Computers and Mathematics with Applications 71 (2016) 133–146

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Comparisons of several algorithms for Toeplitz matrix recovery

Chuan-Long Wang[a], Chao Li[a,*], Jin Wang[b]

[a] Higher Education Key Laboratory of Engineering Science Computing in Shanxi Province, Taiyuan Normal University, Taiyuan 030012, PR China

[b] School of Computer Science and Technology, Beihang University, Beijing 100191, PR China

ARTICLE INFO

Article history:
Received 20 March 2015
Received in revised form 11 October 2015
Accepted 9 November 2015
Available online 30 November 2015

Keywords:
Matrix recovery
Toeplitz matrix
Mean value
Augmented Lagrange multiplier

ABSTRACT

In this paper, we study algorithms for Toeplitz matrix recovery. Inspired by the singular value thresholding (SVT) algorithm for matrix completion and the alternating directions iterative method, we first propose a new mean value algorithm for Toeplitz matrix recovery. Then we apply our idea to the augmented Lagrange multiplier (ALM) algorithm for matrix recovery and put forward four modified ALM algorithms for Toeplitz matrix recovery. Convergence analysis of the new algorithms is discussed. All the iterative matrices generated by the five algorithms keep a Toeplitz structure that ensures the fast singular value decomposition (SVD) of Toeplitz matrices. Compared with the original algorithms, our algorithms are far superior in the time of SVD, as well as the CPU time.

1. Introduction

The study of recovering a corrupted low-rank matrix has experienced amazing growth in recent years. This problem is well known as the matrix recovery (MR) problem, as well as the Robust PCA. It arises in a large number of application areas [1–3].

MR problem was first proposed by Wright [4] and Candès [5]. In [4] Wright showed that a low-rank matrix A from $D = A + E$ with sufficiently errors E can be exactly recovered under rather broad conditions by solving the following convex optimization problem,

$$\begin{aligned} &\min_{A,E} \quad \|A\|_* + \lambda\|E\|_1 \\ &\text{s.t.} \quad D = A + E \end{aligned} \tag{1.1}$$

where $\|A\|_* = \sum_{k=1}^{r} \sigma_k(A)$, $\sigma_k(A)$ denotes the kth largest singular value of $A \in \mathbb{R}^{n_1 \times n_2}$ of rank r. $\|E\|_1$ denotes the sum of the absolute values of matrix entries, and λ is a positive weighting parameter. In [4,5], the best choice of λ is $\frac{1}{\sqrt{n_1}}$. Throughout this paper, unless otherwise specified, we will fix $\lambda = \frac{1}{\sqrt{n_1}}$.

Many algorithms have been proposed to solve the optimization problem (1.1). Wright et al. [4] presented an iterative thresholding (IT) algorithm, which requires a large number of iterations to converge. Then Lin et al. [6,7] gave two new algorithms for solving the optimal problem (1.1), one is the accelerated proximal gradient (APG) algorithm; the other is the

* Corresponding author.
E-mail addresses: clwang1964@163.com (C.-L. Wang), l_chao1989@163.com (C. Li), wjyz91@163.com (J. Wang).

http://dx.doi.org/10.1016/j.camwa.2015.11.010
0898-1221/

dual algorithm. Both the APG algorithm and the dual algorithm are at least 50 times faster than the IT algorithm. In 2010, Lin et al. [8] put forward the augmented Lagrange multiplier (ALM) algorithm, which has been proved to have a Q-linear convergence speed.

On the other hand, as an important special matrix, Toeplitz matrices arise naturally in certain application areas such as the system identification [9], medical imaging [10], the multiple-input multiple-output (MIMO) communication system [11], image restoration [12]. Therefore, some scholars have studied Toeplitz matrices, such as Shaw et al. [13], Kailath et al. [14]. It is worth mentioning that Qiao et al. [15,16] put forward an $O(n^2 \log n)$ algorithm for the fast SVD of Toeplitz and Hankel matrices by combining the Lanczos method [17] and the FFT technique [18].

We can see from the foregoing algorithms for the matrix recovery problem that most of the algorithms need to compute SVD, which is time-consuming and accounts for at least 85% of the CPU time. Therefore, we can take full advantage of the fast SVD of Toeplitz matrices to reduce computational complexity, as well as the CPU time. Together with the value of Toeplitz matrices in the signal and image processing, it is very meaningful to study Toeplitz matrix recovery problem.

In this paper, we focus our attention on the recovery of Toeplitz matrices. Combining the idea of the mean value algorithm for Toeplitz matrix completion [19] and the alternating directions iterative method, we first propose a new mean value algorithm for Toeplitz matrix recovery. Then we present four modified ALM algorithms for Toeplitz matrix recovery. First, we give some definitions.

Definition 1 (*[17]*)**.** An $n \times n$ Toeplitz matrix $T \in \mathbb{R}^{n\times n}$ is of the form,

$$T = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-2} & t_{n-1} \\ t_{-1} & t_0 & \cdots & t_{n-3} & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{-n+2} & t_{-n+3} & \cdots & t_0 & t_1 \\ t_{-n+1} & t_{-n+2} & \cdots & t_{-1} & t_0 \end{pmatrix}.$$

Note: T is determined by its first row and first column, a total of $(2n-1)$ entries.

Definition 2 (*Singular Value Decomposition (SVD) [17]*)**.** The singular value decomposition of a matrix $X \in \mathbb{R}^{n_1\times n_2}$ of rank r is:

$$X = U\Sigma_r V^*, \quad \Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r),$$

where $U \in \mathbb{R}^{n_1\times r}$ and $V \in \mathbb{R}^{n_2\times r}$ are orthogonal, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

Definition 3 (*Singular Value Thresholding Operator [20]*)**.** For each $\tau \geq 0$, the singular value thresholding operator $\mathcal{D}_\tau$ is defined as follows:

$$\mathcal{D}_\tau(X) := U\mathcal{D}_\tau(\Sigma)V^*, \quad \mathcal{D}_\tau(\Sigma) = \mathrm{diag}(\{\sigma_i - \tau\}_+)$$

where $X = U\Sigma_r V^*$ is the SVD of a matrix X of rank r, $\{\sigma_i - \tau\}_+ = \begin{cases} \sigma_i - \tau, & \text{if } \sigma_i > \tau \\ 0, & \text{if } \sigma_i \leq \tau. \end{cases}$

Definition 4 (*Soft-Thresholding (Shrinkage) Operator [8]*)**.** For each $\varepsilon \geq 0$, the soft-thresholding (shrinkage) operator $\mathcal{S}_\varepsilon$ is defined as follows:

$$\mathcal{S}_\varepsilon[x] = \begin{cases} x - \varepsilon, & \text{if } x > \varepsilon, \\ x + \varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{otherwise}, \end{cases}$$

where $x \in \mathbf{R}$.

The rest of this paper is organized as follows. In Section 2, we describe our algorithms for Toeplitz matrix recovery in detail and their convergence is established in Section 3. In Section 4, we compare our algorithms with the ALM algorithm and SVT algorithm through numerical experiments. Conclusions are given in Section 5.

Notation. For convenience, $\mathbf{R}$ denotes the set of real numbers. $\mathbb{R}^{n_1\times n_2}$ denotes $n_1 \times n_2$ real matrices set. $r(X)$ denotes the rank of a matrix X. x_{ij} denotes the (i, j)th entry of a matrix X. The nuclear norm of a matrix is denoted by $\|X\|_*$, the Frobenius norm by $\|X\|_F$, $\|X\|_1$ denotes the sum of the absolute values of matrix entries, and $|X|_0$ denotes the number of nonzero elements of a matrix X. X^* is the conjugate transpose of a matrix X. The standard inner product of two matrices is denoted by $\langle X, Y\rangle = \mathrm{trace}(X^*Y)$. $\Omega = \{-n_1 + 1, \ldots, n_2 - 1\}$ are the indices of diagonals of a matrix X. Vector $\mathrm{diag}(X, l)$ denotes the lth diagonal of a Toeplitz matrix X, $l \in \Omega$. The mean value of a vector x is denoted by $\mathrm{mean}(x)$, the median value by $\mathrm{median}(x)$.

2. Algorithms for Toeplitz matrix recovery

In this section, we focus on the recovery of a Toeplitz matrix, then our problem is expressed as the following convex programming,

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \lambda \|E\|_1 \\ \text{s.t.} \quad & D = A + E \end{aligned} \tag{2.1}$$

where A is a Toeplitz matrix.

For convenience, $[U_k, \Sigma_k, V_k] = lansvd(Y_k)$, denotes the SVD of the matrix Y_k using the Lanczos method. And let

$$R_l = (r_{ij})_{n\times n} = \begin{cases} 1, & j - i = l \\ 0, & j - i \neq l, \end{cases} \quad l = -n+1, \dots, n-1. \tag{2.2}$$

Algorithm 2.1 (*A Mean Value (MV) Algorithm for Toeplitz Matrix Recovery*).

