


Article

A Unified Proximity Algorithm with Adaptive Penalty for Nuclear Norm Minimization

Wenyu Hu ^{1,*} , Weidong Zheng ¹ and Gaohang Yu ²

¹ College of Mathematics and Computer Science, Gannan Normal University, Ganzhou 341000, China; zhengwd_2012@163.com

² Department of Mathematics, School of Science, Hangzhou Dianzi University, Hangzhou 310018, China; maghyu@163.com

* Correspondence: huwenyu@gnnu.edu.cn; Tel.: +86-0797-839-3663

Received: 9 September 2019; Accepted: 8 October 2019; Published: 11 October 2019



Abstract: The nuclear norm minimization (NNM) problem is to recover a matrix that minimizes the sum of its singular values and satisfies some linear constraints simultaneously. The alternating direction method (ADM) has been used to solve this problem recently. However, the subproblems in ADM are usually not easily solvable when the linear mappings in the constraints are not identities. In this paper, we propose a proximity algorithm with adaptive penalty (PA-AP). First, we formulate the nuclear norm minimization problems into a unified model. To solve this model, we improve the ADM by adding a proximal term to the subproblems that are difficult to solve. An adaptive tactic on the proximity parameters is also put forward for acceleration. By employing subdifferentials and proximity operators, an equivalent fixed-point equation system is constructed, and we use this system to further prove the convergence of the proposed algorithm under certain conditions, e.g., the precondition matrix is symmetric positive definite. Finally, experimental results and comparisons with state-of-the-art methods, e.g., ADM, IADM-CG and IADM-BB, show that the proposed algorithm is effective.

Keywords: nuclear norm minimization; matrix completion; alternating direction method; subdifferential; proximity operator

1. Introduction

The rank minimization (RM) problem aims to recover an unknown low-rank matrix from very limited information. It has gained an increasing amount of attention rapidly in recent years, since it has a range of applications in many computer vision and machine learning areas, such as collaborative filtering [1], subspace segmentation [2], non-rigid structure from motion [3] and image inpainting [4]. This paper deals with the following rank minimization problem:

$$\min_{X \in \mathbb{R}^{m \times n}} \text{rank}(X) \quad \text{s.t. } \mathcal{A}X = \mathbf{b}, \quad (1)$$

where $\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^p$ is a linear map and the vector $\mathbf{b} \in \mathbb{R}^p$ is known. The matrix completion (MC) problem is a special case of the RM problem, where \mathcal{A} is a sampling operator in the form of $\mathcal{A}X = X_\Omega$, $\Omega \subset \{1, 2, \dots, m\} \times \{1, 2, \dots, n\}$ is an index subset, and X_Ω is a vector formed by the entries of X with indices in Ω .

Although Label (1) is simple in form, directly solving it is an NP-hard problem due to the discrete nature of the rank function. One popular way is replacing the rank function with the nuclear norm, which is the sum of the singular values of a matrix. This technique is based on the fact that the nuclear

norm minimization (NNM) is the tightest convex relaxation of the rank minimization problem [5]. The obtained new problem is given by

$$\min_{X \in \mathbb{R}^{m \times n}} \|X\|_* \quad \text{s.t. } \mathcal{A}X = \mathbf{b}, \quad (2)$$

where $\|X\|_* := \sum_{i=1}^r \sigma_i(X)$ denotes the nuclear norm. It has been shown that recovering a low-rank matrix can be achieved by solving Label (2) [1,6].

In practical applications, the observed data may be corrupted with noise, namely $b = \mathcal{A}X + e$, where e contains measurement errors dominated by certain normal distribution. In order to recover the low-rank matrix robustly, problem (2) should be modified to the following inequality constrained problem:

$$\min_{X \in \mathbb{R}^{m \times n}} \|X\|_* \quad \text{s.t. } \|\mathcal{A}X - \mathbf{b}\|_2 \leq \delta, \quad (3)$$

where $\|\cdot\|_2$ is the ℓ_2 norm of vector and the constant $\delta \geq 0$ is the noise level. When $\delta = 0$, problem (3) reduces to the noiseless case (2).

Alternatively, problems (2) and (3) can be rewritten as the nuclear norm regularized least-square (NNRLS) under some conditions:

$$\min_{X \in \mathbb{R}^{m \times n}} \|X\|_* + \frac{\gamma}{2} \|\mathcal{A}X - \mathbf{b}\|_2^2, \quad (4)$$

where $\gamma > 0$ is as given parameter.

The studies on the nuclear norm minimization problem are mainly along two directions. The first one is enhancing the precision of a low rank approximation via replacing the nuclear norm by a non-convex regularizer—for instance, the Schatten p -norm [7,8], the truncated nuclear norm [4,9], the log or fraction function based norm [10,11], and so on. The second one is improving the efficiency of solving problems (2), (3) and (4) and their variants. For instance, the authors in [12] treated the problem (2) as a standard linear semidefinite programming (SDP) problem, and proposed the solving algorithm from the dual problem. However, since the SDP solver uses second-order information, with the increase in the size of the matrix, the memory required to calculate the descending direction quickly becomes too large. Therefore, algorithms that use only first-order information are developed, such as the singular value thresholding (SVT) algorithm [13], the fixed point continuation algorithm (FPCA) [14], the accelerated proximal gradient Lagrangian (APGL) method [15], the proximal point algorithm based on indicator function (PPA-IF) [16], the augmented Lagrange multiplier (ALM) algorithm [17] and the alternating direction methods (ADM) [18–21].

In particular, Chen et al. [18] applied the ADM to solve the nuclear norm based matrix completion problem. Due to the simplicity of the linear mapping \mathcal{A} , i.e., $\mathcal{A}\mathcal{A}^* = \mathcal{I}$, all of the ADM subproblems of the matrix completion problem can be solved exactly by an explicit formula; see [18] for details. Here, and hereafter \mathcal{A}^* and \mathcal{I} represent the adjoint of \mathcal{A} and the identity operator. However, for a generic \mathcal{A} with $\mathcal{A}\mathcal{A}^* \neq \mathcal{I}$, some of the resulting subproblems no longer have closed-form solutions. Thus, the efficiency of the ADM depends heavily on how to solve these harder subproblems.

To solve this difficulty, a common strategy is to introduce new auxiliary variables, e.g., in [19], one auxiliary variable was introduced for solving Label (2), while two auxiliary variables were introduced for Label (3). However, with more variables and more constraints, more memory is required and the convergence of ADM also becomes slower. Moreover, to update auxiliary variables, whose subproblems are least square problems, expensive matrix inversions are often necessary. Even worse, the convergence of ADM with more than two variables is not guaranteed. To mitigate these problems, Yang and Yuan [21] presented a linearized ADM to solve the NNRLS (4) as well as problems (2) and (3), where each subproblems admit explicit solutions. Instead of the linearized technique, Xiao and Jin [19] solve one least square subproblem iteratively by the Barzilai–Borwein (BB) gradient method [22].

Unlike [19], Jin et al. [20] used the linear conjugate gradient (CG) algorithm rather than BB to solve the subproblem.

In this paper, we further investigate the efficiency of ADM in solving the nuclear norm minimization problems. We first reformulate the problems (2), (3) and (4) into a unified form as follows:

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) + g(\mathcal{A}X), \quad (5)$$

where $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$ and $g : \mathbb{R}^p \rightarrow \mathbb{R}$. In this paper, we always fix $f(\cdot) = \|\cdot\|_*$. When considering problems (2) and (3), we choose $g(\cdot) = \mathcal{X}_{B_\delta}(\cdot - \mathbf{b})$, where $\mathcal{X}_{B_\delta}(\cdot)$ denotes the indicator function over $B_\delta := \{\mathbf{u} \in \mathbb{R}^p \mid \|\mathbf{u}\|_2 \leq \delta\}$, i.e.,

$$\mathcal{X}_{B_\delta}(\mathbf{x}) := \begin{cases} 0, & \text{if } \mathbf{x} \in B_\delta, \\ +\infty, & \text{otherwise.} \end{cases} \quad (6)$$

When considering problem (4), we choose $g(\cdot) = \frac{\gamma}{2} \|\cdot - \mathbf{b}\|_2^2$. As a result, for a general linear mapping \mathcal{A} , we only need to solve such a problem whose objective function is a sum of two convex functions and one of them contains an affine transformation.

