampling OS=2

Fig. 4.2. Convergence curves for Gaussian matrix completion.

Table 4.2: Comparison of tested methods for 1000×1000 matrices with $\mu = \sigma = 5$.

	rank $r = 10$					rank $r = 20$				
p	AP3	ADMM3	RMC	AOPMC	GRASTA	AP3	ADMM3	RMC	AOPMC	GRASTA
0.000	7.98e-08	3.68e-08	2.29e-06	1.12e-12	9.64e-07	3.04e-07	2.12e-07	6.26e-08	1.12e-11	1.00e-05
0.025	2.07e-01	6.05e-08	8.68e-07	9.30e-01	7.90e-05	1.84e-01	1.91e-06	8.08e-07	4.07e-01	1.79e-05
0.050	3.69e-01	3.36e-07	1.28e-06	1.37e+00	5.85e-04	3.35e-01	8.65e-06	1.21e-06	8.06e-01	2.13e-05
0.075	5.24e-01	2.39e-06	2.74e-01	1.58e+00	2.56e-03	4.96e-01	1.22e-05	1.58e-06	1.34e+00	1.99e-04
0.100	6.81e-01	1.86e-06	1.46e-01	1.72e+00	3.74e-03	6.60e-01	1.34e-05	1.96e-06	1.45e+00	3.21e-04
0.125	8.33e-01	5.86e-06	3.92e-01	1.83e+00	7.94e-03	8.12e-01	1.62e-05	6.34e-02	1.65e+00	3.09e-04
0.150	9.35e-01	9.08e-06	9.84e-01	1.90e+00	1.41e-02	9.30e-01	2.54e-05	3.18e-01	1.78e+00	5.93e-04
0.175	9.96e-01	2.05e-05	1.01e+00	2.07e+00	4.43e-02	1.01e+00	2.74e-05	1.01e+00	1.91e+00	4.02e-03
0.200	1.01e+00	4.96e-05	1.01e+00	2.09e+00	5.78e-02	1.03e+00	4.69e-05	1.02e+00	1.99e+00	1.24e-02

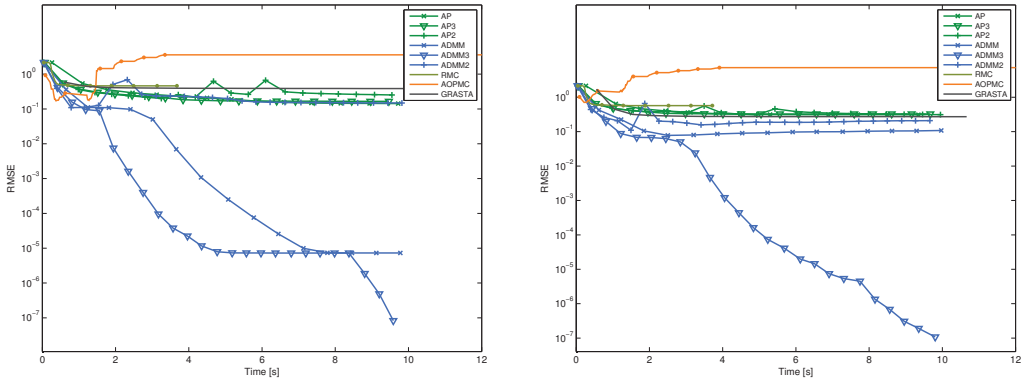
Table 4.3: Comparison of tested methods for 1000×1000 matrices with $\mu = \sigma = 5$.