Step 0. Set parameters $\tau_0, \lambda, c(0 < c < 1)$, tolerance ϵ, set initial matrix $D, X_0 = 0, E_0 = 0, k := 0$;
Step 1. Compute $a_l = \text{mean}(\text{diag}(D, l)), l \in \Omega$,
set

$$A_0 = \sum_{l\in\Omega} a_l R_l;$$

Step 2. Compute the SVD of A_k,

$$[U_k, \Sigma_k, V_k] = lansvd(A_k),$$

set

$$\begin{aligned} X_{k+1} &= U_k \mathcal{D}_{\tau_k}(\Sigma_k) V_k^*, \\ E_{k+1} &= \mathcal{S}_{\lambda\tau_k}[D - X_{k+1}], \\ \bar{A}_{k+1} &= D - E_{k+1}; \end{aligned}$$

Step 3. Compute $a_l = \text{mean}(\text{diag}(\bar{A}_{k+1}, l)), l \in \Omega$,
set

$$A_{k+1} = \sum_{l\in\Omega} a_l R_l,$$

go to Step 4;
Step 4. If $\|A_{k+1} - A_k\|_F / \|A_k\|_F < \epsilon$, stop; Otherwise, if $\|A_{k+1}\|_* + \lambda\|E_{k+1}\|_1 > \|A_k\|_* + \lambda\|E_k\|_1$, $\tau_{k+1} = c\tau_k$, $A_{k+1} = A_k$, $E_{k+1} = E_k$; $k := k+1$, go to Step 2.

Algorithm 2.2 (*A Mean-Value Modified Augmented Lagrange Multiplier (M-ALM) Algorithm for Toeplitz Matrix Recovery*).

Step 0. Set parameters μ_0, λ, ρ, tolerances ϵ_1, ϵ_2, set initial matrix $Y_0, A_0 = 0, E_0 = 0, k := 0$;
Step 1. Compute $a_l = \text{mean}(\text{diag}(D - E_k + \mu_k^{-1} Y_k, l)), l \in \Omega$,
set

$$\bar{A}_k = \sum_{l\in\Omega} a_l R_l;$$

Step 2. Compute the SVD of $\bar{A}_k$,

$$[U_k, \Sigma_k, V_k] = lansvd(\bar{A}_k),$$

set

$$\begin{aligned} A_{k+1} &= U_k \mathcal{D}_{\mu_k^{-1}}(\Sigma_k) V_k^*, \\ E_{k+1} &= \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu^{-1} Y_k]; \end{aligned}$$

Step 3. If $\|D - A_{k+1} - E_{k+1}\|_F / \|D\|_F < \epsilon$ and $\mu_k \|E_{k+1} - E_k\|_F / \|D\|_F < \epsilon_2$, stop, then
compute $b_l = \text{mean}(\text{diag}(A_{k+1}, l)), l \in \Omega$, set

$$\hat{A}_{k+1} = \sum_{l\in\Omega} b_l R_l;$$

Otherwise, go to Step 4;

Step 4. Set $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$, $\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \mu_k\|E_{k+1} - E_k\|_F/\|D\|_F < \epsilon_2, \\ \mu_k, & \text{otherwise,} \end{cases}$ $k := k+1$; go to Step 1.

Algorithm 2.3 (*A Double Mean-Value Modified Augmented Lagrange Multiplier (2M-ALM) Algorithm for Toeplitz Matrix Recovery*)**.**

Step 0. Set parameters μ_0, λ, ρ, tolerances ϵ_1, ϵ_2, set initial matrix $Y_0, A_0 = 0, E_0 = 0, k := 0$;
Step 1. Compute $a_l = \text{mean}(\text{diag}(D - E_k + \mu_k^{-1}Y_k, l)), l \in \Omega$,
set

$$\bar{A}_k = \sum_{l\in\Omega} a_l R_l;$$

Step 2. Compute the SVD of $\bar{A}_k$,

$$[U_k, \Sigma_k, V_k] = lansvd(\bar{A}_k),$$

set

$$\hat{A}_{k+1} = U_k \mathcal{D}_{\mu_k^{-1}}(\Sigma_k)V_k^*,$$

compute $b_l = \text{mean}(\text{diag}(\hat{A}_{k+1}, l)), l \in \Omega$, set

$$A_{k+1} = \sum_{l\in\Omega} b_l R_l;$$

$$E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu^{-1}Y_k];$$

Step 3. If $\|D - A_{k+1} - E_{k+1}\|_F/\|D\|_F < \epsilon$ and $\mu_k\|E_{k+1} - E_k\|_F/\|D\|_F < \epsilon_2$, stop; otherwise, go to Step 4;
Step 4. Set $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$, $\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \mu_k\|E_{k+1} - E_k\|_F/\|D\|_F < \epsilon_2, \\ \mu_k, & \text{otherwise,} \end{cases}$ $k := k+1$; go to Step 1.

Algorithm 2.4 (*A Mid-Value Modified Augmented Lagrange Multiplier (Mid-ALM) Algorithm for Toeplitz Matrix Recovery*)**.**

Step 0. Set parameters μ_0, λ, ρ, tolerances ϵ_1, ϵ_2, set initial matrix $Y_0, A_0 = 0, E_0 = 0, k := 0$;
Step 1. Compute $a_l = \text{median}(\text{diag}(D - E_k + \mu_k^{-1}Y_k, l)), l \in \Omega$,
set

$$\bar{A}_k = \sum_{l\in\Omega} a_l R_l;$$

Step 2. Compute the SVD of $\bar{A}_k$,

$$[U_k, \Sigma_k, V_k] = lansvd(\bar{A}_k),$$

set

$$A_{k+1} = U_k \mathcal{D}_{\mu_k^{-1}}(\Sigma_k)V_k^*,$$

$$E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu^{-1}Y_k];$$

Step 3. If $\|D - A_{k+1} - E_{k+1}\|_F/\|D\|_F < \epsilon_1$ and $\mu_k\|E_{k+1} - E_k\|_F/\|D\|_F < \epsilon_2$, stop, then
compute $b_l = \text{median}(\text{diag}(A_{k+1}, l)), l \in \Omega$, set

$$\hat{A}_{k+1} = \sum_{l\in\Omega} b_l R_l;$$

Otherwise, go to Step 4;

Step 4. Set $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$, $\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \mu_k\|E_{k+1} - E_k\|_F/\|D\|_F < \epsilon_2, \\ \mu_k, & \text{otherwise,} \end{cases}$
$k := k+1$; go to Step 1.

Algorithm 2.5 (*A Double Mid-Value Modified Augmented Lagrange Multiplier (2Mid-ALM) Algorithm for Toeplitz Matrix Recovery*)**.**

Step 0. Set parameters μ_0, λ, ρ, tolerances ϵ_1, ϵ_2, set initial matrix $Y_0, A_0 = 0, E_0 = 0, k := 0$;
Step 1. Compute $a_l = \text{median}(\text{diag}(D - E_k + \mu_k^{-1}Y_k, l)), l \in \Omega$,
set

$$\bar{A}_k = \sum_{l\in\Omega} a_l R_l;$$

Step 2. Compute the SVD of $\bar{A}_k$,

$$[U_k, \Sigma_k, V_k] = lansvd(\bar{A}_k),$$

set

$$\hat{A}_{k+1} = U_k \mathcal{D}_{\mu_k^{-1}}(\Sigma_k) V_k^*,$$

compute $b_l = \text{median}(\text{diag}(\hat{A}_{k+1}, l))$, $l \in \Omega$, set

$$A_{k+1} = \sum_{l \in \Omega} b_l R_l;$$

$$E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu^{-1} Y_k];$$

Step 3. If $\|D - A_{k+1} - E_{k+1}\|_F / \|D\|_F < \epsilon_1$ and $\mu_k \|E_{k+1} - E_k\|_F / \|D\|_F < \epsilon_2$, stop; otherwise, go to Step 4;

Step 4. Set $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$, $\mu_{k+1} = \begin{cases} \rho\mu_k, & \text{if } \mu_k \|E_{k+1} - E_k\|_F / \|D\|_F < \epsilon_2, \\ \mu_k, & \text{otherwise}, \end{cases}$

$k := k + 1$; go to Step 1.

3. Convergence analysis

We set $f(A, E) = \|A\|_* + \lambda \|E\|_1$. And let

$$M(X) = (\bar{x}_{ij}) = \sum_{l \in \Omega} a_l R_l, \qquad Mid(X) = \sum_{l \in \Omega} b_l R_l,$$

where $X \in \mathbb{R}^{n \times n}$, $a_l = \text{mean}(\text{diag}(X, l))$, $b_l = \text{median}(\text{diag}(X, l))$, $l \in \Omega$, R_l is defined as (2.2).