Motivated by the ADM algorithms above, we then present a unified proximity algorithm with adaptive penalty (PA-AP) to solve Label (5). In particular, we employ the proximity idea to deal with the problems encountered by the present ADM, by adding a proximity term to one of the subproblems. We term the proposed algorithm as a proximity algorithm because we can rewrite it as a fixed-point equation system of proximity operators of f and g . By analyzing the fixed-point equations and applying the ‘‘Condition-M’’ [23], the convergence of the algorithm is proved under some assumptions. Furthermore, to improve the efficiency, an adaptive tactic on the proximity parameters is put forward. This paper is closely related to the works [23–26]. However, this paper is motivated to improve ADM to solve the nuclear norm minimization problem with linear affine constraints.

The organization of this paper is as follows. In Section 2, a review of ADM and its application on NNM are provided. In addition, the properties about subdifferentials and proximity operators are introduced. To improve ADM, a proximity algorithm with adaptive penalty is proposed, and convergence of the proposed algorithm is obtained in Section 3. Section 4 demonstrates the performance and effectiveness of the algorithm through numerical experiment. Finally, we will make a conclusion in Section 5.

2. Preliminaries

In this section, we give a brief review on ADM and its applications to the NNM problem (2) developed in [19,20]. In addition, some preliminaries on subdifferentials and proximity operators are given. Throughout this paper, linear maps are denoted with calligraphic letters (e.g., \mathcal{A}), while capital letters represent matrices (e.g., A), and boldface lowercase letters represent vectors (e.g., \mathbf{x}).

We begin with introducing the ADM. The basic idea of ADM goes back to the work of Gabay and Mercier [27]. ADM is designed to solving the separable convex minimization problem:

$$\min_{\mathbf{x}, \mathbf{y}} \theta_1(\mathbf{x}) + \theta_2(\mathbf{y}), \text{ s.t. } A\mathbf{x} + B\mathbf{y} = \mathbf{c}, \quad (7)$$

where $\theta_1 : \mathbb{R}^s \rightarrow \mathbb{R}$, $\theta_2 : \mathbb{R}^t \rightarrow \mathbb{R}$ are convex functions, and $A \in \mathbb{R}^{l \times s}$, $B \in \mathbb{R}^{l \times t}$ and $\mathbf{c} \in \mathbb{R}^l$. The corresponding augmented Lagrangian function is

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \lambda) = \theta_1(\mathbf{x}) + \theta_2(\mathbf{y}) - \langle \lambda, A\mathbf{x} + B\mathbf{y} - \mathbf{c} \rangle + \frac{\beta}{2} \|A\mathbf{x} + B\mathbf{y} - \mathbf{c}\|_2^2, \quad (8)$$

where $\lambda \in \mathbb{R}^l$ is the Lagrangian multiplier and $\beta > 0$ is a penalty parameter. ADM is to minimize (8) first with respect to \mathbf{x} , then with respect to \mathbf{y} , and finally update λ iteratively, i.e.,

$$\begin{cases} \mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}_k, \lambda_k), \\ \mathbf{y}_{k+1} = \arg \min_{\mathbf{y}} \mathcal{L}(\mathbf{x}_{k+1}, \mathbf{y}, \lambda_k), \\ \lambda_{k+1} = \lambda_k - \beta(\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{y}_{k+1} - \mathbf{c}). \end{cases} \quad (9)$$

The main advantage of ADM is to make use of the separability structure of the objective function $\theta_1(\mathbf{x}) + \theta_2(\mathbf{y})$.

To solve (2) based on ADM, the authors in [19,20] introduced an auxiliary variable Y , and equivalently transformed the original model into

$$\min_{X, Y} \|X\|_* \quad \text{s.t. } X - Y = 0, \mathcal{A}Y = \mathbf{b}. \quad (10)$$

The augmented Lagrangian function to (10) is

$$\mathcal{L}(X, Y, Z, \mathbf{z}) = \|X\|_* - \langle Z, X - Y \rangle + \frac{\mu}{2} \|X - Y\|_F^2 - \langle \mathbf{z}, \mathcal{A}Y - \mathbf{b} \rangle + \frac{\gamma}{2} \|\mathcal{A}Y - \mathbf{b}\|_2^2, \quad (11)$$

where $Z \in \mathbb{R}^{m \times n}$, $\mathbf{z} \in \mathbb{R}^p$ are Lagrangian multipliers, $\mu, \gamma > 0$ are penalty parameters and $\langle \cdot \rangle$ denotes the Frobenius inner product, i.e., the matrices are treated like vectors. Following the idea of ADM, given (X_k, Y_k) , the next pair (X_{k+1}, Y_{k+1}) is determined by alternating minimizing (11),

$$\begin{cases} X_{k+1} = \arg \min_X \mathcal{L}(X, Y_k, Z_k, \mathbf{z}_k), \\ Y_{k+1} = \arg \min_Y \mathcal{L}(X_{k+1}, Y, Z_k, \mathbf{z}_k), \\ Z_{k+1} = Z_k - \mu(X_{k+1} - Y_{k+1}), \\ \mathbf{z}_{k+1} = \mathbf{z}_k - \gamma(\mathcal{A}Y_{k+1} - \mathbf{b}). \end{cases} \quad (12)$$

Firstly, X_{k+1} can be updated by

$$\begin{aligned} X_{k+1} &= \arg \min_X \|X\|_* - \langle Z_k, X - Y_k \rangle + \frac{\mu}{2} \|X - Y_k\|_F^2 \\ &= \arg \min_X \|X\|_* + \frac{\mu}{2} \|X - (Y_k + \frac{1}{\mu} Z_k)\|_F^2, \end{aligned} \quad (13)$$

which in fact corresponds to evaluating the proximal operator of $\|\cdot\|_*$, i.e., $\text{prox}_{\frac{1}{\beta}\|\cdot\|_*}$ which is defined below.

Secondly, given $(X_{k+1}, Z_k, \mathbf{z}_k)$, Y_{k+1} can be updated by

$$\begin{aligned} Y_{k+1} &= \arg \min_Y - \langle Z_k, X_{k+1} - Y \rangle + \frac{\mu}{2} \|X_{k+1} - Y\|_F^2 - \langle \mathbf{z}_k, \mathcal{A}Y - \mathbf{b} \rangle + \frac{\gamma}{2} \|\mathcal{A}Y - \mathbf{b}\|_2^2 \\ &= \arg \min_Y \frac{\mu}{2} \|Y - (X_{k+1} - \frac{1}{\mu} Z_k)\|_F^2 + \frac{\gamma}{2} \|\mathcal{A}Y - (\mathbf{b} + \frac{1}{\gamma} \mathbf{z}_k)\|_2^2, \end{aligned} \quad (14)$$

which is a least square subproblem. Its solution can be found by solving a linear equation:

$$(\mu \mathcal{I} + \gamma \mathcal{A}^* \mathcal{A})Y = \mu X_{k+1} - Z_k - \mathcal{A}^*(\gamma \mathbf{b} + \mathbf{z}_k).$$

However, computing the matrix inverse $(\mu \mathcal{I} + \gamma \mathcal{A}^* \mathcal{A})^{-1}$ is too costly to implement. Though [19,20] adopted inverse-free methods, i.e., BB and CG, to solve (14) iteratively, they are still inefficient, which will be shown in Section 4.

Next, we review definitions of subdifferential and proximity operator, which play an important role in the algorithm and convergence analysis. The subdifferential of a convex function θ at a point $\mathbf{x} \in \mathbb{R}^d$ is the set defined by

$$\partial_{\theta(\cdot)}(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^d : \theta(\mathbf{z}) \geq \theta(\mathbf{x}) + \langle \mathbf{y}, \mathbf{z} - \mathbf{x} \rangle, \forall \mathbf{z} \in \mathbb{R}^d\}. \quad (15)$$

The conjugate function of θ is denoted by θ^* , which is defined by

$$\theta^*(\mathbf{y}) := \sup_{\mathbf{x}} \{\langle \mathbf{x}, \mathbf{y} \rangle - \theta(\mathbf{x})\}.$$

For $\mathbf{x} \in \text{dom}(\theta)$, $\mathbf{y} \in \text{dom}(\theta^*)$, it holds that

$$\mathbf{y} \in \partial_{\theta(\cdot)}(\mathbf{x}) \Leftrightarrow \mathbf{x} \in \partial_{\theta^*(\cdot)}(\mathbf{y}), \quad (16)$$

where $\text{dom}(\cdot)$ denotes the domain of a function.

