Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 8812701, 14 pages
https://doi.org/10.1155/2020/8812701

Research Article

A Singular Value Thresholding with Diagonal-Update Algorithm for Low-Rank Matrix Completion

Yong-Hong Duan,[1] Rui-Ping Wen ,[2] and Yun Xiao[2]

[1]*Department of Applied Mathematics, Taiyuan University, Taiyuan 030600, China*
[2]*Key Laboratory for Engineering and Computational Science, Shanxi Provincial Department of Education, Taiyuan Normal University, Jinzhong 030619, Shanxi Province, China*

Correspondence should be addressed to Rui-Ping Wen; wenrp@163.com

Received 31 August 2020; Revised 14 November 2020; Accepted 10 December 2020; Published 24 December 2020

Academic Editor: He Chen

The singular value thresholding (SVT) algorithm plays an important role in the well-known matrix reconstruction problem, and it has many applications in computer vision and recommendation systems. In this paper, an SVT with diagonal-update (D-SVT) algorithm was put forward, which allows the algorithm to make use of simple arithmetic operation and keep the computational cost of each iteration low. The low-rank matrix would be reconstructed well. The convergence of the new algorithm was discussed in detail. Finally, the numerical experiments show the effectiveness of the new algorithm for low-rank matrix completion.

1. Introduction

The problem of completing low-rank and sparse matrices from some of its observed entries occurs frequently in many areas of engineering and applied science such as machine learning [1, 2], model reduction [3], compressed sensing [4], control [5], pattern recognition [6], signal and imaging inpainting [7–10], and computer vision [11]. From the pioneering work on low-rank approximation by Fazel [12] as well as on matrix completion by Candès and Recht [13], there has been a lot of study (see [1–35] and references therein) both from theoretical and algorithmic aspects on the problem of recovering a low-rank matrix from partial entries, also known as matrix completion. There is a rapidly growing interest for this issue. Explicitly seeking the lowest rank matrix consistent with the known entries is mathematically expressed as

$$\begin{aligned} &\min_{X\in\mathfrak{R}^{m\times n}} \operatorname{rank}(X),\\ &\text{s.t.}\quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{1}$$

where the matrix $M \in \mathfrak{R}^{m\times n}$ is the underlying matrix to be reconstructed and $\mathcal{P}_\Omega$ is the associated sampling orthogonal projection operator which acquires only the entries indexed by $\Omega \subset \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ with Ω is a random subset of indices for the known entries.

The general problem (1), however, is nonconvex and is NP hard [14] due to the rank objective. There are a few of algorithms for solving this model directly. Alternatively, Candès and Rechat [13] switched (1) to another simple convex optimization problem (2) as follows:

$$\begin{aligned} &\min_{X\in\mathfrak{R}^{m\times n}} \|X\|_*,\\ &s.t.\quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{2}$$

where the nuclear norm $\|X\|_*$ is the sum of all singular values of the matrix X.

Furthermore, it has proved that the sequence $\{X^k\}$ converges to the unique solution of the following optimization problem closely related to (2) in [15]:

$$\begin{aligned} &\min_{X\in\mathfrak{R}^{m\times n}} \tau\|X\|_* + \frac{1}{2}\|X\|_F^2.\\ &\text{s.t.}\quad \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M). \end{aligned} \tag{3}$$

As for the solution of problems (2) and (3), there have been many computational efficient algorithms which are designed for a broad class of matrices such as mainly the accelerated proximal gradient (APG) algorithm [16], the augmented Lagrange multiplier (ALM) algorithm [17], several methods [18–21] resulted in alternating optimization based on the bilinear factorization $M = XY$ with $X \in \mathfrak{R}^{m\times r}, Y \in \mathfrak{R}^{r\times n}$ and rank $(M) = r$, and the singular value theresholding (SVT) algorithm, as well as its improvements [15, 22–24]. However, the computations of partial singular value decomposition (SVD) were required at each iteration in the most direct implementation of these algorithms. The computational cost of computing the SVD has complexity of $O(n^3)$ when the rank r and matrix-size n are proportional, resulting in computing the SVD to be the dominant computational cost at each iteration and then limits their applicability for large n. In view of its outstanding performance and elegant mathematical properties, the SVT algorithm obtains widespread attention [25–27]. The variants and extended applications of the SVT algorithm have been studied later: Candès et al. [28] presented a unbiased risk estimate formula of the SVT for the noisy observations; Chatterjee [29] studied a general method for matrix denoising using SVT, which covers the stochastic block model as a special case; Donoho and Gavish [30] pointed out several ways that these matrix denoising results for singular value soft thresholding (SVST) estimation of low-rank matrices parallel results for soft thresholding of sparse vectors; Dutta et al. [31] proposed an alternative solution to the sensitivity of the classical principal component analysis (PCAs) to the outliers for solving the problem