Lemma 3.1. *Let Y be an $n \times n$ matrix, and X be a Toeplitz matrix. Then*

$$\langle Y - M(Y), X \rangle = 0. \tag{3.1}$$

Proof.

$$\begin{aligned} \langle Y, X \rangle &= \sum_{i=1}^{n} \sum_{k=1}^{n} y_{ki} x_{ki} \\ &= y_{n1} x_{n1} + (y_{n-11} + y_{n2}) x_{n2} + \cdots + (y_{ki} + y_{k+1i+1} + \cdots + y_{ni+n-k}) x_{ki} + \cdots + y_{1n} x_{1n} \\ &= \sum_{i=1}^{n} \sum_{k=1}^{n} \bar{y}_{ki} x_{ki} \\ &= \langle M(Y), X \rangle. \end{aligned}$$

Hence, (3.1) holds.

Theorem 3.2. *Let $(\hat{A}, \hat{E})$ be the optimal solution of (2.1), if and only if there exists a subgradient V such that*

$$M(V) = 0, \quad V \in \frac{\partial f}{\partial \hat{A}}. \tag{3.2}$$

Proof. Let A be a Toeplitz matrix,

$$\begin{aligned} f(A) &\geq f(\hat{A}) + \langle V, A - \hat{A} \rangle \\ &= f(\hat{A}) + \langle M(V), A - \hat{A} \rangle. \end{aligned}$$

If there exists a subgradient V such that $M(V) = 0$, then $\hat{A}$ is an optimal solution.

On the other hand, as we know, $\frac{\partial f}{\partial \hat{A}}$ is a closed convex set, so $M(\frac{\partial f}{\partial \hat{A}})$ is also a closed convex set. Using separation theorem of convex set, if $0 \notin M(V)$, $V \in \frac{\partial f}{\partial \hat{A}}$, there exists a Toeplitz matrix Y such that $\langle M(V), Y \rangle < 0, \forall V \in \frac{\partial f}{\partial \hat{A}}$. Let $A = Y + \hat{A}$, then

$$f(A) = f(\hat{A}) + \sup_{V \in \frac{\partial f}{\partial \hat{A}}} \langle V, A - \hat{A} \rangle = f(\hat{A}) + \sup_{V \in \frac{\partial f}{\partial \hat{A}}} \langle M(V), Y \rangle < f(\hat{A}),$$

which is a contradiction with assumption of $\hat{A}$.

Corollary 3.1. *Let $\hat{A}$ be an $n \times n$ matrix with the singular value decomposition $\hat{A} = U\Sigma_r V^*$, if*

$$M(UV^* - \lambda(D - \hat{A})_+^+) = 0 \tag{3.3}$$

where $(D-\hat{A})_+^+ = \begin{cases} 1, & d_{ij} - a_{ij} > 0 \\ 0, & d_{ij} - a_{ij} = 0, \\ -1, & d_{ij} - a_{ij} < 0 \end{cases}$ *Then, $\hat{A}$ is an optimal matrix of (1.1).*

Proof.

$$\begin{aligned} \frac{\partial f}{\partial \hat{A}} &= \frac{\partial \|\hat{A}\|_*}{\partial \hat{A}} - \lambda \frac{\|D - \hat{A}\|_1}{\partial \hat{A}} \\ &= UV^* + W - \lambda(D - \hat{A})_+ \end{aligned}$$

where the row of W is orthogonal to the row of V, the column of W is orthogonal to the column of U, and $\|W\|_2 \le 1$,

$$(D - \hat{A})_+ = \begin{cases} 1, & d_{ij} - a_{ij} > 0, \\ [-1, 1], & d_{ij} - a_{ij} = 0, \\ -1, & d_{ij} - a_{ij} < 0. \end{cases}$$

Let $W = 0$,

$$(D - \hat{A})_+^+ = \begin{cases} 1, & d_{ij} - a_{ij} > 0, \\ 0, & d_{ij} - a_{ij} = 0, \\ -1, & d_{ij} - a_{ij} < 0. \end{cases}$$

From Theorem 3.2, we obtain the corollary.

Lemma 3.3. *Let A be an $n \times n$ matrix with rank n, then there is a positive number τ such that*

$$\|\mathcal{D}_\tau(A)\|_* + \lambda\|A - \mathcal{D}_\tau(A)\|_1 \le \|A\|_*. \tag{3.4}$$

Proof. From the assumption of A, there is a positive number τ such that

$$\|\mathcal{D}_\tau(A)\|_* \le \|A\|_* - n\tau. \tag{3.5}$$

On the other hand,

$$\|A - \mathcal{D}_\tau(A)\|_1 = \tau\|UV^*\|_1 \le n\sqrt{n}\tau,$$

because $\lambda \le \frac{1}{\sqrt{n}}$, so (3.4) holds.

Theorem 3.4. *Let (A_k, E_k) be the sequence of matrices generated by Algorithm 2.1, and $(\hat{A}, \hat{E})$ be the optimal solution of (2.1). Then any accumulation point of (A_k, E_k) is $(\hat{A}, \hat{E})$,*

$$\lim_{k\to\infty} A_k = \hat{A}, \qquad \lim_{k\to\infty} E_k = \hat{E}. \tag{3.6}$$

Proof. From Algorithm 2.1, we have

$$f(A_k) \le f(A_{k-1}).$$

If there is some k_0 such that $0 \in M(\frac{\partial f}{\partial A_{k_0}})$, then (A_{k_0}, E_{k_0}) is the optimal solution of (2.1).

Otherwise, suppose $\lim_{k\to\infty} f(A_k) = f(\hat{A})$, there is a subsequence $\{A_{ki}\}$ such that $\{A_{ki}\} \to \hat{A}$. Since $\tau_k \to 0$, so $\|A_{k+1} - A_k\|_F \to 0$, and $\lim_{k\to\infty} A_k = \hat{A}$. Because $D = A_k + E_k$, thus $\lim_{k\to\infty} E_k = \hat{E}$.

Next, we prove that $(\hat{A}, \hat{E})$ is the optimal solution of (2.1). First, let $\langle S, X\rangle = \sup_{V\in S}\langle V, X\rangle$, where $S \subset \mathbb{R}^{n\times n}$, X is an $n \times n$ matrix.

If $(\hat{A}, \hat{E})$ is not the optimal solution of (2.1), then $0 \notin M(\frac{\partial f}{\partial \hat{A}})$. Since $-U_{\hat{A}}V_{\hat{A}}^*$ is a descent direction of $\|A\|_*$ and $\lambda(D - \hat{A})_+^+$ is also a descent direction of $\|\lambda\hat{E}\|_1$, so

$$\left\langle M\left(\frac{\partial f}{\partial \hat{A}}\right), -U_{\hat{A}}V_{\hat{A}}^* + \lambda(D - \hat{A})_+^+\right\rangle < 0. \tag{3.7}$$

Combining $\lim_{k\to\infty}\frac{\partial f}{\partial A_k}\subset\frac{\partial f}{\partial \hat{A}}$, we have

$$\begin{aligned} f(A_{k+1}) &\le f(A_k)+\left\langle \frac{\partial f}{\partial A_{k+1}}, A_{k+1}-A_k\right\rangle \\ &\le f(A_k)+\left\langle \frac{\partial f}{\partial \hat{A}}, A_{k+1}-A_k\right\rangle. \end{aligned}$$

From Algorithm 2.1, we have

$$A_{k+1}=A_k-\tau_k M(-U_{A_k}\Sigma_{A_k}V_{A_k}^*+\lambda(D-A_k)_+),$$

and

$$\lim_{k\to\infty} -U_{A_k}\Sigma_{A_k}V_{A_k}^*+\lambda(D-A_k)_+=-U_{\hat{A}}V_{\hat{A}}^*+\lambda(D-\hat{A})_+^+,$$

where $\Sigma_{A_k}=\begin{pmatrix}1 &&&&&& \\ &\ddots&&&&& \\ &&1&&&& \\ &&&c_{i_k}&&& \\ &&&&\ddots&& \\ &&&&&c_{r_k}& \\ &&&&&&0\end{pmatrix}$, $r_k=\operatorname{rank}(A_k),\ 0<c_j<1,\ j=i_k,\dots,r_k,\ i_k\ge r(\hat{A})$.

Combining (3.7), there is a $\tau>0$ such that $\tau_k\ge\tau$ and $f(A_{k+1})\le f(A_k)$, which is a contradiction with $\{\tau_k\to 0\}$.