For a given positive definite matrix \mathcal{H} , the weighted inner product is defined by $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}} = \langle \mathcal{H}\mathbf{x}, \mathbf{y} \rangle$. Furthermore, the proximity operator of θ at \mathbf{x} with respect to \mathcal{H} [23] is defined by

$$\text{prox}_{\theta(\cdot), \mathcal{H}}(\mathbf{x}) = \arg \min \{\theta(\mathbf{u}) + \frac{1}{2} \|\mathbf{u} - \mathbf{x}\|_{\mathcal{H}}^2 : \mathbf{u} \in \mathbb{R}^d\}. \quad (17)$$

If $\mathcal{H} = \beta \mathcal{I}$, then $\text{prox}_{\theta(\cdot), \mathcal{H}}(\cdot)$ is reduced to

$$\text{prox}_{\theta(\cdot), \beta \mathcal{I}}(\mathbf{x}) = \arg \min \{\theta(\mathbf{u}) + \frac{\beta}{2} \|\mathbf{u} - \mathbf{x}\|_2^2 : \mathbf{u} \in \mathbb{R}^d\},$$

and $\text{prox}_{\frac{1}{\beta}\theta(\cdot)}(\cdot)$ is short for $\text{prox}_{\theta(\cdot), \beta \mathcal{I}}(\cdot)$. A relation between subdifferentials and proximity operators is that

$$\mathbf{y} \in \partial_{\theta(\cdot)}(\mathbf{x}) \Leftrightarrow \mathbf{x} = \text{prox}_{\theta(\cdot)}(\mathbf{x} + \mathbf{y}), \quad (18)$$

which is frequently used to construct fixed-point equations and prove convergence of the algorithm.

3. Proximity Algorithm with Adaptive Penalty

In Section 2, it is shown that the current ADM results in expensive matrix inverse computation when solving (2). Therefore, it is desirable to improve it. In this section, we propose a proximity algorithm with adaptive penalty (PA-AP) to solve the unified problem (5).

3.1. Proximity Algorithm

We derive our algorithm from ADM. First of all, we introduce an auxiliary variable $\mathbf{y} \in \mathbb{R}^p$, and convert (5) to

$$\min_{X, \mathbf{y}} f(X) + g(\mathbf{y}), \text{ s.t. } \mathcal{A}X - \mathbf{y} = 0. \quad (19)$$

The augmented Lagrangian function of (19) is defined by

$$\mathcal{L}(X, \mathbf{y}) = f(X) + g(\mathbf{y}) - \langle \lambda, \mathcal{A}X - \mathbf{y} \rangle + \frac{\mu}{2} \|\mathcal{A}X - \mathbf{y}\|_2^2, \quad (20)$$

where $\lambda \in \mathbb{R}^p$ is the Lagrangian multiplier and $\mu > 0$ is a penalty parameter. ADM first updates X by minimizing $\mathcal{L}(X, \mathbf{y})$ with \mathbf{y} being fixed and then updates \mathbf{y} with X fixed at its latest value, until some convergence criteria are satisfied. After some simplification, we get

$$\begin{cases} X_{k+1} = \arg \min_X f(X) + \frac{\mu}{2} \|\mathcal{A}X - (\mathbf{y}_k + \frac{1}{\mu} \lambda_k)\|_2^2, \\ \mathbf{y}_{k+1} = \arg \min_{\mathbf{y}} g(\mathbf{y}) + \frac{\mu}{2} \|\mathbf{y} - (\mathcal{A}X_{k+1} - \frac{1}{\mu} \lambda_k)\|_2^2, \\ \lambda_{k+1} = \lambda_k - \mu(\mathcal{A}X_{k+1} - \mathbf{y}_{k+1}). \end{cases} \quad (21)$$

Note that the subproblem of X_{k+1} usually has no closed-form solutions when \mathcal{A} is not the identity. In order to design efficient algorithms, we add a proximity term to this subproblem. More precisely, we propose the following algorithm:

$$\begin{cases} X_{k+1} = \arg \min_X f(X) + \frac{\mu}{2} \|\mathcal{A}X - (\mathbf{y}_k + \frac{1}{\mu} \lambda_k)\|_2^2 + \frac{1}{2} \|X - X_k\|_Q^2, \\ \mathbf{y}_{k+1} = \arg \min_{\mathbf{y}} g(\mathbf{y}) + \frac{\mu}{2} \|\mathbf{y} - (\mathcal{A}X_{k+1} - \frac{1}{\mu} \lambda_k)\|_2^2, \\ \lambda_{k+1} = \lambda_k - \mu(\mathcal{A}X_{k+1} - \mathbf{y}_{k+1}), \end{cases} \quad (22)$$

where $Q \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix.

The next lemma shows that (22) is equivalent to a fixed-point equation of some proximity operators. The proof is similar to [25].

Lemma 1. *The problem (22) is equivalent to solving the following equations:*

$$\begin{cases} \mathbf{z}_{k+1} = \text{prox}_{\mu g^*(\cdot)}(\mu \mathcal{A}X_k + \mathbf{z}_k), \\ X_{k+1} = \text{prox}_{\tau f(\cdot)}(X_{k+1} - \tau(\mu \mathcal{A}^* \mathcal{A} + Q)(X_{k+1} - X_k) - \tau \mathcal{A}^*(2\mathbf{z}_{k+1} - \mathbf{z}_k)), \end{cases} \quad (23)$$

where $\tau > 0$ is arbitrary.

Proof. By changing the order of iterations, the first-order optimality condition of (22) is

$$\begin{cases} 0 \in \partial_{g(\cdot)}(\mathbf{y}_{k+1}) + \mu(\mathbf{y}_{k+1} - \mathcal{A}X_k + \frac{1}{\mu} \lambda_k), \\ \lambda_{k+1} = \lambda_k - \mu(\mathcal{A}X_k - \mathbf{y}_{k+1}), \\ 0 \in \partial_{f(\cdot)}(X_{k+1}) + \mu \mathcal{A}^*(\mathcal{A}X_{k+1} - \mathbf{y}_{k+1} - \frac{1}{\mu} \lambda_{k+1}) + Q(X_{k+1} - X_k). \end{cases} \quad (24)$$

From the first line of (24) and (16), we obtain

$$\mu \mathbf{y}_{k+1} \in \partial_{\mu g^*(\cdot)}(\mu(\mathcal{A}X_k - \mathbf{y}_{k+1}) - \lambda_k). \quad (25)$$

Denote $\mathbf{z}_k := -\lambda_k$. Then,

$$\mathbf{z}_{k+1} = \mu \mathcal{A}X_k + \mathbf{z}_k - \mu \mathbf{y}_{k+1}. \quad (26)$$

Using (18), it follows from (25) and (26) that

$$\mathbf{z}_{k+1} = \text{prox}_{\mu g^*(\cdot)}(\mu \mathcal{A}X_k + \mathbf{z}_k). \quad (27)$$

By (26), we have

$$\mu \mathcal{A}X_k + \mathbf{z}_k = \mu \mathbf{y}_{k+1} + \mu(\mathcal{A}X_k - \mathbf{y}_{k+1} - \frac{1}{\mu} \lambda_k) = \mu \mathbf{y}_{k+1} + \mathbf{z}_{k+1},$$

and thus

$$\mu \mathcal{A}^*(\mathcal{A}X_{k+1} - \mathbf{y}_{k+1} - \frac{1}{\mu}\lambda_{k+1}) + Q(X_{k+1} - X_k) = (\mu \mathcal{A}^* \mathcal{A} + Q)(X_{k+1} - X_k) + \mathcal{A}^*(2\mathbf{z}_{k+1} - \mathbf{z}_k). \quad (28)$$

From (28) and the third line of (24), given $\tau > 0$, we have

$$-\tau(\mu \mathcal{A}^* \mathcal{A} + Q)(X_{k+1} - X_k) - \tau \mathcal{A}^*(2\mathbf{z}_{k+1} - \mathbf{z}_k) \in \partial_{\tau f(\cdot)}(X_{k+1}),$$

which yields

$$X_{k+1} = \text{prox}_{\tau f(\cdot)}(X_{k+1} - \tau(\mu \mathcal{A}^* \mathcal{A} + Q)(X_{k+1} - X_k) - \tau \mathcal{A}^*(2\mathbf{z}_{k+1} - \mathbf{z}_k)). \quad (29)$$

Therefore, the results in (23) are achieved by combining (27) and (29). \square

We now discuss the choice of Q . To make the first subproblem of (22) have closed-form solutions, in this paper, we simply choose $Q = \mu\eta\mathcal{I} - \mu\mathcal{A}^*\mathcal{A}$. To make sure Q is positive definite, η must satisfy $\eta > \|\mathcal{A}\|_2^2$, where $\|\cdot\|_2$ is the spectral norm. By substituting Q into (22), we obtain

$$\begin{cases} X_{k+1} = \arg \min_X f(X) + \frac{\mu\eta}{2} \|X - (X_k + \frac{\mathcal{A}^*(\lambda_k - \mu(\mathcal{A}X_k - \mathbf{y}_k))}{\mu\eta})\|_2^2, \\ \mathbf{y}_{k+1} = \arg \min_{\mathbf{y}} g(\mathbf{y}) + \frac{\mu}{2} \|\mathbf{y} - (\mathcal{A}X_{k+1} - \frac{1}{\mu}\lambda_k)\|_2^2, \\ \lambda_{k+1} = \lambda_k - \mu(\mathcal{A}X_{k+1} - \mathbf{y}_{k+1}). \end{cases} \quad (30)$$

The subproblems in (30) can be solved explicitly based on proximity operators. Specifically, we have

$$X_{k+1} = \text{prox}_{\frac{1}{\mu\eta}\|\cdot\|_*}(X_k + \frac{\mathcal{A}^*(\lambda_k - \mu(\mathcal{A}X_k - \mathbf{y}_k))}{\mu\eta}) = U \max\{\Sigma - \frac{1}{\mu\eta}, 0\} V^T, \quad (31)$$

where $U\Sigma V^T$ is the singular value decomposition (SVD) of $X_k + \frac{\mathcal{A}^*(\lambda_k - \mu(\mathcal{A}X_k - \mathbf{y}_k))}{\mu\eta}$, $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing the singular values.