$$\min_{X\in\mathfrak{R}^{m\times n}} \left\{\frac{1}{2}\|(M - X)W\|_F^2 + \tau\|X\|_*\right\}, \tag{4}$$

with the nonsingular weight matrix $W \in \mathfrak{R}^{n\times m}$ which is user provided or automatically inferred from the data, called as WSVT problem; Klopp [32] introduced a variant of the SVT iteration; Ma and Xu [33] recovered the received signal strength (RSS) reading and achieve good localization performance based on the SVT theory; Zhang et al. [34] put forward a lower bound guaranteeing exact matrix completion via the SVT algorithm; Zhang et al. [35] have considered the low-rank tensor completion problem through a hybrid singular value thresholding scheme.

This paper develops a modification of the SVT algorithm for approximately solving the nuclear norm minimization problem (2). By using a diagonal-update technique for the approximated sequence at each step, the iteration matrices generated by the new algorithm are approximated to the true solution well, which saves significant computational cost of the SVT algorithm performance. And we also establish the convergence theory in detail. Experimental results show that the new algorithm outperforms the standard SVT and its several variants algorithms, especially when problem (2) comes to large-scale issues.

The rest of the paper is organized as follows. After we provide some notations in this section, we review briefly the standard SVT algorithm, the accelerated singular value thresholding (ASVT) algorithm as well as its modification, and a modified algorithm of the SVT algorithm with diagonal-update (D-SVT) is proposed in Section 2. In Section 3, the convergence theory of the new algorithm is established. Then, numerical experiments are shown and compared in Section 4. Finally, we end the paper with the concluding remarks in Section 5.

Here are some notations. $\mathfrak{R}^{m\times n}$ denotes $m\times n$ real matrices set, $\mathfrak{R}_+^{m\times n}$ represents the $m\times n$ nonnegative and real matrices' set. The nuclear norm $\|X\|_*$ of a matrix X is defined by $\|X\|_* = \sum_{k=1}^r \sigma_k(X)$, $\sigma_k(X)$ denotes the kth largest singular value of the real matrix $X \in \mathfrak{R}^{m\times n}$ of rank r, and the Frobenius norm $\|X\|_F$ of a matrix $X = (x_{ij}) \in \mathfrak{R}^{m\times n}$ is $\|X\|_F = (\sum_{i=1}^m \sum_{j=1}^n |x_{ij}|^2)^{1/2} = \sqrt{\text{trace}(A^T A)}$. X^T is the transpose of a matrix X. $\langle X, Y\rangle = \text{trace}(X^T Y)$ denotes the inner product between two matrices. $\Omega \subset \{1,2,\ldots,m\} \times \{1,2,\ldots,n\}$ is the indices of the observed entries, and $\overline{\Omega}$ is the complementary set of Ω. $\mathcal{P}_\Omega$ is the orthogonal projector onto Ω, satisfying,

$$\mathcal{P}_\Omega(X) = \begin{cases} X_{ij}, & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega. \end{cases} \tag{5}$$

2. Related Algorithms

In order to completing comparison subsequently, we briefly review and introduce some algorithms for solving the matrix completion problem (2).