Theorem 3.5. *Let (A_k,E_k) be the matrix sequences generated by Algorithm 2.3, and $(\hat{A},\hat{E})$ be the optimal solution of (2.1). Suppose that $\mu_k\to\infty$ and $\sum_{k=1}^{\infty}\mu_k^{-1}=+\infty$, then*

$$\lim_{k\to\infty}A_k=\hat{A},\qquad \lim_{k\to\infty}E_k=\hat{E}. \tag{3.8}$$

Proof. From $E_{k+1}=\mathcal{S}_{\lambda\mu_k^{-1}}(D-A_{k+1}+\mu_k^{-1}Y_k)$, we have

$$E_{k+1}=D-A_{k+1}+\frac{1}{\mu_k}Y_k-\frac{\lambda}{\mu_k}B,$$

where $\|B\|_\infty\le n$. So,

$$\begin{aligned} Y_{k+1} &= Y_k+\mu_k(D-A_{k+1}-E_{k+1}) \\ &= Y_k+\mu_k\left(-\frac{1}{\mu_k}Y_k+\frac{\lambda}{\mu_k}B\right) \\ &= \lambda B. \end{aligned}$$

Further, $\|Y_{k+1}\|_\infty\le\lambda\|B\|_\infty\le\lambda n=\sqrt{n}$, $\{Y_k\}$ is bounded, which implies: when $\mu_k\to\infty$,

$$\lim_{k\to\infty}A_k+E_k=D. \tag{3.9}$$

Let $\hat{Y}_{k+1}=Y_k+\mu_k(D-A_{k+1}-E_k)$, $\hat{Y}$ is the optimal Lagrange multiplier. Using the same analysis of Lemma 2 in [8], we have

$$\begin{aligned} \|E_{k+1}-\hat{E}\|_F^2+\mu_k^{-2}\|Y_{k+1}-\hat{Y}\|_F^2 &= \|E_k-\hat{E}\|_F^2+\mu_k^{-2}\|Y_k-\hat{Y}\|_F^2-\|E_{k+1}-E_k\|_F^2 \\ &\quad -\mu_k^{-2}\|Y_{k+1}-Y_k\|_F^2-2\mu_k^{-1}(\langle Y_{k+1}-Y_k, E_{k+1}-E_k\rangle \\ &\quad +\langle A_{k+1}-\hat{A}, \hat{Y}_{k+1}-\hat{Y}\rangle+\langle E_{k+1}-\hat{E}, Y_{k+1}-\hat{Y}\rangle). \end{aligned} \tag{3.10}$$

From Algorithm 2.3, $M(\hat{Y}_{k+1})\in\partial\|A_{k+1}\|_*$.

Hence, from Lemma 3.1,

$$\begin{aligned} \langle A_{k+1}-\hat{A}, \hat{Y}_{k+1}-\hat{Y}\rangle &= \langle A_{k+1}-\hat{A}, M(\hat{Y}_{k+1}-\hat{Y})\rangle \\ &= \langle A_{k+1}-\hat{A}, M(\hat{Y}_{k+1})-M(\hat{Y})\rangle \\ &= \langle A_{k+1}-\hat{A}, M(\hat{Y}_{k+1})-\hat{Y}\rangle \\ &\ge 0. \end{aligned} \tag{3.11}$$

Table 1
Comparison between MV algorithm and SVT algorithm. Corresponding to each triplet $\{n, r\}$, the MR problem was solved for the same data matrix D using the two different algorithms.

n	r	*Algorithm*	*iter*	$\lvert\hat{E}\rvert_0 - \lvert E^\star\rvert_0$	$\frac{\|\hat{A}-A^\star\|_F}{\|A^\star\|_F}$	$\frac{\|\hat{E}-E^\star\|_F}{\|E^\star\|_F}$	t(SVD) (s)	Time (s)
$\lvert E^\star\rvert_0 = 0.05n^2$								
800	8	MV	85	0	3.8119e−09	2.2101e−08	4.522	13.5247
		SVT	86	0	8.7185e−09	1.1421e−08	16.850	31.9596
1000	10	MV	86	0	2.7442e−09	1.0429e−08	10.985	25.9431
		SVT	87	0	9.8249e−09	8.4060e−09	27.796	50.4343
1500	15	MV	86	185	4.5101e−04	4.5588e−04	6.901	34.0967
		SVT	87	185	4.5101e−04	4.5588e−04	53.561	76.2599
2000	20	MV	87	0	3.4323e−09	2.0359e−08	11.053	55.4128
		SVT	89	0	6.4041e−09	8.4654e−09	130.701	170.3904
2500	25	MV	87	998	6.7984e−04	6.0053e−04	14.470	73.6957
		SVT	87	998	6.7984e−04	6.0053e−04	129.715	185.6879
3000	30	MV	87	0	3.8082e−09	9.0090e−09	23.822	109.1658
		SVT	88	0	1.3742e−08	7.2255e−09	216.006	293.1819
4000	30	MV	88	0	3.5361e−09	6.7979e−09	50.394	211.9642
		SVT	88	0	1.5964e−08	6.8111e−09	524.141	670.6480

Since $Y_{k+1} \in \partial(\|\lambda E_{k+1}\|_1)$, $f(A, E)$ is a convex function,

$$\langle E_{k+1} - \hat{E}, Y_{k+1} - \hat{Y}\rangle \geq 0, \tag{3.12}$$
$$\langle E_{k+1} - E_k, Y_{k+1} - Y_k\rangle \geq 0. \tag{3.13}$$

Thus, Lemma 4 in [8] holds for $\{A_k, Y_k, E_k\}$ generated by Algorithm 2.3. Using the same proof of Theorem 2 in [8], we obtain

$$\lim_{k\to\infty} A_k = \hat{A}, \qquad \lim_{k\to\infty} E_k = \hat{E}.$$

Theorem 3.6. *Let (A_k, E_k) be the matrix sequences generated by Algorithm 2.2, and $(\hat{A}, \hat{E})$ be the optimal solution of (2.1). Suppose that $\mu_k \to \infty$ and $\sum_{k=1}^{\infty} \mu_k^{-1} = \infty$. Let $\hat{Y}_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_k)$ and $\bar{Y}_{k+1} = M(Y_k) + \mu_k(M(D - E_k) - A_{k+1})$. If*

$$\langle A_{k+1} - \hat{A}, \hat{Y}_{k+1} - \hat{Y}\rangle \geq \langle A_{k+1} - \hat{A}, \bar{Y}_{k+1} - \hat{Y}\rangle, \tag{3.14}$$

then

$$\lim_{k\to\infty} A_k = \hat{A}, \qquad \lim_{k\to\infty} E_k = \hat{E}. \tag{3.15}$$

Proof. For $\{A_k, E_k, Y_k\}$ generated by Algorithm 2.2, using the same proof as Theorem 3.5, (3.9) and (3.10) are obtained. Since $Y_{k+1} \in \partial(\|\lambda E_{k+1}\|_1)$ and $f(A, E)$ is a convex function, (3.12) and (3.13) hold.

Now, we prove (3.11) for $\{A_k, E_k, Y_k\}$ generated by Algorithm 2.2. Because $\bar{Y}_{k+1} \in \partial\|A_{k+1}\|_*$, so

$$\langle A_{k+1} - \hat{A}, \bar{Y}_{k+1} - \hat{Y}\rangle \geq 0.$$

Thus,

$$\langle A_{k+1} - \hat{A}, \hat{Y}_{k+1} - \hat{Y}\rangle \geq \langle A_{k+1} - \hat{A}, \bar{Y}_{k+1} - \hat{Y}\rangle \geq 0,$$

(3.11) and Lemma 4 in [8] hold. Using the same proof of Theorem 2 in [8], Theorem 3.6 is obtained easily.