	rank $r = 30$					rank $r = 50$				
p	AP3	ADMM3	RMC	AOPMC	GRASTA	AP3	ADMM3	RMC	AOPMC	GRASTA
0.000	5.22e-07	2.89e-07	6.70e-08	8.43e-08	6.62e-05	1.85e-07	7.43e-08	1.06e-06	9.34e-10	4.03e-04
0.025	1.69e-01	1.58e-05	7.94e-07	3.78e-01	1.17e-04	1.61e-01	2.28e-05	3.10e-04	7.07e-01	7.08e-04
0.050	3.24e-01	2.71e-05	1.19e-06	6.70e-01	2.70e-04	3.13e-01	4.44e-05	4.58e-04	1.01e+00	1.47e-03
0.075	4.90e-01	3.26e-05	1.55e-06	1.29e+00	4.25e-04	4.86e-01	5.91e-05	5.99e-04	1.24e+00	1.59e-02
0.100	6.70e-01	3.91e-05	3.87e-05	1.47e+00	8.57e-04	6.75e-01	8.39e-05	7.34e-04	1.43e+00	4.18e-03
0.125	8.35e-01	4.69e-05	1.31e-01	1.62e+00	1.97e-03	8.75e-01	1.00e-04	1.76e-02	1.60e+00	6.78e-03
0.150	9.66e-01	6.23e-05	3.93e-01	1.77e+00	2.76e-03	1.03e+00	1.36e-04	6.90e-02	1.74e+00	1.05e-02
0.175	1.04e+00	6.59e-05	1.02e+00	1.89e+00	7.08e-03	1.16e+00	1.61e-04	6.11e-01	1.88e+00	1.74e-02
0.200	1.07e+00	1.25e-04	1.06e+00	2.00e+00	2.77e-02	1.20e+00	2.54e-04	1.14e+00	1.99e+00	3.26e-02

4.2. Matrices with exponentially decaying singular values

We test also matrix completion problems where the underlying low-rank matrix has exponentially decaying singular values. We create two random matrices $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ with i.i.d. Gaussian entries. Let $orth(\boldsymbol{U})$ and $orth(\boldsymbol{V})$ be their orthogonalization respectively. A non-negative diagonal matrix Σ with exponentially decaying diagonals is generated by the Matlab command `logspace(-10*log10(CN),0,r)`, where CN is a parameter controlling the

DOI https://doi.org/10.4208/jcm.1809-m2018-0106 | Generated on 2024-12-19 01:44:08



(a) probability $p = 1\%$, $\mu = \sigma = 1$ (b) probability $p = 1\%$, $\mu = \sigma = 5$

Fig. 4.3. Convergence curves for matrix completion with decaying singular value.

rate of decay. Then, the testing low-rank matrix is defined by $\boldsymbol{M} = orth(\boldsymbol{U}) \cdot \boldsymbol{\Sigma} \cdot orth(\boldsymbol{V})^T$. Obviously, these $\boldsymbol{M}$ have large condition numbers. Thus, they are more challenging testing examples than those in the previous section. The tested algorithms are the same as before.

The underlying low-rank matrix is of size 500 with rank 10. The oversampling ratio is 4. The noise ratio is set to 1% and the magnitude of the noise are $\mu = \sigma = 1$ and $\mu = \sigma = 5$ respectively. The noise can not be set large for this hard completing problem. We plot the convergence curves in terms of computational time in Fig. 4.3. We see again both the two inexact projections discussed in Section 3 brings significant acceleration in ADMM and AP. Different to the case of well-conditioned underlying low-rank matrices, the inexact projection via tangent space projection gains more acceleration than the inexact projection via one-step of power iteration. That is, ADMM3 and AP3 are faster than their counterparts ADMM2 and AP3 respectively. Furthermore, the proposed algorithm ADMM3 is the fastest among all tested algorithms.

For this harder case, the rank adaption is utilized in our ADMM algorithmic framework. If the rank adaption is not applied. The recovery RMSE of ADMM method will stagnate at the level of the RMC. From Fig. 4.3, the ADMM3 generally outperforms ADMM2. The rank increasing is very straightforward for the exact projection ADMM and ADMM3. For exact projection, we just locate SVD with additional rank. For ADMM3 inexact projection, we get the rank r approximation is by truncating a $2r \times 2r$ matrix. The ADMM2 inexact projection generally appends a random column, which will result in the increasing of the RMSE temporally. Overall, the ADMM3 outperforms other methods for the hard matrix completion.

5. Conclusion

We consider the algorithm for robust matrix completion with a fraction of observed entries corrupted by outlier noise. We first formulate the robust matrix completion as a set feasibility problem and solve it by the alternating projection algorithm. We then devise the ADMM framework with rank constraint and ℓ_1-norm data fidelity. The ADMM algorithm uses only a small amount of memory, and it involves only sparse matrices and low-rank matrices. Consequently, our ADMM framework scales well for large-scale problem. Though there exist the penalty strength and stepsize to tune in our ADMM algorithmic framework, the performance is not sensitive to the parameter. Once we fix the penalty and stepsize parameters, unlike the

related works for robust matrix completion, our framework is almost parameter free.