Moreover, $\mathbf{y}_{k+1} = \text{prox}_{\frac{1}{\mu}g(\cdot)}(\mathcal{A}X_{k+1} - \frac{1}{\mu}\lambda_k)$, which depends on the choice of $g(\cdot)$. If $g(\cdot) = \mathcal{X}_{B_\delta}(\cdot - \mathbf{b})$, then

$$\text{prox}_{\frac{1}{\mu}g(\cdot)}(\mathbf{x}) = \begin{cases} \mathbf{x}, & \mathbf{x} - \mathbf{b} \in B_\delta, \\ \mathbf{b} + \frac{\delta}{\|\mathbf{x} - \mathbf{b}\|_2}(\mathbf{x} - \mathbf{b}), & \text{otherwise.} \end{cases} \quad (32)$$

If $g(\mathbf{x}) = \frac{\gamma}{2}\|\mathbf{x} - \mathbf{b}\|_2^2$, then

$$\text{prox}_{\frac{1}{\mu}g(\cdot)}(\mathbf{x}) = \frac{\gamma\mathbf{b} + \mu\mathbf{x}}{\gamma + \mu}. \quad (33)$$

3.2. Adaptive Penalty

In previous proximity algorithms [23,28,29], the penalty parameter μ is usually fixed. In view of the linearized ADM, Liu et al. [26] presented an adaptive updating strategy for the penalty parameter μ . Motivated by it, we update μ by

$$\mu_{k+1} = \min\{\mu_{max}, \rho\mu_k\}, \quad (34)$$

where μ_{max} is an upper bound of $\{\mu_k\}$. The value of ρ is defined as

$$\rho = \begin{cases} \rho_0, & \text{if } \mu_k \max\{\sqrt{\eta}\|X_{k+1} - X_k\|_F, \|\mathbf{y}_{k+1} - \mathbf{y}_k\|_2\} / \|\mathbf{b}\|_2 \leq \epsilon_1, \\ 1, & \text{otherwise,} \end{cases}$$

where $\rho_0 > 1$ and $\epsilon_1 > 0$ are given.

Based on the above analysis in Sections 3.1 and 3.2, the proximity algorithm with adaptive penalty (abbr. PA-AP) for solving (5) can be outlined in Algorithm 1.

Algorithm 1 PA-AP for solving (5)

Input: Observation vector \mathbf{b} , linear mapping \mathcal{A} , and some parameters $\mu_0, \rho_0, \epsilon_1, \epsilon_2 > 0, \mu_{max} \gg \mu_0 > 0$.
Initialize: Set X_0 and \mathbf{y}_0 to zero matrix and vector, respectively. Set $\mu = \mu_0$ and $\eta > \|\mathcal{A}\|_2^2$. Set $k = 0$.
while not converged, do
 step 1: Update X_{k+1} , \mathbf{y}_{k+1} and λ_{k+1} in turn by (30).
 step 2: Update μ_{k+1} by (34), and let $\mu \leftarrow \mu_{k+1}$.
 step 3: $k \leftarrow k + 1$.
end while

3.3. Convergence

In this section, we establish the convergence of (22). For problem (5), Li et al. [23] presented a general formula of fixed-point algorithms. Given two symmetric positive definite matrices $S \in \mathbb{R}^{p \times p}$ and $T \in \mathbb{R}^{m \times m}$, denote

$$S_{\mathcal{A}} := \begin{pmatrix} 0 & \mathcal{A} \\ -\mathcal{A}^* & 0 \end{pmatrix}, R := \begin{pmatrix} S & 0 \\ 0 & T \end{pmatrix}, E := \mathcal{I} + R^{-1}S_{\mathcal{A}},$$

where $S_{\mathcal{A}}, R$ and E can be treated as linear maps which map $(\mathbb{R}^p) \times (\mathbb{R}^{m \times n})$ into itself.

Defining $Z := (\mathbf{z}, X) \in (\mathbb{R}^p) \times (\mathbb{R}^{m \times n})$ and $\mathcal{T} := \text{prox}_{(g^*+f)(\cdot), R}$, then the solution to (5) is equivalent to

$$Z = (\mathcal{T} \circ E)(Z).$$

Furthermore, a multi-step proximity algorithm was proposed, which is

$$Z_{k+1} = \mathcal{T}(E_0 Z_{k+1} + R^{-1} \sum_{i=1}^l M_i Z_{k-i+1}), \quad (35)$$

where $E_0 = E - R^{-1}(S_{\mathcal{A}} - M_0)$ and $M_0 = \sum_{i=1}^l M_i$. Let $\mathcal{M} := \{M_i : i = 0, 1, \dots, l\}$. Li et al. [23] proved that the sequence $\{Z_k\}$ generated by (35) converges to a solution of problem (5) if \mathcal{M} satisfies the ‘‘Condition-M’’, which refers to

- (i) $M_0 = \sum_{i=1}^l M_i$,
- (ii) $M_1 = M_2 = \dots = M_{l-1}$,
- (iii) $H := M_0 + M_l$, H is symmetric positive definite,
- (iv) $\mathcal{N}(H) \subseteq \mathcal{N}(M_l) \cap \mathcal{N}(M_l^T)$,
- (v) $\|(H^\dagger)^{\frac{1}{2}} M_l (H^\dagger)^{\frac{1}{2}}\|_2 < \frac{1}{2}$,

where $\mathcal{N}(H)$ and H^\dagger are the null space and the Moore–Penrose pseudo-inverse matrix of H , respectively.

By checking the Condition-M, we prove the convergence of (22).

Theorem 1. *If Q is symmetric positive definite, and $\|\mathcal{A}(\mathcal{A}^* \mathcal{A} + \frac{1}{\mu} Q)^{-\frac{1}{2}}\|_2 < 1$, then the sequence generated by (22) converges to the solution of (5).*

Proof. By comparing (23) and (35), it can be found that $l = 2$ and

$$E_0 = \begin{pmatrix} 0 & 0 \\ -2\tau \mathcal{A}^* & \mathcal{I} - \tau(\mu \mathcal{A}^* \mathcal{A} + Q) \end{pmatrix}, R = \begin{pmatrix} \frac{1}{\mu} \mathcal{I} & 0 \\ 0 & \frac{1}{\tau} \mathcal{I} \end{pmatrix}, M_0 = M_1 = \begin{pmatrix} \frac{1}{\mu} \mathcal{I} & \mathcal{A} \\ \mathcal{A}^* & \mu \mathcal{A}^* \mathcal{A} + Q \end{pmatrix}, M_2 = 0.$$

We clearly see that Item (i) and Item (ii) of Condition-M hold. Furthermore, $H = M_0 + M_2 = M_0$, which is symmetric. By Lemma 6.2 in [23], H is positive definite if and only if $\|(\mu\mathcal{A}^*\mathcal{A} + Q)^{-\frac{1}{2}}\mathcal{A}^*(\frac{1}{\mu})^{-\frac{1}{2}}\|_2 < 1$, which is equivalent to $\|\mathcal{A}(\mathcal{A}^*\mathcal{A} + \frac{1}{\mu}Q)^{-\frac{1}{2}}\|_2 < 1$. Hence, Item (iii) of Condition-M also holds.

Since H is positive definite, it yields that $\mathcal{N}(H) = \{0\}$, which implies Item (iv) of Condition-M holds. Finally, $M_2 = 0$ implies that Item (v) holds. Consequently, the sequence generated by (23) converges to the solution of (5). The equivalence of (22) and (23) proves the result. \square

Corollary 1. Let $Q = \mu\eta\mathcal{I} - \mu\mathcal{A}^*\mathcal{A}$. If $\eta > \|\mathcal{A}\|_2^2$, then the sequence generated by (30) converges to the solution of (5).