Theorem 3.7. *Let (A_k, E_k) be the matrix sequences generated by Algorithm 2.4, and $(\hat{A}, \hat{E})$ be the optimal solution of (2.1). Suppose that $\mu_k \to \infty$ and $\sum_{k=1}^{\infty} \mu_k^{-1} = \infty$. Let $\hat{Y}_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_k)$ and $\tilde{Y}_{k+1} = Mid(Y_k + \mu_k(D - E_k)) - \mu_k A_{k+1}$. If*

$$\langle A_{k+1} - \hat{A}, \hat{Y}_{k+1} - \hat{Y}\rangle \geq \langle A_{k+1} - \hat{A}, \tilde{Y}_{k+1} - \hat{Y}\rangle, \tag{3.16}$$

then

$$\lim_{k\to\infty} A_k = \hat{A}, \qquad \lim_{k\to\infty} E_k = \hat{E}. \tag{3.17}$$

Table 2
Comparison between MV algorithm and SVT algorithm. Corresponding to each triplet $\{n, r\}$, the MR problem was solved for the same data matrix D using the two different algorithms.

n	r	*Algorithm*	*iter*	$\lvert\hat{E}\rvert_0 - \lvert E^\star\rvert_0$	$\frac{\lVert\hat{A}-A^\star\rVert_F}{\lVert A^\star\rVert_F}$	$\frac{\lVert\hat{E}-E^\star\rVert_F}{\lVert E^\star\rVert_F}$	t(SVD) (s)	Time (s)
$\lvert E^\star\rvert_0 = 0.1n^2$								
800	8	MV	88	0	3.2221e−09	9.6472e−09	5.043	15.2575
		SVT	87	0	1.0443e−08	9.7513e−09	21.495	30.2207
1000	10	MV	88	0	3.6186e−09	7.1137e−09	5.756	18.4896
		SVT	89	0	9.3849e−09	5.7418e−09	29.910	40.5902
1500	15	MV	88	197	4.5101e−04	3.2681e−04	7.020	30.4996
		SVT	90	197	4.5101e−04	3.2681e−04	60.886	88.9610
2000	20	MV	88	0	3.6410e−09	1.1129e−08	10.776	48.5334
		SVT	90	0	7.6481e−09	7.2444e−09	109.505	145.6478
2500	25	MV	89	995	6.7984e−04	4.2986e−04	14.020	75.5474
		SVT	89	995	6.7984e−04	4.2986e−04	164.127	212.9038
3000	30	MV	91	0	3 3.2581e−09	3.9646e−09	27.035	119.0347
		SVT	91	0	1.0558e−08	3.9763e−09	243.155	314.2432
4000	30	MV	92	0	3.7095e−09	3.6657e−09	27.620	191.5742
		SVT	93	0	9.6198e−09	2.9391e−09	544.371	711.1360

Table 3
Comparison between different algorithms. Corresponding to each triplet $\{n, r\}$, the MR problem was solved for the same data matrix D using the five different algorithms.

n	r	*Algorithm*	*iter*	$\lvert\hat{E}\rvert_0 - \lvert E^\star\rvert_0$	$\frac{\lVert\hat{A}-A^\star\rVert_F}{\lVert A^\star\rVert_F}$	$\frac{\lVert\hat{E}-E^\star\rVert_F}{\lVert E^\star\rVert_F}$	t(SVD) (s)	Time (s)
$\lvert E^\star\rvert_0 = 0.05n^2$								
800	8	M-ALM	39	0	4.2280e−11	2.7940e−10	1.186	4.6732
		2M-ALM	31	0	1.2617e−09	4.1616e−10	1.016	5.2110
		Mid-ALM	35	0	1.3430e−10	3.2041e−10	0.783	5.3471
		2Mid-ALM	28	0	1.2459e−09	3.7932e−10	0.550	5.8262
		ALM	28	0	2.6799e−10	1.3918e−09	5.209	5.9682
1000	10	M-ALM	37	0	9.9716e−11	2.0978e−10	4.134	8.6973
		2M-ALM	34	0	3.6407e−10	7.1350e−11	4.051	10.4427
		Mid-ALM	34	0	6.9301e−11	2.1345e−10	1.231	7.3839
		2Mid-ALM	34	0	5.5128e−11	1.0469e−11	1.294	9.8729
		ALM	34	0	1.8127e−11	6.6256e−11	10.652	12.1485
1500	15	M-ALM	43	148	4.5101e−04	4.5588e−04	4.055	14.3499
		2M-ALM	43	148	4.5101e−04	4.5588e−04	2.545	17.8468
		Mid-ALM	43	148	4.5101e−04	4.5588e−04	2.605	18.1875
		2Mid-ALM	43	148	4.5101e−04	4.5588e−04	2.820	24.2781
		ALM	43	148	4.5101e−04	4.5588e−04	24.687	28.9429
2000	20	M-ALM	40	0	2.6649e−11	2.8989e−10	4.098	19.8100
		2M-ALM	36	0	1.5026e−10	4.3802e−11	4.086	25.0084
		Mid-ALM	40	0	1.3756e−10	2.7348e−10	3.980	29.9102
		2Mid-ALM	36	0	1.3398e−10	3.9088e−11	3.690	35.2204
		ALM	36	0	1.0646e−11	4.9162e−11	52.526	58.8401
2500	25	M-ALM	46	750	6.7984e−04	6.0053e−04	8.377	35.0438
		2M-ALM	43	750	6.7984e−04	6.0053e−04	7.490	45.5847
		Mid-ALM	47	750	6.7984e−04	6.0053e−04	6.806	50.8791
		2Mid-ALM	43	750	6.7984e−04	6.0053e−04	5.820	60.5610
		ALM	43	750	6.7984e−04	6.0053e−04	55.281	76.7875
3000	30	M-ALM	35	0	1.2980e−10	1.7236e−10	9.176	37.2113
		2M-ALM	33	0	2.9048e−10	3.3657e−11	8.847	46.7830
		Mid-ALM	33	0	4.7411e−11	1.6859e−11	5.724	51.2628
		2Mid-ALM	33	0	4.7041e−11	5.7392e−12	5.210	67.5591
		ALM	33	0	9.8886e−11	2.7999e−11	78.992	91.5987
4000	30	M-ALM	35	0	1.1863e−10	1.1171e−10	29.374	83.8732
		2M-ALM	35	0	1.1859e−10	1.1190e−10	30.467	104.1782
		Mid-ALM	35	0	5.7707e−11	1.7954e−11	34.875	126.6837
		2Mid-ALM	35	0	5.7286e−11	5.6062e−12	33.460	175.1587
		ALM	35	0	1.1919e−10	1.6476e−11	249.038	289.4529

Proof. For $\{A_k, E_k, Y_k\}$ generated by Algorithm 2.4, using the same proof as Theorem 3.5, (3.9) and (3.10) are obtained. Since $Y_{k+1} \in \partial(\|\lambda E_{k+1}\|_1)$ and $f(A, E)$ is a convex function, (3.12) and (3.13) hold.

Now, we prove (3.11) for $\{A_k, E_k, Y_k\}$ generated by Algorithm 2.4. Because $\tilde{Y}_{k+1} \in \partial\|A_{k+1}\|_*$, so

$$\langle A_{k+1} - \hat{A}, \tilde{Y}_{k+1} - \hat{Y}\rangle \geq 0.$$

Table 4
Comparison between different algorithms. Corresponding to each triplet $\{n, r\}$, the MR problem was solved for the same data matrix D using the five different algorithms.

n	r	*Algorithm*	*iter*	$\|\hat{E}\|_0 - \|E^\star\|_0$	$\frac{\\|\hat{A}-A^\star\\|_F}{\\|A^\star\\|_F}$	$\frac{\\|\hat{E}-E^\star\\|_F}{\\|E^\star\\|_F}$	t(SVD) (s)	Time (s)
$\|E^\star\|_0 = 0.1n^2$								
800	8	M-ALM	39	0	6.7697e−09	3.0715e−10	1.584	5.1289
		2M-ALM	32	0	1.3412e−09	3.7897e−10	1.394	5.9161
		Mid-ALM	35	0	3.8585e−10	4.0083e−10	0.801	5.2513
		2Mid-ALM	27	0	4.6333e−10	1.3273e−10	0.710	5.8979
		ALM	29	0	4.2293e−10	1.2150e−09	6.443	7.2684
1000	10	M-ALM	37	0	1.4831e−10	3.2610e−10	2.108	6.6389
		2M-ALM	34	0	5.3812e−09	1.2386e−10	1.871	8.0010
		Mid-ALM	35	0	1.3457e−10	2.9676e−10	1.327	7.5656
		2Mid-ALM	34	0	1.9100e−10	7.0285e−11	0.930	9.6740
		ALM	34	0	5.5385e−11	1.0643e−10	10.649	12.2503
1500	15	M-ALM	41	129	4.5101e−04	3.2681e−04	2.495	12.3784
		2M-ALM	40	129	4.5101e−04	3.2681e−04	2.481	16.6004
		Mid-ALM	41	129	4.5101e−04	3.2681e−04	2.403	17.7977
		2Mid-ALM	41	129	4.5101e−04	3.2681e−04	2.750	23.0139
		ALM	41	129	4.5101e−04	3.2681e−04	27.104	31.1201
2000	20	M-ALM	40	0	4.0694e−11	2.9805e−10	4.659	20.1004
		2M-ALM	35	0	3.4458e−10	9.8487e−10	4.446	24.3857
		Mid-ALM	39	0	6.6843e−11	3.9826e−10	4.119	29.3323
		2Mid-ALM	35	0	2.0151e−10	3.0817e−10	3.380	33.3326
		ALM	35	0	1.1183e−10	6.6000e−10	39.578	48.9716
2500	25	M-ALM	43	750	6.7984e−04	4.2986e−04	7.800	32.8044
		2M-ALM	42	750	6.7984e−04	4.2986e−04	5.440	41.2743
		Mid-ALM	43	714	6.7984e−04	4.2986e−04	6.929	47.8549
		2Mid-ALM	32	714	6.7984e−04	4.2986e−04	5.390	58.6226
		ALM	42	690	6.7984e−04	4.2986e−04	60.261	71.5052
3000	30	M-ALM	36	0	1.2014e−10	4.0452e−10	10.265	39.7200
		2M-ALM	36	0	1.2012e−10	3.6632e−10	10.809	52.8253
		Mid-ALM	36	0	4.6491e−11	1.1119e−10	8.943	56.1172
		2Mid-ALM	36	0	4.4280e−11	1.1041e−10	8.640	79.1988
		ALM	36	0	1.1113e−10	3.4770e−10	102.586	116.4265
4000	30	M-ALM	35	−1	1.1598e−10	3.1887e−10	16.489	88.4341
		2M-ALM	35	−1	1.6189e−10	8.8129e−10	17.007	105.1441
		Mid-ALM	36	−1	6.0677e−11	5.9241e−10	15.740	103.0109
		2Mid-ALM	35	−1	6.0748e−11	5.9057e−10	14.190	150.1534
		ALM	36	−1	1.6644e−10	8.9379e−10	200.2100	234.379