To save computational cost, we propose two inexact projections to accelerate the truncated SVD, the most time-consuming step in the ADMM algorithm. By embedding the inexact projections into ADMM, we form obtain our proposed algorithms ADMM2 and ADMM3. The numerical simulation validates the robustness and efficiency of the proposed algorithm. Both inexact projections are effective in accelerating the original ADMM algorithm. Furthermore, the proposed algorithms ADMM2 and ADMM3 outperforms state-of-the-art robust matrix completion algorithms, especially for those challenging test problems with severe noise, small oversampling ratio, and/or ill-conditioned underlying matrix.

Acknowledgments. JL was supported by China Postdoctoral Science Foundation grant No. 2017M620589. JFC was supported in part by Hong Kong Research Grant Council (HKRGC) grants 16300616 and 16306317. HK Zhao was supported in part by NSF grants DMS-1418422 and DMS-1622490.

References

[1] P.A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, 2009.

[2] L. Balzano, R. Nowak, and B. Recht, Online identification and tracking of subspaces from highly incomplete information, **26** (2010), 704–711.

[3] L. Balzano, A. Szlam, and J. He, Incremental gradient on the grassmannian for online foreground and background separation in subsampled video, In *Computer Vision and Pattern Recognition*, (2012), 1568–1575.

[4] J. Bennett, S. Lanning, et al, The netflix prize, In *Proceedings of KDD Cup and Workshop*, volume 2007, p 35. New York, NY, USA, 2007.

[5] N. Boumal and P.a. Absil, Rtrmc: A riemannian trust-region method for low-rank matrix completion, In *Advances in Neural Information Processing Systems*, (2011), 406–414.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Foundations and Trends® in Machine learning*, **3**:1 (2011), 1–122.

[7] H. Cai, J.F. Cai, and K. Wei, Accelerated alternating projections for robust principal component analysis, *arXiv preprint arXiv:1711.05519*, 2017.

[8] J.F. Cai, E.J. Candès, and Z. Shen, A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.*, **20**:4 (2010), 1956–1982.

[9] L. Cambier and P.A. Absil, Robust low-rank matrix completion by riemannian optimization, *SIAM J. Sci. Comput.*, **38**:5 (2016), S440–S460.

[10] E.J. Candès, X. Li, Y. Ma, and J. Wright, Robust principal component analysis, *J. ACM*, **58**:3 (2011), 11.

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

[12] E.J. Candès and B. Recht, Exact matrix completion via convex optimization, *Foundations of Computational mathematics*, **9**:6 (2009), 717.

[13] E.J. Candès and T. Tao, The power of convex relaxation: Near-optimal matrix completion, *IEEE Trans. Inf. Theory*, **56**:5 (2010), 2053–2080.

[14] Y. Chen, A. Jalali, S. Sanghavi, and C. Caramanis, Low-rank matrix recovery from errors and erasures, In *IEEE International Symposium on Information Theory Proceedings*, 2011, 2313–2317.

[15] Y. Chen and M.J. Wainwright, Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees, *arXiv preprint arXiv:1509.03025*, 2015.

[16] Y. Chen, H. Xu, C. Caramanis, and S. Sanghavi, Robust matrix completion with corrupted columns, *IEEE Trans. Inf. Theory*, **62**:1 (2011), 503–526.

[17] M.A. Davenport and J. Romberg, An overview of low-rank matrix recovery from incomplete observations, *IEEE J. Sel. Topics Signal Process.*, **10**:4 (2016), 608–622.

[18] Duchi, John, ShalevShwartz, Shai, Singer, Yoram, Chandra, and Tushar, Efficient projections onto the l1-ball for learning in high dimensions, In *International Conference on Machine Learning*, 2008, 272–279.

[19] A. Eriksson and A. Van Den Hengel, Efficient computation of robust low-rank matrix approximations in the presence of missing data using the l 1 norm, In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, 771–778. IEEE.

[20] H. Gao, J.F. Cai, Z. Shen, and H. Zhao, Robust principal component analysis-based four-dimensional computed tomography, *Phys. Med. Biol.*, **56**:11 (2011), 3181.

[21] R. Ge, J.D. Lee, and T. Ma, Matrix completion has no spurious local minimum, In *Advances in Neural Information Processing Systems*, (2016), 2973–2981.