Proof. Since $\eta > \|\mathcal{A}\|_2^2$ if and only if $\|\mathcal{A}(\mathcal{A}^*\mathcal{A} + \frac{1}{\mu}Q)^{-\frac{1}{2}}\|_2 < 1$, the result is true by Theorem 1. \square

4. Numerical Experiments

In this section, we present some numerical experiments to show the effectiveness of the proposed algorithm (PA-AP). To this end, we test algorithms to solve the nuclear norm minimization problem, the noiseless matrix completion problem ($\delta = 0$), noisy matrix completion ($\delta > 0$) and low-rank image recovery problem. We compare PA-AP against the ADM [18], IADM-CG [20] and IADM-BB [19]. All experiments are performed under Windows 10 and MATLAB R2016 running on a Lenovo laptop with an Intel CORE i7 CPU at 2.7 GHz and 8 GB of memory. In the numerical experiments of the first two parts, we use randomly generated square matrices for simulations. We denote the true solution by $X^* \in \mathbb{R}^{m \times m}$. We generate the rank- r matrix X^* as a product of $X_L X_R^T$, where X_L and X_R are independent $m \times r$ matrices with i.i.d. Gaussian entries. For each test, the stopping criterion is

$$\text{RelChg} = \frac{\|X_k - X_{k-1}\|_F}{\|X_{k-1}\|_F} \leq \varepsilon_2,$$

where $\varepsilon_2 > 0$. The algorithms are also forced to stop when the iteration number exceeds 10^3 .

Let \hat{X} be the solution obtained by the algorithms. We use the relative error to measure the quality of \hat{X} compared to the original matrix X^* , i.e.,

$$\text{RelErr} = \frac{\|\hat{X} - X^*\|_F}{\|X^*\|_F}.$$

It is obvious that, in each iteration of computing X^{k+1} , PA-AP contains an SVD computation that computes all singular values and singular vectors. However, we actually only need the ones that are bigger than $\frac{1}{\mu\eta}$. This causes the main computational load by using full SVD. Fortunately, this disadvantage can be smoothed by using the software PROPACK [30], which is designed to compute the singular values bigger than a threshold and the corresponding vectors. Although PROPACK can calculate the first fixed number of singular values, it cannot automatically determine the number of singular values greater than $\frac{1}{\mu\eta}$. Therefore, in order to perform a local SVD, we need to predict the number of singular values and vectors calculated in each iteration, which is expressed by sv_k . We initialize $sv_0 = 0.01m$, and update it in each iteration as follows:

$$sv_{k+1} = \begin{cases} svp_k + 1, & \text{if } svp_k < sv_k, \\ svp_k + 5, & \text{if } svp_k = sv_k, \end{cases}$$

where svp_k is the number of singular values in the sv_k singular values that are bigger than $\frac{1}{\mu\eta}$.

We use r and p to represent the rank of an $(m \times n)$ matrix and the cardinality of the index set Ω , i.e., $p = |\Omega|$, and use $sr = p/(mn)$ to represent the sampling rate. The “degree of freedom” of a matrix with rank r is defined by $dof = r(m + n - r)$. For PA-AP, we set $\varepsilon_1 = 10^{-4}$, $\mu_0 = \frac{1}{\|\mathbf{b}\|_2}$,

$\mu_{max} = \max\{10^3 \mu_0, 10^{-2}\}$, $\rho_0 = 1.7$, and $\eta = 1.01 \|\mathcal{A}\|_2^2$. In all the experimental results, the boldface numbers always indicate the best results.

4.1. Nuclear Norm Minimization Problem

In this subsection, we use PA-AP to solve the three types of problems including (2)–(4). The linear map \mathcal{A} is chosen as a partial discrete cosine transform (DCT) matrix. Specifically, in the noiseless model (2), \mathcal{A} is generated by the following MATLAB scripts:

$$indices = randsample(m * n, p); b = dct2(X^*); b = b(indices),$$

which shows that \mathcal{A} maps $\mathbb{R}^{m \times n}$ into \mathbb{R}^p . In the noise model (3), we further set

$$\mathbf{b} = \mathcal{A}X^* + \omega,$$

where ω is the additive Gaussian noise of zero mean and standard deviation σ . In (3), the noise level δ is chosen as $\|\omega\|_2$.

The results are listed in Table 1, where the number of iterations (Iter) and CPU time in seconds (Time) besides RelErr are reported. To further illustrate the efficiency of PA-AP, we test problems with different matrix sizes and sampling rates (*sr*). In Table 2, we compare the PA-AP with IADM-CG and IADM-BB for solving the NNRLS problem (4). It shows that our method is more efficient than the other two methods, and thus it is suitable for solving large-scale problems.

Table 1. PA-AP for noiseless and noisy DCT matrix ($m = n, \varepsilon_2 = 10^{-4}$).

(n, r)	p/dof	sr	Prob. (3) ($\delta = 0$)			Prob. (3) ($\delta = 10^{-2}$)			Prob. (4)		
			Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr
(128, 3)	4.32	0.2	86	2.03	3.942×10^{-3}	83	1.83	5.706×10^{-3}	83	2.14	8.050×10^{-3}
(128, 3)	8.64	0.4	31	0.57	4.042×10^{-4}	33	0.69	5.535×10^{-3}	34	0.70	6.196×10^{-3}
(128, 3)	12.95	0.6	20	0.44	1.376×10^{-4}	20	0.45	5.652×10^{-3}	20	0.48	5.714×10^{-3}
(128, 3)	17.27	0.8	13	0.29	6.251×10^{-5}	12	0.34	5.952×10^{-3}	13	0.28	5.966×10^{-3}
(256, 5)	5.17	0.2	53	2.04	3.178×10^{-4}	53	2.02	3.846×10^{-3}	53	2.08	4.353×10^{-3}
(256, 5)	10.34	0.4	31	1.22	1.649×10^{-4}	31	1.16	4.424×10^{-3}	31	1.28	4.404×10^{-3}
(256, 5)	15.51	0.6	20	0.78	1.139×10^{-4}	20	0.78	4.414×10^{-3}	20	0.76	4.419×10^{-3}
(256, 5)	20.68	0.8	13	0.50	3.893×10^{-5}	13	0.52	4.458×10^{-3}	14	0.52	4.297×10^{-3}
(512, 10)	5.17	0.2	55	7.63	2.397×10^{-4}	55	7.77	2.858×10^{-3}	54	7.86	2.939×10^{-3}
(512, 10)	10.34	0.4	34	4.22	1.259×10^{-4}	34	4.55	3.068×10^{-3}	34	4.46	3.081×10^{-3}
(512, 10)	15.51	0.6	22	2.85	1.195×10^{-4}	22	2.98	3.106×10^{-3}	22	2.96	3.177×10^{-3}
(512, 10)	20.68	0.8	13	1.81	1.009×10^{-4}	13	1.85	3.129×10^{-3}	13	1.86	3.262×10^{-3}

Table 2. Comparisons of PA-AP, IADM-CG and IADM-BB for DCT matrix ($m = n, \varepsilon_2 = 10^{-4}$).

(n, r)	p/dof	sr	PA-AP			IADM-CG			IADM-BB		
			Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr
(500, 10)	5.05	0.2	54	8.65	2.881×10^{-3}	33	15.09	8.312×10^{-3}	39	26.65	9.448×10^{-3}
(500, 10)	10.10	0.4	33	4.14	3.035×10^{-3}	23	8.45	4.433×10^{-3}	26	16.58	6.484×10^{-3}
(500, 10)	15.15	0.6	22	2.15	3.20×10^{-3}	15	5.56	3.968×10^{-3}	18	9.81	4.581×10^{-3}
(500, 10)	20.20	0.8	13	1.72	3.224×10^{-3}	10	4.79	3.292×10^{-3}	12	7.50	3.528×10^{-3}
(1000, 20)	5.05	0.2	70	38.20	2.110×10^{-3}	33	69.23	8.806×10^{-3}	40	141.57	5.047×10^{-3}
(1000, 20)	10.10	0.4	38	15.14	2.379×10^{-3}	22	41.34	3.442×10^{-3}	23	78.78	7.097×10^{-3}
(1000, 20)	15.15	0.6	21	8.83	2.322×10^{-3}	17	30.52	3.362×10^{-3}	19	60.35	4.277×10^{-3}
(1000, 20)	20.20	0.8	14	6.60	2.379×10^{-3}	13	25.36	3.665×10^{-3}	20	57.51	2.724×10^{-3}
(2000, 20)	5.05	0.2	74	144.89	2.270×10^{-3}	39	581.64	1.039×10^{-2}	46	1399.31	8.316×10^{-3}
(2000, 20)	10.10	0.4	33	62.68	2.348×10^{-3}	22	328.02	4.755×10^{-3}	22	628.66	5.090×10^{-3}
(2000, 20)	15.15	0.6	21	34.60	2.368×10^{-3}	14	172.70	3.405×10^{-3}	17	438.01	4.527×10^{-3}
(2000, 20)	20.20	0.8	13	23.40	2.401×10^{-3}	20	128.73	2.752×10^{-3}	15	359.59	3.260×10^{-3}

4.2. Matrix Completion

This subsection adopts the PA-AP method to solve the noiseless matrix completion problem (2) and the noisy matrix completion problem (3) to verify its validity. The mapping \mathcal{A} is a linear projection operator defined as $\mathcal{A}X^* = X_\Omega$, where X_Ω is a vector formed by the components of X^* with indices in Ω . The indicators of the selected elements are randomly arranged to form a column vector, and the index set of the first $sr \times m \times n$ is selected to form the set Ω . For noisy matrix completion problems, we take $\delta = 10^{-2}$.