2.1. The Standard Singular Value Thresholding (SVT) Algorithm

Definition 1. The singular value decomposition (SVD) of a matrix $X \in \mathfrak{R}^{m\times n}$ of rank r is

$$\begin{aligned} X &= U\Sigma_r V^T, \\ \Sigma_r &= diag(\sigma_1,\ldots,\sigma_r), \end{aligned} \tag{6}$$

where $U \in \mathfrak{R}^{m\times r}$ and $V \in \mathfrak{R}^{n\times r}$ are two orthogonal matrices, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

Definition 2 (see [15]). For each $\tau \geq 0$, the singular value thresholding operator $\mathcal{D}_\tau(X)$ is defined as follows, say the "shrinkage":

$$\begin{aligned} \mathcal{D}_\tau(X) &:= U\mathcal{D}_\tau(\Sigma_r)V^T, \\ \mathcal{D}_\tau(\Sigma_r) &= \text{diag}(\{\sigma_i - \tau\}_+), \end{aligned} \tag{7}$$

where $\{\sigma_i - \tau\}_+ = \begin{cases} \sigma_i - \tau, & \text{if } \sigma_i > \tau, \\ 0, & \text{if } \sigma_i \leq \tau. \end{cases}$

The standard SVT algorithm proposed in [15] is a solution for solving the convex optimization (2).

Remark 1. Due to the ability of producing low-rank solutions with the soft-thresholding operator, the SVT algorithm was shown to be extremely efficient at addressing problems with low-rank optimal solutions such as recommender systems.

Input: sampled set Ω and sampled entries $\mathcal{P}_\Omega(M)$, step size $\delta \in (0,2)$, tolerance ε, parameter $\tau > 0$, increment ℓ, and maximum iteration count $k_{\max}$
Output: X^{opt}
Description: recover a low-rank matrix M from a subset of sampled entries
(1) Set $Y^0 = k_0 \delta \mathcal{P}_\Omega(M)$, k_0 is an integer with $(\tau/\delta\|\mathcal{P}_\Omega(M)\|_2] \in (k_0 - 1, k_0)$
(2) Set $r_0 = 0$
(3) **for** $k = 1$ to $k_{\max}$
(4) Set $s_k = r_{k-1} + 1$
(5) **repeat**
(6) Compute $[U^{k-1}, \Sigma^{k-1}, V^{k-1}]_{s_k}$
(7) Set $s_k = s_k + \ell$
(8) **until** $\sigma^{k-1}_{s_k - \ell} \le \tau$
(9) Set $r_k = \max\{j : \sigma^{k-1}_j > \tau\}$
(10) Set $X^k = \sum_{j=1}^{r_k} (\sigma^{k-1}_j - \tau) u^{k-1}_j v^{k-1}_j$
(11) **if** $\|\mathcal{P}_\Omega(X^k - M)\|_F < \varepsilon \|\mathcal{P}_\Omega(M)\|_F$ **then break**
(12) Set $Y^k_{ij} = \begin{cases} Y^{k-1}_{ij} + \delta(M_{ij} - X^k_{ij}), & \text{if } (i,j) \in \Omega, \\ 0, & \text{if } (i,j) \notin \Omega \end{cases}$
(13) **end** *for k*
(14) Set $X^{\text{opt}} = X^k$

Algorithm 1: Singular value thresholding (SVT) algorithm, algorithm 1 of [15].

2.2. The Accelerated Singular Value Thresholding (ASVT) Algorithm. Introduce the Lagrangian function of problem (3) as

$$\mathcal{L}(X,Y) = \tau\|X\|_* + \langle Y, \mathcal{P}_\Omega(M - X)\rangle + \frac{1}{2}\|X\|_F^2, \quad (8)$$

where Y is the Lagrangian variable.

In terms of the dual approach, $f(Y) = \inf_X \mathcal{L}(X,Y)$ is the dual function of $\mathcal{L}(X,Y)$, which is concave. Define

$$h(Y) = -f(Y) = -\Big(\tau\big\|\mathcal{D}_\tau(\mathcal{P}_\Omega(Y))\big\|_* + \langle Y, \mathcal{P}_\Omega(M - \mathcal{D}_\tau(\mathcal{P}_\Omega(Y)))\rangle + \frac{1}{2}\big\|\mathcal{D}_\tau(\mathcal{P}_\Omega(Y))\big\|_F^2\Big), \quad (9)$$

which is convex. Thus, we can solve problem (3) by firstly minimizing the objective function $h(Y)$, namely,

$$\min_{Y \in \mathfrak{R}^{m\times n}} h(Y). \quad (10)$$

Problem (10) was computed via Nesterov's method with an adaptive line search scheme. Then, Algorithm 2 has been provided.