Thus,

$$\langle A_{k+1} - \hat{A}, \hat{Y}_{k+1} - \hat{Y} \rangle \geq \langle A_{k+1} - \hat{A}, \tilde{Y}_{k+1} - \hat{Y} \rangle \geq 0,$$

(3.11) and Lemma 4 in [8] hold. Using the same proof of Theorem 2 in [8], we obtain Theorem 3.7.

Theorem 3.8. *Let (A_k, E_k) be the matrix sequences generated by Algorithm 2.5, and $(\hat{A}, \hat{E})$ be the optimal solution of (2.1). Suppose that $\mu_k \to \infty$ and $\sum_{k=1}^{\infty} \mu_k^{-1} = \infty$. Let $\hat{Y}_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_k)$ and $\tilde{\tilde{Y}}_{k+1} = Mid(Y_k + \mu_k(D - E_k - A_{k+1}))$. If*

$$\langle A_{k+1} - \hat{A}, \hat{Y}_{k+1} - \hat{Y} \rangle \geq \langle A_{k+1} - \hat{A}, \tilde{\tilde{Y}}_{k+1} - \hat{Y} \rangle, \tag{3.18}$$

then

$$\lim_{k\to\infty} A_k = \hat{A}, \qquad \lim_{k\to\infty} E_k = \hat{E}. \tag{3.19}$$

Proof. For $\{A_k, E_k, Y_k\}$ generated by Algorithm 2.5, using the same proof as Theorem 3.5, (3.9) and (3.10) are obtained. Since $Y_{k+1} \in \partial(\|\lambda E_{k+1}\|_1)$ and $f(A, E)$ is a convex function, (3.12) and (3.13) hold.

Now, we prove (3.11) for $\{A_k, E_k, Y_k\}$ generated by Algorithm 2.5. Because $\tilde{\tilde{Y}}_{k+1} \in \partial\|A_{k+1}\|_*$, so

$$\langle A_{k+1} - \hat{A}, \tilde{\tilde{Y}}_{k+1} - \hat{Y} \rangle \geq 0.$$

Thus,

$$\langle A_{k+1} - \hat{A}, \hat{Y}_{k+1} - \hat{Y} \rangle \geq \langle A_{k+1} - \hat{A}, \tilde{\tilde{Y}}_{k+1} - \hat{Y} \rangle \geq 0,$$

(3.11) and Lemma 4 in [8] hold. Using the same proof of Theorem 2 in [8], the theorem is obtained.

Table 5
Comparison between different algorithms. Corresponding to each triplet $\{n, r\}$, the MR problem was solved for the same data matrix D using the five different algorithms.

n	r	*Algorithm*	*iter*	$\|\hat{E}\|_0 - \|E^\star\|_0$	$\frac{\|\hat{A}-A^\star\|_F}{\|A^\star\|_F}$	$\frac{\|\hat{E}-E^\star\|_F}{\|E^\star\|_F}$	t(SVD) (s)	Time (s)
$\|E^\star\|_0 = 0.3n^2$								
800	8	M-ALM	37	0	3.7576e−10	5.3844e−10	1.713	4.9740
		2M-ALM	35	0	8.8876e−10	1.5195e−10	1.589	6.3912
		Mid-ALM	35	0	2.4670e−10	4.3492e−10	0.560	5.0827
		2Mid-ALM	35	0	2.4185e−10	6.0909e−11	0.510	7.4134
		ALM	35	0	2.6075e−10	1.6425e−10	11.512	12.6865
1000	10	M-ALM	36	0	4.5305e−10	4.5195e−10	2.078	6.6863
		2M-ALM	34	0	1.0624e−09	1.1950e−10	2.042	8.4681
		Mid-ALM	34	0	3.2199e−10	3.8999e−10	0.989	7.2539
		2Mid-ALM	34	0	3.1601e−10	3.5288e−11	1.120	10.0302
		ALM	34	0	2.7091e−10	1.1350e−10	21.443	23.1530
1500	15	M-ALM	38	−82	4.5101e−04	1.1429e−04	2.734	11.7930
		2M-ALM	38	−82	4.5101e−04	1.1429e−04	2.664	16.1076
		Mid-ALM	38	−82	4.5101e−04	1.1429e−04	1.591	15.6780
		2Mid-ALM	38	−82	4.5101e−04	1.1429e−04	1.560	22.9721
		ALM	38	−82	4.5101e−04	1.1429e−04	60.524	64.7674
2000	20	M-ALM	38	0	1.8501e−10	4.6030e−10	3.508	18.5424
		2M-ALM	35	0	6.6755e−10	9.3934e−10	3.279	23.6172
		Mid-ALM	37	0	1.4779e−10	5.2474e−10	2.715	26.7548
		2Mid-ALM	35	0	2.0417e−10	9.7757e−10	2.280	34.2279
		ALM	35	0	5.0140e−10	1.0400e−09	122.992	129.9311
2500	25	M-ALM	40	−484	6.7984e−04	1.5057e−04	6.366	29.5111
		2M-ALM	39	−444	6.7984e−04	1.5057e−04	5.991	39.4548
		Mid-ALM	40	−486	6.7984e−04	1.5057e−04	3.042	41.3720
		2Mid-ALM	39	−468	6.7984e−04	1.5057e−04	2.980	59.1736
		ALM	39	−464	6.7984e−04	1.5057e−04	187.579	200.5668
3000	30	M-ALM	35	−2	3.3153e−10	8.4264e−10	11.029	39.3535
		2M-ALM	35	−2	3.3116e−10	8.1038e−10	10.841	52.5110
		Mid-ALM	35	−2	8.3701e−11	8.2644e−10	3.130	52.5106
		2Mid-ALM	35	−2	8.5389e−11	8.2436e−10	2.980	73.3612
		ALM	35	−2	3.6460e−10	8.2504e−10	227.810	242.9331
4000	30	M-ALM	33	−2	5.5756e−10	5.4921e−10	22.197	83.6595
		2M-ALM	33	−2	5.5685e−10	4.4799e−10	20.010	101.7431
		Mid-ALM	33	−4	1.4338e−09	4.3309e−10	11.190	107.9956
		2Mid-ALM	33	−2	1.0151e−10	4.2587e−10	13.820	138.9718
		ALM	33	−13	3.4651e−10	4.9249e−10	560.359	597.7770

4. Numerical experiments

In this section, we compare our algorithms with the ALM algorithm presented in [8] and the SVT algorithm [20] through numerical experiments, and all the experiments are conducted on the same workstation.

We denote the true solution by the ordered pair $(A^\star, E^\star)$, and the output by $(\hat{A}, \hat{E})$. In order to generate a rank-r Toeplitz matrix $A^\star$, we generate a Toeplitz matrix A through Matlab command; Firstly, compute the first r singular values of the Toeplitz matrix A using the Lanczos method, thus we obtain $\hat{A} = U \begin{pmatrix} \Sigma_r & \\ & 0 \end{pmatrix} V^*$ ($\hat{A}$ is not a Toeplitz matrix); Secondly, we compute $a_l = \text{mean}(\text{diag}(\hat{A}, l))$, $l \in \Omega$, and set $A = \sum_{l \in \Omega} a_l R_l$, obviously, A is a Toeplitz matrix. Repeat the above steps until the rank of the Toeplitz matrix A is equal to r. We generate $E^\star$ as a sparse matrix whose entries are chosen uniformly at random. The input to the algorithms is the matrix $D = A^\star + E^\star$. We choose a fixed weighting parameter $\lambda = \frac{1}{\sqrt{n}}$.