In Table 3, we report the numerical results of PA-AP for noiseless and noisy matrix completion problems, taking $m = n = 1000$ and $m = n = 2000$. Only the rank of the original matrix is considered to be $r = 10$ and $r = 20$. As can be seen from Table 3, the PA-AP method can effectively solve these problems. Compared with the noiseless problem, PA-AP solves the noisy problems accuracy of the solution dropped. Moreover, the number of iterations and the running time decrease as sr increases.

Table 3. PA-AP for noiseless and noisy matrix completion problems ($m = n, \varepsilon_2 = 10^{-5}$).

(n, r)	p/dof	sr	Prob. (3) ($\delta = 0$)			Prob. (3) ($\delta = 10^{-2}$)			Prob. (4)		
			Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr
(1000, 10)	10.05	0.2	76	21.94	2.439×10^{-5}	76	21.52	3.015×10^{-3}	76	22.92	5.245×10^{-4}
(1000, 10)	20.10	0.4	40	9.28	1.159×10^{-5}	40	9.68	3.085×10^{-3}	39	9.55	7.436×10^{-4}
(1000, 10)	30.15	0.6	22	5.72	1.061×10^{-6}	23	6.36	3.105×10^{-3}	23	6.10	8.251×10^{-4}
(1000, 10)	40.20	0.8	12	3.88	2.859×10^{-6}	14	3.76	3.075×10^{-3}	14	3.74	8.668×10^{-4}
(1000, 20)	5.05	0.2	87	39.10	3.037×10^{-5}	88	43.58	1.995×10^{-3}	87	43.71	5.469×10^{-4}
(1000, 20)	10.10	0.4	46	13.02	1.277×10^{-5}	46	13.69	2.115×10^{-3}	46	13.47	7.467×10^{-4}
(1000, 20)	15.15	0.6	25	7.56	1.230×10^{-5}	25	7.54	2.158×10^{-3}	25	7.64	8.232×10^{-4}
(1000, 20)	20.20	0.8	14	4.86	9.617×10^{-6}	14	4.87	2.194×10^{-3}	14	4.88	8.624×10^{-4}
(2000, 10)	20.05	0.2	80	46.73	2.538×10^{-5}	80	67.92	3.086×10^{-3}	80	54.29	7.422×10^{-4}
(2000, 10)	40.10	0.4	37	26.02	1.299×10^{-5}	37	31.65	3.091×10^{-3}	37	27.20	8.641×10^{-4}
(2000, 10)	60.15	0.6	23	19.30	9.672×10^{-6}	23	21.64	3.160×10^{-3}	23	20.02	9.024×10^{-4}
(2000, 10)	80.20	0.8	13	11.99	2.584×10^{-6}	13	13.31	3.153×10^{-3}	13	11.71	9.276×10^{-4}
(2000, 20)	10.05	0.2	91	108.16	2.853×10^{-5}	91	132.04	2.133×10^{-3}	91	129.30	7.461×10^{-4}
(2000, 20)	20.10	0.4	40	42.54	1.146×10^{-5}	40	44.62	2.190×10^{-3}	40	42.49	8.674×10^{-4}
(2000, 20)	30.15	0.6	24	27.07	8.683×10^{-6}	24	28.24	2.226×10^{-3}	24	27.43	9.058×10^{-4}
(2000, 20)	40.20	0.8	13	15.81	3.772×10^{-6}	13	17.19	2.205×10^{-3}	13	16.18	9.268×10^{-4}

To further verify the validity of the PA-AP method, it is compared with ADM, IADM-CG and IADM-BB. When RelChg is lower than 10^{-5} , the algorithms are set to terminate. The numerical results of the four methods for solving the noiseless and noisy MC problem are recorded in Tables 4 and 5, from which we can see that the calculation time of the PA-AP method is much less than IADM-BB and IADM-CG, and the number of iterations and calculation time of PA-AP and ADM are almost the same, while our method is relatively more accurate. From the limited experimental data, the PA-AP method is shown to be more effective than the ADM, IADM-BB and IADM-CG.

Table 4. Comparisons of PA-AP, ADM, IADM-CG and IADM-BB for noiseless matrix completion.

(n, r)	p	$pldof$	sr	PA-AP			ADM			IADM-CG			IADM-BB		
				Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr
(1024, 5)	209235	10.29	0.2	80	17.52	2.218×10^{-5}	81	18.74	6.133×10^{-4}	52	58.41	3.266×10^{-3}	50	112.50	1.801×10^{-3}
(1024, 5)	314529	30.80	0.3	50	10.25	1.743×10^{-5}	56	11.60	3.988×10^{-4}	37	39.52	3.639×10^{-3}	51	96.33	3.023×10^{-3}
(1024, 5)	419547	41.06	0.4	37	7.57	1.036×10^{-5}	40	8.31	3.023×10^{-4}	36	36.77	2.439×10^{-3}	37	67.10	2.655×10^{-3}
(1024, 5)	525213	51.33	0.5	28	6.44	8.484×10^{-6}	31	7.08	2.600×10^{-4}	31	33.22	1.102×10^{-3}	29	57.78	1.414×10^{-3}
(1024, 5)	628736	61.59	0.6	22	4.89	6.311×10^{-6}	26	6.01	2.063×10^{-4}	21	22.91	2.330×10^{-3}	32	55.61	1.020×10^{-3}
(1024, 5)	733429	71.86	0.7	17	4.20	4.239×10^{-6}	21	5.42	1.738×10^{-4}	40	40.23	1.361×10^{-4}	53	49.85	1.743×10^{-4}
(1024, 5)	838513	82.12	0.8	13	3.17	3.840×10^{-6}	16	3.95	1.593×10^{-4}	41	29.95	4.171×10^{-4}	30	41.92	1.031×10^{-4}
(1024, 5)	943801	92.39	0.9	13	2.58	2.782×10^{-6}	12	2.58	1.386×10^{-4}	11	12.16	8.341×10^{-4}	12	19.25	6.523×10^{-4}
(1024, 10)	209469	10.29	0.2	77	15.94	2.341×10^{-5}	71	14.79	6.273×10^{-4}	51	50.90	2.646×10^{-3}	53	109.16	2.210×10^{-3}
(1024, 10)	315457	15.44	0.3	55	11.95	1.848×10^{-5}	52	11.63	4.264×10^{-4}	40	42.72	2.328×10^{-3}	39	79.82	2.258×10^{-3}
(1024, 10)	419442	20.58	0.4	40	10.66	1.005×10^{-5}	39	10.79	3.031×10^{-4}	32	33.68	1.268×10^{-3}	34	68.76	8.858×10^{-3}
(1024, 10)	524145	25.73	0.5	30	8.97	8.688×10^{-5}	31	9.49	2.525×10^{-4}	32	35.26	1.137×10^{-3}	29	55.39	1.536×10^{-3}
(1024, 10)	629555	30.87	0.6	23	5.26	7.085×10^{-6}	24	5.56	1.972×10^{-4}	39	37.55	4.376×10^{-4}	32	51.02	7.468×10^{-3}
(1024, 10)	733285	36.02	0.7	18	4.98	6.149×10^{-6}	19	5.13	1.817×10^{-4}	27	28.35	6.503×10^{-4}	30	48.98	3.295×10^{-4}
(1024, 10)	838650	41.16	0.8	14	4.53	2.647×10^{-6}	16	5.28	1.557×10^{-4}	31	33.23	5.794×10^{-4}	17	30.96	6.046×10^{-4}
(1024, 10)	943738	46.31	0.9	12	3.36	2.130×10^{-6}	12	3.62	1.370×10^{-4}	34	34.15	8.934×10^{-4}	23	40.62	3.884×10^{-4}

Table 5. Comparisons of PA-AP, ADM, IADM-CG and IADM-BB for noisy matrix completion ($\delta = 10^{-2}$).