Based on the above, the accelerated singular value thresholding (ASVT) algorithm has been proposed in [22]. Furthermore, Wang et at. [23] presented the Ne-SVT by replacing the adaptive linear search with Nemirovski's technique and the M-ASVT algorithms by using the same search technique in the ASVT algorithm. The overall steps of the later can be organized as Algorithm 3.

It is reported that M-ASVT needs much fewer iterations than ASVT algorithm under the same level of accuracy and the same cost of computing.

2.3. The Singular Value Thresholding with Diagonal-Update (D-SVT) Algorithm. We are now in the position to introduce a modified singular value thresholding algorithm by using a diagonal-update technique, as shown in Algorithm 4.

Set $D{:} = \mathcal{P}_\Omega(\widetilde{X}^k)$ for short. The difference with the standard SVT algorithm may seem at k th step, replacing the iteration matrix $\widetilde{X}^k$ with the diagonal-update DW_k of its projector D, where $W_k = \operatorname{diag}(w^{(k)}_{11}, w^{(k)}_{22}, \ldots, w^{(k)}_{nn}) \in \mathfrak{R}^{n\times n}_+$ is obtained by

$$W_k = \arg\min_W \big\|\mathcal{P}_\Omega(M) - DW\big\|_F. \quad (11)$$

Equation (11) is easy to compute since it is so simple just some arithmetic operation required, without extra cost. In fact, the exact solution of (11) is given by

$$w^{(k)}_{jj} = \frac{\langle M(:,j), X_k(:,j)\rangle}{\langle X_k(:,j), X_k(:,j)\rangle}, \quad j = 1, 2, \ldots, n. \quad (12)$$

Remark 2. The sequence matrices generated by the new algorithm are approximated to the true solution well, which saves significant computational cost of the SVT algorithm performance, without actually extra complexity. It is designed as Algorithm 4 by plugging some steps into the SVT method. It should be seen that Algorithm 4 has three lines (as shown in lines 10–12) more than Algorithm 1. The new algorithm includes Algorithm 1 as special case when $W = I$.

3. Convergence Analysis

In this section, the convergence theory is discussed for the singular value thresholding which involves diagonal-update algorithm.

Theorem 1. *Suppose that* $\left\{\mathcal{P}_\Omega(\widetilde{X}^k)(I - W_k)\right\}_{k=1}^{\infty}$ *is uniformly bounded. Then,*

Input: $\widetilde{\mu}, \alpha_{-1} = 0.5, Y_{-1} = Y_0, L_{-1} = L_0, \gamma_0 \geq \widetilde{\mu}, \lambda_0 = 1$
Output: Y^N
(1) **for** $k = 0, 1$ to N **do**
(2) **while** 1 **do**
(3) Compute $\alpha_k \in (0,1)$ as the root of $L_k \alpha_k^2 = (1-\alpha_k)\gamma_k + \alpha_k \widetilde{\mu}$
(4) Compute $\gamma_{k+1} = (1-\alpha_k)\gamma_k + \alpha_k \widetilde{\mu}, \beta_k = (\gamma_k (1-\alpha_{k-1})/\alpha_{k-1}(\gamma_k + L_k \alpha_k))$
(5) Compute $S_k = Y_k + \beta_k (Y_k - Y_{k-1})$
(6) Compute $Y_{k+1} = S_k - (1/L_k) h'(S_k)$
(7) **if** $h(Y_{k+1}) \leq h(S_k) - (1/2L_k)\|h'(S_k)\|_F^2$ **then**
(8) go to Line 13
(9) **else**
(10) $L_k = 2L_k$
(11) **end if**
(12) **end while**
(13) Set $\omega = 2L_k (h(S_k) - h(Y_{k+1})/\|h'(S_k)\|_F^2), L_{k+1} = h(\omega) L_k, h(\omega) = \begin{cases} 1, & \text{if } 1 \leq \omega \leq 5, \\ 0.8, & \text{if } \omega > 5 \end{cases}$
(14) Set $\lambda_{k+1} = (1-\alpha_k)\lambda_k$
(15) **end for**

Algorithm 2: The adaptive linear search scheme, algorithm 1 of [22].