For Algorithm 2.1, we empirically set $\tau_0 = 0.5\|D\|_2$, $c = 0.8$, $\epsilon = 10^{-9}$. For Algorithms 2.2–2.5, we set $Y_0 = \frac{D}{J(D)}$, $(J(D) = \max(\|D\|_2, \lambda^{-1}\|D\|_\infty)$ [8]), $\mu_0 = 1.25/\|D\|_2$, $\rho = 1.5$, $\epsilon_1 = 10^{-9}$, $\epsilon_2 = 10^{-6}$.

The brief comparisons of the algorithms are presented in Tables 1–6, where t(SVD) denotes the time of SVD. The comparison of CPU time between different algorithms is shown in Fig. 1, where the only difference between subfigure (f) and subfigure (g) is the scale of ordinate. Subfigure (h) describes a multiple relationship between the CPU time of the original ALM algorithm and the CPU time of the four modified ALM algorithms, where $T1 = \frac{C_{ALM}}{C_{M-ALM}}$, $T2 = \frac{C_{ALM}}{C_{2M-ALM}}$, $T3 = \frac{C_{ALM}}{C_{Mid-ALM}}$, $T4 = \frac{C_{ALM}}{C_{2Mid-ALM}}$, where C_{ALM} denotes the CPU time of ALM algorithm.

From Tables 1–2 and subfigures (a)–(b), we can see that the MV algorithm and SVT algorithm achieve almost the same iterations and relative errors. However, our algorithm is advantageous over the SVT algorithms on the time of SVD, as well as the CPU. The superiority of the MV algorithm is more significant for large-scale Toeplitz matrices.

From Tables 3–6 and subfigures (c)–(h), we can see that the four modified ALM algorithms and ALM algorithm have almost the same accuracies. Although there is a little difference in iterations among the five algorithms, the time of the SVD,

Table 6
Comparison between different algorithms. Corresponding to each triplet $\{n, r\}$, the MR problem was solved for the same data matrix D using the five different algorithms.

n	r	*Algorithm*	*iter*	$\lvert\hat{E}\rvert_0 - \lvert E^\star\rvert_0$	$\frac{\lVert\hat{A}-A^\star\rVert_F}{\lVert A^\star\rVert_F}$	$\frac{\lVert\hat{E}-E^\star\rVert_F}{\lVert E^\star\rVert_F}$	t(SVD) (s)	Time (s)
$\lvert E^\star\rvert_0 = 0.5n^2$								
800	8	M-ALM	37	−1	3.6426e−10	8.5245e−10	1.732	5.0813
		2M-ALM	33	−1	2.0200e−09	7.2703e−10	1.433	5.9110
		Mid-ALM	36	−1	5.0555e−10	8.2443e−10	1.375	6.6085
		2Mid-ALM	33	−1	1.5097e−09	6.9301e−10	1.184	7.4246
		ALM	33	−1	1.8049e−09	9.6901e−10	134.729	135.8247
1000	10	M-ALM	36	0	7.0904e−10	6.7826e−10	3.122	7.6581
		2M-ALM	33	0	2.4897e−09	2.7718e−10	2.667	8.9317
		Mid-ALM	33	0	7.8893e−10	7.4926e−10	1.018	7.0023
		2Mid-ALM	33	0	7.1747e−10	8.9273e−11	1.080	10.2788
		ALM	33	0	2.1807e−09	9.5415e−10	232.149	234.8623
1500	15	M-ALM	38	69	4.5101e−04	9.2784e−05	2.633	11.7973
		2M-ALM	38	69	4.5101e−04	9.2784e−05	2.671	16.2109
		Mid-ALM	38	69	4.5101e−04	9.2784e−05	1.579	15.8591
		2Mid-ALM	38	69	4.5101e−04	9.2784e−05	1.650	22.9693
		ALM	38	69	4.5101e−04	9.2784e−05	934.773	940.0092
2000	20	M-ALM	37	0	3.8257e−10	7.5226e−10	4.462	18.9546
		2M-ALM	36	0	5.8503e−10	4.8562e−10	4.306	24.9112
		Mid-ALM	37	0	2.3841e−10	5.5914e−10	2.806	26.6667
		2Mid-ALM	36	−1	3.4652e−10	5.7873e−10	2.730	36.5719
		ALM	36	0	9.6887e−10	5.5528e−10	2285.344	2.2952e+03
2500	25	M-ALM	39	412	6.7984e−04	1.2207e−04	4.649	27.2414
		2M-ALM	39	410	6.7984e−04	1.2207e−04	4.590	38.3527
		Mid-ALM	39	412	6.7984e−04	1.2207e−04	2.917	40.5175
		2Mid-ALM	39	410	6.7984e−04	1.2207e−04	2.980	55.7794
		ALM	39	412	6.7984e−04	1.2207e−04	4975.522	4.9923e+03
3000	30	M-ALM	35	−2	4.4157e−10	5.0540e−10	11.783	39.8834
		2M-ALM	35	−2	4.4302e−10	4.0415e−10	11.965	57.0603
		Mid-ALM	35	−2	1.1013e−10	4.0905e−10	3.030	52.2024
		2Mid-ALM	35	−2	1.0988e−10	4.0314e−10	3.310	74.5576
		ALM	34	−2	1.1120e−09	8.2871e−10	8078.607	8.0995e+03
4000	30	M-ALM	34	−7	5.3936e−10	6.9716e−10	21.450	95.0662
		2M-ALM	34	−7	5.3790e−10	6.2958e−10	20.900	114.0800
		Mid-ALM	34	−7	8.7365e−11	6.2854e−10	20.220	142.3991
		2Mid-ALM	34	−7	8.8196e−11	6.2509e−10	20.560	172.3583
		ALM	34	−7	9.1543e−10	2.7638e−10	2.7051e+4	2.7104e+4

as well as the CPU, in the four modified ALM algorithms is significantly less than the ALM algorithm. With the increasing of matrix dimension or the number of nonzero elements of the matrix E, the advantage of the fast SVD of Toeplitz matrices is much more apparent, which contributes to reducing CPU time.

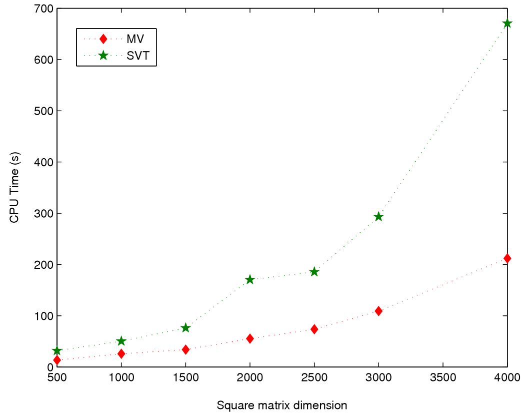
Therefore, the five new algorithms are not only effective but also efficient, which are promising for practical use.

5. Conclusion

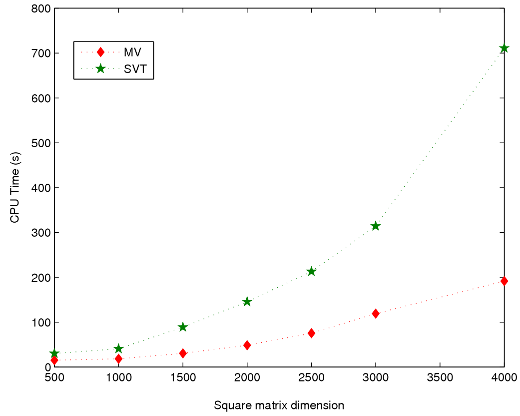
In this paper, according to the special structure of Toeplitz matrices, we have presented five different algorithms for Toeplitz matrix recovery based on the SVT algorithm and ALM algorithm respectively. Throughout the iterative process, the iterative matrices of all the algorithms keep the Toeplitz structure that ensures the fast SVD of Toeplitz matrices, which helps to reduce the CPU time. Thus, all the new algorithms are faster than the original algorithms, and much more efficient for large-scale matrices recovery problem with higher density of E. By further comparison, it is easy to see that the four modified ALM algorithms are at least 2 times faster than the MV algorithm. Moreover, a further analysis of the four modified ALM algorithms indicates that the 2M-ALM algorithm has the following advantages: (a) a better convergence analysis (see Theorem 3.5); (b) less iteration. However, the algorithm is a little longer than the M-ALM algorithm in CPU time because of one more mean-value modification.