(n, r)	p	$pldof$	sr	PA-AP			ADM			IADM-CG			IADM-BB		
				Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr	Iter	Time	RelErr
(1024, 5)	210151	20.53	0.2	61	16.96	4.345×10^{-3}	60	13.61	4.354×10^{-3}	40	95.66	9.546×10^{-3}	45	237.45	5.234×10^{-3}
(1024, 5)	314332	30.80	0.3	43	10.68	4.413×10^{-3}	43	10.88	4.448×10^{-3}	27	64.23	1.215×10^{-2}	31	164.52	5.319×10^{-3}
(1024, 5)	418708	41.06	0.4	31	8.95	4.416×10^{-3}	33	9.50	4.419×10^{-3}	25	59.75	6.410×10^{-3}	24	123.38	5.318×10^{-3}
(1024, 5)	524429	51.33	0.5	24	7.65	4.387×10^{-3}	25	7.74	4.402×10^{-3}	21	49.00	4.920×10^{-3}	19	97.81	4.791×10^{-3}
(1024, 5)	628736	61.59	0.6	19	5.25	4.452×10^{-3}	19	4.93	4.452×10^{-3}	19	40.69	5.035×10^{-3}	14	73.53	5.177×10^{-3}
(1024, 5)	734131	71.86	0.7	15	4.47	4.335×10^{-3}	17	4.56	4.337×10^{-3}	17	39.24	4.791×10^{-3}	21	76.87	4.524×10^{-3}
(1024, 5)	838476	82.12	0.8	12	3.64	4.444×10^{-3}	14	3.97	4.446×10^{-3}	19	39.52	1.184×10^{-3}	15	54.21	4.479×10^{-3}
(1024, 5)	944170	92.39	0.9	10	3.16	4.486×10^{-3}	11	2.94	4.557×10^{-3}	10	21.29	4.591×10^{-3}	10	36.21	4.531×10^{-3}
(1024, 10)	210118	10.29	0.2	77	20.56	3.017×10^{-3}	71	19.10	3.081×10^{-3}	53	62.98	3.894×10^{-3}	48	123.01	4.089×10^{-3}
(1024, 10)	314614	15.44	0.3	54	15.21	3.089×10^{-3}	52	15.27	3.119×10^{-3}	40	44.45	3.842×10^{-3}	41	85.27	3.816×10^{-3}
(1024, 10)	420191	20.58	0.4	40	11.53	3.060×10^{-3}	39	11.01	3.075×10^{-3}	32	36.46	3.306×10^{-3}	34	72.74	3.178×10^{-3}
(1024, 10)	523405	25.73	0.5	30	7.59	3.087×10^{-3}	31	7.70	3.097×10^{-3}	52	52.59	3.087×10^{-3}	28	54.28	3.647×10^{-3}
(1024, 10)	628935	30.87	0.6	23	6.04	3.061×10^{-3}	24	6.76	3.068×10^{-3}	44	45.74	3.061×10^{-3}	39	61.59	3.113×10^{-3}
(1024, 10)	734096	36.02	0.7	18	5.17	3.090×10^{-3}	19	4.97	3.095×10^{-3}	32	33.06	3.134×10^{-3}	47	72.07	3.116×10^{-3}
(1024, 10)	838509	41.16	0.8	14	4.26	3.121×10^{-3}	16	4.65	3.125×10^{-3}	31	31.85	3.175×10^{-3}	38	60.78	3.183×10^{-3}
(1024, 10)	944068	46.31	0.9	12	3.76	3.093×10^{-3}	12	3.54	3.096×10^{-3}	34	34.64	3.216×10^{-3}	33	45.15	3.095×10^{-3}

4.3. Low-Rank Image Recovery

In the section, we turn to solve problem (2) for low-rank image recovery. The effectiveness of the PA-AP method is verified by testing three 512×512 grayscale images. First, the original images are transformed into low-rank images with rank 40. Then, we lose some elements from the low rank matrix to get the damaged image, and restore them by using the PA-AP, ADM, IADM-BB and IADM-CG, respectively. The iteration process is stopped when RelChg falls below 10^{-5} . The original images, the corresponding low-rank images, the damaged images, and the restored images by the PA-AP are depicted in Figure 1. Observing the figure, we clearly see that our algorithm performs well.



Figure 1. Original 512×512 images (Lena, Pirate, Cameraman) with full rank (first column); Corresponding low rank images with $r = 40$ (second column); Randomly masked images from rank 40 images with $sr = 40\%$ (third column); Recovered images by PA-AP (last column).

To evaluate the recovery performance, we employ the Peak Signal-to-Noise Ratio (PSNR), which is defined as

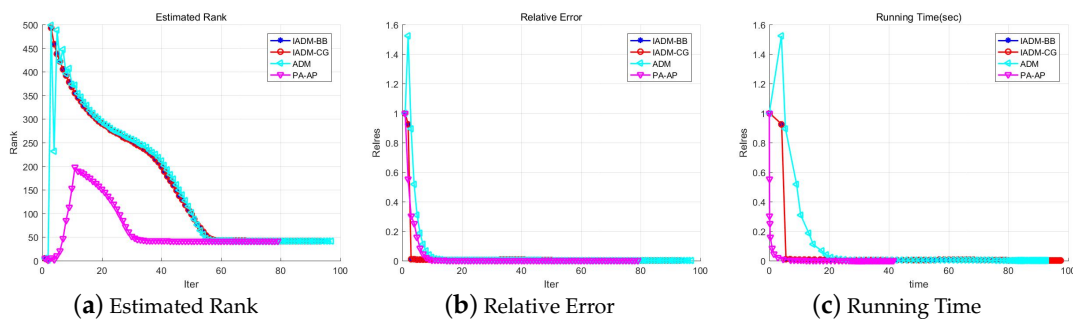
$$PSNR = 10 \cdot \log_{10} \left(\frac{\|X^*\|_{\infty}^2}{\frac{1}{mn} \|X^* - \hat{X}\|_F^2} \right),$$

where $\|X^*\|_{\infty}$ is the infinity norm of X^* , defined as the maximum absolute value of the elements in X^* . From the definition, higher PSNR indicates a better recovery result.

Table 6 shows the cost time, relative error and PSNR of recovery image by different methods. From Table 6, we can note that the PA-AP method is able to obtain higher PSNR as sr increases. Moreover, the running time of PA-AP is always much less than the other methods with different settings. Figure 2 shows the executing process of the different methods. From Figure 2, it is clear that our method can estimate the rank exactly after 30 iterations, and runs much less time before termination than other methods.

Table 6. Comparisons of PA-AP, ADM, IADM-CG and IADM-BB for low-rank image recovery.

Name	sr	PA-AP			ADM			IADM-CG			IADM-BB		
		Time	RelErr	PSNR	Time	RelErr	PSNR	Time	RelErr	PSNR	Time	RelErr	PSNR
Lena	0.2	58.66	1.730×10^{-4}	26.17	127.86	1.751×10^{-4}	26.61	148.71	1.625×10^{-4}	26.78	161.59	1.633×10^{-4}	26.78
	0.4	46.09	3.984×10^{-5}	84.67	94.65	4.578×10^{-5}	85.52	94.86	1.191×10^{-4}	76.06	110.74	1.244×10^{-4}	75.58
	0.6	7.52	2.071×10^{-5}	93.42	55.92	2.717×10^{-5}	92.60	54.78	8.376×10^{-5}	81.65	65.75	9.504×10^{-5}	80.40
	0.8	3.41	1.025×10^{-5}	101.59	37.32	1.771×10^{-5}	99.04	33.46	4.374×10^{-5}	89.62	39.09	6.070×10^{-5}	86.65
Pirate	0.2	54.23	1.561×10^{-4}	26.03	121.73	1.745×10^{-4}	26.25	140.40	1.643×10^{-4}	26.41	154.24	1.651×10^{-4}	26.41
	0.4	38.58	3.961×10^{-5}	85.42	92.33	4.905×10^{-5}	86.24	94.57	1.140×10^{-4}	76.98	112.77	8.735×10^{-5}	79.81
	0.6	8.15	1.313×10^{-5}	97.97	53.13	2.839×10^{-5}	92.92	54.81	4.376×10^{-5}	86.89	66.42	6.227×10^{-5}	84.26
	0.8	3.45	8.451×10^{-6}	104.65	33.83	1.967×10^{-5}	99.05	29.41	6.704×10^{-5}	91.70	36.86	1.001×10^{-5}	88.64
Cameraman	0.2	50.51	1.717×10^{-4}	24.10	124.54	1.965×10^{-4}	24.32	145.18	2.159×10^{-4}	24.43	159.92	2.167×10^{-4}	24.43
	0.4	46.81	5.290×10^{-5}	80.18	102.31	5.570×10^{-5}	81.30	106.05	1.182×10^{-5}	73.72	115.48	1.313×10^{-4}	73.00
	0.6	10.03	2.389×10^{-5}	90.92	59.87	3.090×10^{-5}	90.36	58.43	6.099×10^{-5}	83.07	71.95	5.343×10^{-5}	84.40
	0.8	4.18	1.613×10^{-5}	96.46	37.86	2.036×10^{-5}	96.03	35.31	3.360×10^{-5}	90.22	45.53	2.106×10^{-5}	93.04

**Figure 2.** Convergence behavior of the four methods (*Lena*, $sr = 0.4$, $\varepsilon_2 = 10^{-5}$). The first subfigure is the estimated rank; the second is the relative error to the original matrix; and the last is the running time.