[22] D. Gross, Recovering low-rank matrices from few coefficients in any basis, *IEEE Trans. Inf. Theory*, **57**:3 (2011), 1548–1566.

[23] M. Hardt, Understanding alternating minimization for matrix completion, In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, (2014), 651–660. IEEE.

[24] J. He, L. Balzano, and J.C.S. Lui, Online robust subspace tracking from partial information, *Mathematics*, 2011.

[25] P. Jain, R. Meka, and I.S. Dhillon, Guaranteed rank minimization via singular value projection, In *Advances in Neural Information Processing Systems*, (2010), 937–945.

[26] P. Jain, P. Netrapalli, and S. Sanghavi, Low-rank matrix completion using alternating minimization, In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, (2013), 665–674. ACM.

[27] X. Jiang, Z. Zhong, X. Liu, and H.C. So, Robust matrix completion via alternating projection, *IEEE Signal Process. Lett.*, **24**:5 (2017), 579–583.

[28] X. Li, Compressed sensing and matrix completion with constant proportion of corruptions, *Constructive Approximation*, **37**:1 (2011), 73–99.

[29] Y.J. Liu, D. Sun, and K.-C. Toh, An implementable proximal point algorithmic framework for nuclear norm minimization, *Math. Program.*, **133**:1 (2012), 399–436.

[30] C. Ma, K. Wang, Y. Chi, and Y. Chen, Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution, *arXiv preprint arXiv:1711.10467*, 2017.

[31] S. Ma, D. Goldfarb, and L. Chen, Fixed point and bregman iterative methods for matrix rank minimization, *Math. Program.*, **128**:1 (2011), 321–353.

[32] M. Michenkov, Numerical algorithms for low-rank matrix completion problems, **2** (2011), 1–24.

[33] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre, Fixed-rank matrix factorizations and riemannian low-rank optimization, *Computational Statistics*, **29**:3-4 (2014), 591–621, nov 2013.

[34] B. Mishra and R. Sepulchre, R3mc: A riemannian three-factor algorithm for low-rank matrix completion, In *53rd IEEE Conference on Decision and Control*, (2014), 1137–1142.

[35] P. Netrapalli, U. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, Non-convex robust pca, In *Advances in Neural Information Processing Systems*, (2014), 1107–1115.

[36] B. Recht, A simpler approach to matrix completion, *Journal of Machine Learning Research*, **12**:Dec (2011), 3413–3430.

[37] B. Recht, M. Fazel, and P.A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, *SIAM review*, **52**:3 (2010), 471–501.

[38] J.D. Rennie and N. Srebro, Fast maximum margin matrix factorization for collaborative prediction, In *Proceedings of the 22nd International Conference on Machine Learning*, (2005), 713–719. ACM.

[39] R. Sun and Z.Q. Luo, Guaranteed matrix completion via non-convex factorization, *IEEE Trans. Inf. Theory*, **62**:11 (2016), 6535–6579.

[40] J. Tanner and K. Wei, Normalized iterative hard thresholding for matrix completion, *SIAM J. Sci. Comput.*, **35**:5 (2013), S104–S125.

[41] K.C. Toh and S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, *Pacific Journal of Optimization*, **6**:15 (2010), 615–640.

[42] B. Vandereycken, Low-rank matrix completion by riemannian optimization, *SIAM J. Optim.*, **23**:2 (2013), 1214–1236.

[43] K. Wei, J.F. Cai, T.F. Chan, and S. Leung, Guarantees of riemannian optimization for low rank matrix completion, *arXiv preprint arXiv:1603.06610*, 2016.

[44] K. Wei, J.F. Cai, T.F. Chan, and S. Leung, Guarantees of riemannian optimization for low rank matrix recovery, *SIAM J. Matrix Anal. Appl.*, **37**:3 (2016), 1198–1222.

[45] Z. Wen, W. Yin, and Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, *Mathematical Programming Computation*, (2012), 1–29.

[46] M. Yan, Y. Yang, and S. Osher, Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit, *J. Sci. Comput.*, **56**:3 (2013), 433–449.

[47] Y. Yang, Y. Feng, and J. Suykens, A nonconvex relaxation approach to robust matrix completion, *Esat.kuleuven.ac.be*.

[48] X. Yi, D. Park, Y. Chen, and C. Caramanis, Fast algorithms for robust pca via gradient descent, In *Advances in neural information processing systems*, (2016), 4152–4160.