Acknowledgments

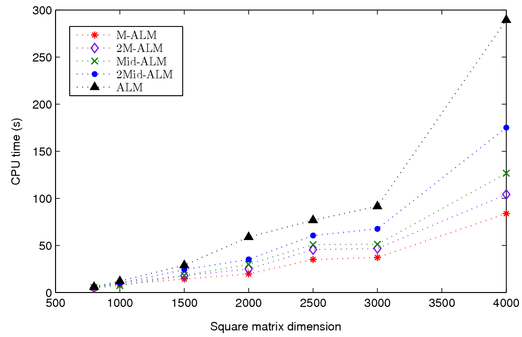
This work was supported by National Natural Science Foundation of China (Grant No. 11371275). The authors are very much indebted to the anonymous referees for their helpful comments and suggestions which greatly improved the original manuscript.



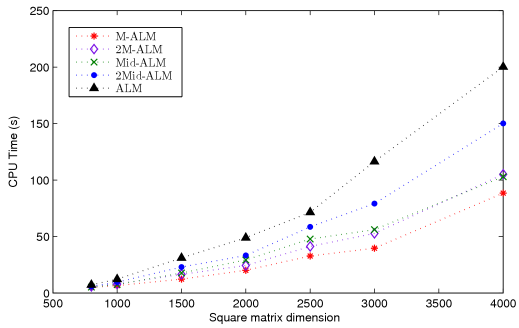
(a) $|E^{\star}|_0 = 0.05n^2$.



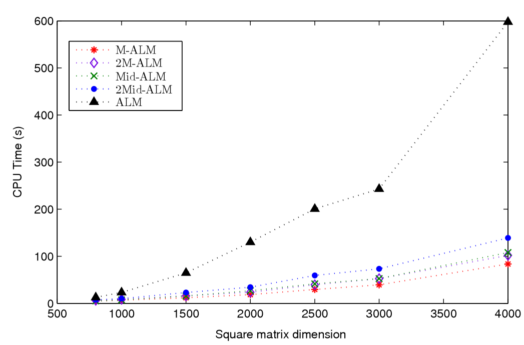
(b) $|E^{\star}|_0 = 0.1n^2$.



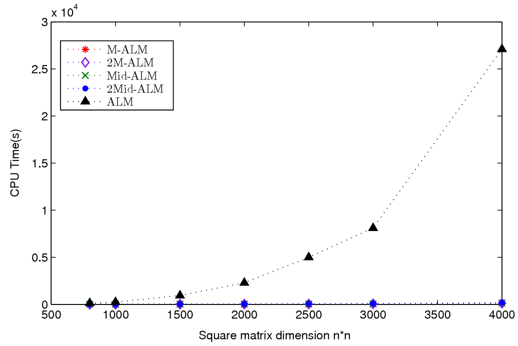
(c) $|E^{\star}|_0 = 0.05n^2$.



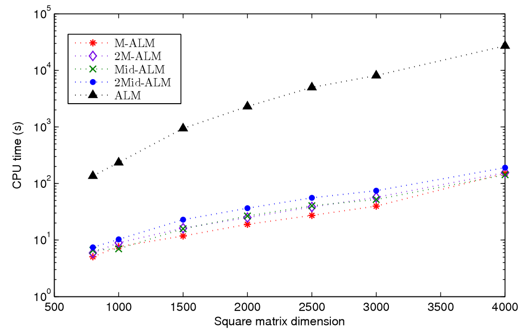
(d) $|E^{\star}|_0 = 0.1n^2$.



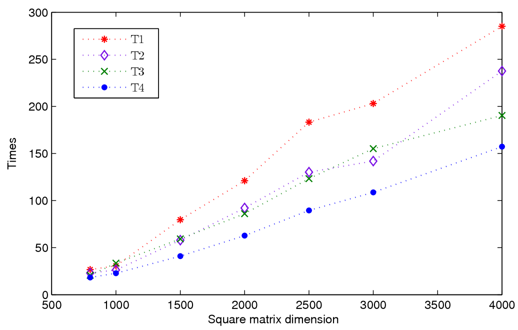
(e) $|E^{\star}|_0 = 0.3n^2$.



(f) $|E^{\star}|_0 = 0.5n^2$.



(g) $|E^{\star}|_0 = 0.5n^2$.



(h) $|E^{\star}|_0 = 0.5n^2$.

Fig. 1. Comparison of CPU time between different algorithms.

References

[1] Y. Peng, A. Ganesh, J. Wright, W. Xu, Y. Ma, Rasl: robust alignment by sparse and low-rank decomposition for linearly correlated images, in: Computer Vision and Pattern Recognition, CVPR, IEEE Conference on, Jun. 2010, pp. 763–770. http://dx.doi.org/10.1109/CVPR.2010.5540138.
[2] W.T. Tan, G. Cheung, Y. Ma, Face recovery in conference video streaming using robust principal component analysis, in: Image Processing, ICIP, 18th IEEE International Conference on, Sep. 2011, pp. 3225–3228. http://dx.doi.org/10.1109/ICIP.2011.6116356.
[3] Z. Zhang, X. Liang, Y. Ma, Unwrapping low-rank textures on generalized cylindrical surfaces, in: Computer Vision, ICCV, IEEE International Conference on, Nov. 2011, pp. 1347–1354. http://dx.doi.org/10.1109/ICCV.2011.6126388.
[4] J. Wright, A. Ganesh, S. Rao, Y. Ma, Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization, in: Proceedings of the Conference on Neural Information Processing Systems, NIPS, 2009.
[5] E.J. Candès, X. Li, Y. Ma, J. Wright, Robust principal component analysis? J. ACM 58 (2011) http://dx.doi.org/10.1145/1970392.1970395.
[6] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, Y. Ma, Fast convex optimization algorithms for exact recovery of a corrupted low rank matrix, University of Illinois Technical report UILU-ENG-09-2214, 2009.
[7] A. Ganesh, Z. Lin, J. Wright, L. Wu, M. Chen, L. Ma, Fast algorithms for recovering a corrupted low-rank matrix, in: Computational Advances in Multi-Sensor Adaptive Processing, CAMSAP, 3rd IEEE International Workshop on, Dec. 2009, pp. 213–216. http://dx.doi.org/10.1109/CAMSAP.2009.5413299.
[8] Z. Lin, M. Chen, L. Wu, Y. Ma, The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices, UIUC Technicial Report UIUL-ENG-09-2214, 2010.
[9] W.U. Bajwa, J.D. Haupt, G.M. Raz, S.J. Wright, R.D. Nowak, Toeplitz-structures compressed sensing matrices, in: Statistical signal Processing, 2007, SSP'07. IEEE/SP 14th Workshop on, Aug. 2007, pp. 294–298. http://dx.doi.org/10.1109/SSP.2007.4301266.
[10] F. Sebert, Y.M. Zhou, L. Ying, Toeplitz block matrices in compressed sensing and their applications in imaging, in: Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on, May 2008, pp. 47–50. http://dx.doi.org/10.1109/ITAB.2008.4570587.
[11] J. Gutiérrez-Gutiérrez, P.M. Crespo, A. Böttcher, Functions of banded Hermitian block Toeplitz matrices in signal processing, Linear Algebra Appl. 422 (2007) 788–807. http://dx.doi.org/10.1016/j.laa.2006.12.008.
[12] J. Huang, T.Z. Huang, X.L. Zhao, Z.B. Xu, Image restoration with shifting reflective boundary conditions, Sci. China Inf. Sci. 56 (2013) 1–15. http://dx.doi.org/10.1007/s11432-011-4425-2.
[13] A.K. Shaw, S. Pokala, R. Kumaresan, Toeplitz and Hankel matrix approximation using structured approach, in: Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on, vol. 4, 1998, pp. 2349–2352. http://dx.doi.org/10.1109/ICASSP.1998.681621.
[14] T. Kailath, A.H. Sayed (Eds.), Fast Reliable Algorithms for Matrices with Structure, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.
[15] F.T. Luk, S. Qiao, A fast singular value algorithm for Hankel matrices, in: V. Olshevsky (Ed.), Fast Algorithms for Structured Matrices: Theory and Applications, in: Contemporary Mathematics, vol. 323, American Mathematical Society, 2003, pp. 169–177.
[16] W. Xu, S. Qiao, A fast symmetric SVD algorithm for square Hankel matrices, Linear Algebra Appl. 428 (2008) 550–563. http://dx.doi.org/10.1016/j.laa.2007.05.027.
[17] G.H. Golub, C.F. VanLoan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[18] C.F. VanLoan, Computational Frameworks for the Fast Fourier Transform, Society for Industrial and Applied Mathematics (SIAM), 1992.
[19] C.L. Wang, C. Li, A mean value algorithm for Toeplitz matrix completion, Appl. Math. Lett. 41 (2015) 35–40. http://dx.doi.org/10.1016/j.aml.2014.10.013.
[20] J.F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (2010) 1956–1982. http://dx.doi.org/10.1137/080738970.