5. Conclusions and Future Work

In this paper, a unified model and algorithm for the matrix nuclear norm minimization problem are proposed. In each iteration, the proposed algorithm mainly includes computing matrix singular value decompositions and solving proximity operators of two convex functions. In addition, the convergence of the algorithm is also proved. A large number of experimental results and numerical comparisons show that the algorithm is superior to IADM-BB, IADM-CG and ADM algorithms.

The problem of tensor completion has been widely studied recently [31]. One of our future works is to extend the proposed PA-AP algorithm to tensor completion.

Author Contributions: Methodology, W.H. and W.Z.; Investigation, W.H., W.Z. and G.Y.; Writing—original draft preparation, W.H. and W.Z.; Writing—review and editing, W.H. and G.Y.; Software, W.Z.; Project administration, G.Y.

Funding: This research was funded by the National Natural Science Foundation of China (Nos. 61863001, 11661007, 61702244, 11761010 and 61562003), the National Natural Science Foundation of Jiangxi Province (Nos. 20181BAB202021, JXJG-18-14-11, and 20192BAB205086), the National Science Foundation of Zhejiang Province, China (LD19A010002), Research programme (Nos. 18zb04, YCX18B001) and the ‘XieTong ChuangXin’ project of Gannan Normal University.

Acknowledgments: The authors would like to thank the anonymous reviewers that have highly improved the final version of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Candès, E.J.; Recht, B. Exact Matrix Completion via Convex Optimization. *Found. Comput. Math.* **2009**, *9*, 717–772.
2. Liu, G.; Lin, Z.C.; Yu, Y. Robust subspace segmentation by low-rank representation. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 663–670.

3. Dai, Y.; Li, H.; He, M. A simple prior-free method for non-rigid structure-from-motion factorization. *Int. J. Comput. Vis.* **2014**, *107*, 101–122. [[CrossRef](#)]
4. Hu, W.; Wang, Z.; Liu, S.; Yang, X.; Yu, G.; Zhang, J.J. Motion capture data completion via truncated nuclear norm regularization. *IEEE Signal Proc. Lett.* **2018**, *25*, 258–262. [[CrossRef](#)]
5. Lin, X.F.; Wei, G. Accelerated reweighted nuclear norm minimization algorithm for low rank matrix recovery. *Signal Process.* **2015**, *114*, 24–33. [[CrossRef](#)]
6. Candès, E.J.; Plan, Y. Matrix completion with noise. *Proc. IEEE* **2010**, *98*, 925–936.
7. Nie, F.; Huang, H.; Ding, C.H. Low-rank matrix recovery via efficient Schatten p-norm minimization. In Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, Toronto, ON, Canada, 22–26 July 2012; pp. 655–661.
8. Mohan, K.; Fazel, M. Iterative reweighted algorithms for matrix rank minimization. *J. Mach. Learn. Res.* **2012**, *13*, 3441–3473.
9. Zhang, D.; Hu, Y.; Ye, J.; Li, X.; He, X. Matrix completion by truncated nuclear norm regularization. In Proceedings of the Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 2192–2199.
10. Nie, F.; Hu, Z.; Li, X. Matrix completion based on non-convex low rank approximation. *IEEE Trans. Image Process.* **2019**, *28*, 2378–2388. [[CrossRef](#)]
11. Cui, A.; Peng, J.; Li, H.; Zhang, C.; Yu, Y. Affine matrix rank minimization problem via non-convex fraction function penalty. *J. Comput. Appl. Math.* **2018**, *336*, 353–374. [[CrossRef](#)]
12. Tütüncü, R.H.; Toh, K.C.; Todd, M.J. Solving semidefinite-quadratic-linear programs using SDPT3. *Math. Program.* **2003**, *95*, 189–217. [[CrossRef](#)]
13. Cai, J.F.; Candès, E.J.; Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **2010**, *20*, 1956–1982. [[CrossRef](#)]
14. Ma, S.; Goldfarb, D.; Chen, L. Fixed point and Bregman iterative methods for matrix rank minimization. *Math. Progr.* **2009**, *128*, 321–353. [[CrossRef](#)]
15. Toh, K.C.; Yun, S. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pac. J. Optim.* **2010**, *6*, 615–640.
16. Geng, J.; Wang, L.; Wang, X. Nuclear norm and indicator function model for matrix completion. *J. Inverse Ill-Posed Probl.* **2015**, *24*, 1–11. [[CrossRef](#)]
17. Lin, Z.; Chen, M.; Ma, Y. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv* **2010**, arXiv:1009.5055.
18. Chen, C.; He, B.; Yuan, X.M. Matrix completion via an alternating direction method. *IMA J. Numer. Anal.* **2012**, *32*, 227–245. [[CrossRef](#)]
19. Xiao, Y.H.; Jin, Z.F. An alternating direction method for linear-constrained matrix nuclear norm minimization. *Numer. Linear Algebra* **2012**, *19*, 541–554. [[CrossRef](#)]
20. Jin, Z.F.; Wang, Q.; Wan, Z. Recovering low-rank matrices from corrupted observations via the linear conjugate gradient algorithm. *J. Comput. Appl. Math.* **2014**, *256*, 114–120. [[CrossRef](#)]
21. Yang, J.; Yuan, X.M. Linearized augmented Lagrangian and alternating direction methods for nuclear minimization. *Math. Comput.* **2013**, *82*, 301–329. [[CrossRef](#)]
22. Barzilai, J.; Borwein, J.M. Two point step size gradient method. *IMA J. Numer. Anal.* **1988**, *4*, 141–148. [[CrossRef](#)]
23. Li, Q.; Shen, L.; Xu, Y. Multi-step fixed-point proximity algorithms for solving a class of optimization problems arising from image processing. *Adv. Comput. Math.* **2015**, *41*, 387–422. [[CrossRef](#)]
24. Zhang, X.; Burger, M.; Osher, S. A unified primal-dual algorithm framework based on Bregman iteration. *J. Sci. Comput.* **2011**, *46*, 20–46. [[CrossRef](#)]
25. Wang, J.H.; Meng, F.Y.; Pang, L.P.; Hao, X.H. An adaptive fixed-point proximity algorithm for solving total variation denoising models. *Inform. Sci.* **2017**, *402*, 69–81. [[CrossRef](#)]
26. Lin, Z.C.; Liu, R.; Su, Z. Linearized alternating direction method with adaptive penalty for low-rank representation. *Proc. Adv. Neural Inf. Process. Syst.* **2011**, *104*, 612–620.
27. Gabay, D.; Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Comput. Math. Appl.* **1976**, *2*, 17–40. [[CrossRef](#)]
28. Chen, P.; Huang, J.; Zhang, X. A primal-dual fixed point algorithm for convex separable minimization with applications to image restoration. *Inverse Probl.* **2013**, *29*, 025011. [[CrossRef](#)]

29. Micchelli, C.A.; Shen, L.; Xu, Y. Proximity algorithms for image models: Denoising. *Inverse Probl.* **2011**, *27*, 045009. [[CrossRef](#)]
30. Larsen, R.M. PROPACK-Software for Large and Sparse SVD Calculations. Available online: <http://sun.stanford.edu/srmunk/PROPACK/> (accessed on 1 September 2019).
31. Li, X.T.; Zhao, X.L.; Jiang, T.X.; Zheng, Y.B.; Ji, T.Y.; Huang, T.Z. Low-rank tensor completion via combined non-local self-similarity and low-rank regularization. *Neurocomputing* **2019**, *267*, 1–12.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).