Input: $L_1, t_{-1} = 0, t_0 = 1, Y_1 = Y_0 = 0, \eta \in (0,1)$
Output: $Y_{N+1}, X_{N+1} = \mathcal{D}_\tau(\mathcal{P}_\Omega(Y_{N+1}))$
Description: recover a low-rank matrix Y
(1) **for** $k = 1, 2, \ldots, N$ **do**
(2) Compute $k = 1, 2, \ldots, N$
(3) Compute $S_k = Y_k + \beta_k (Y_k - Y_{k-1})$
(4) **while** 1 **do**
(5) Compute $Y_{k+1} = S_k - (1/L_k) h'(S_k)$
(6) **if** $h(Y_{k+1}) \leq h(S_k) - (1/2L_k)\|h'(S_k)\|_F^2$ **then**
(7) $L_{k+1} = \eta L_k$, go to Step 1
(8) **else**
(9) $L_k = 2L_k$
(10) **end if**
(11) **end while**
(12) Set $t_k = (1 + \sqrt{1 + 4t_{k-1}^2})/2$
(13) **end for**

Algorithm 3: The modified ASVT (m-asvt) algorithm, algorithm 1 of [23]

$$\lim_{k \longrightarrow \infty} \mathcal{P}_\Omega\left(M - X^k\right) = \lim_{k \longrightarrow \infty} \mathcal{P}_\Omega\left(M - \widetilde{X}^k\right) = 0. \tag{13}$$

Proof. It follows from Algorithm 4 that

$$\begin{aligned} Y^k - \mathcal{P}_\Omega(M) &= Y^{k-1} - \mathcal{P}_\Omega(M) + \delta_k\left(\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega\left(X^k\right)\right) \\ &= \left(1-\delta_k\right)\left(Y^{k-1} - \mathcal{P}_\Omega(M)\right) + \delta_k\left(Y^{k-1} - \mathcal{P}_\Omega\left(X^k\right)\right) \\ &= \left(1-\delta_k\right)\left(Y^{k-1} - \mathcal{P}_\Omega(M)\right) + \delta_k\left(Y^{k-1} - \mathcal{P}_\Omega\left(\widetilde{X}^k\right) + \mathcal{P}_\Omega\left(\widetilde{X}^k\right) - \mathcal{P}_\Omega\left(X^k\right)\right) \\ &= \left(1-\delta_k\right)\left(Y^{k-1} - \mathcal{P}_\Omega(M)\right) + \delta_k\left(Y^{k-1} - \mathcal{P}_\Omega\left(\widetilde{X}^k\right)\right) + \delta_k \mathcal{P}_\Omega\left(\widetilde{X}^k - X^k\right). \end{aligned} \tag{14}$$

Input: sampled set Ω and sampled entries $\mathcal{P}_\Omega(M)$, step size $\delta \in (0,2)$, tolerance ε, parameter $\tau > 0$, increment ℓ, and maximum iteration count $k_{\max}$
Output: X^{opt}
Description: recover a low-rank matrix M from a subset of sampled entries
Set $Y^0 = k_0 \delta \mathcal{P}_\Omega(M)$, k_0 is an integer with $(\tau/\delta \|\mathcal{P}_\Omega(M)\|_2] \in (k_0 - 1, k_0)$
Set $r_0 = 0$
for $k = 1$ to $k_{\max}$
  Set $s_k = r_{k-1} + 1$
  repeat
    Compute $[U^{k-1}, \Sigma^{k-1}, V^{k-1}]_{s_k}$
   Set $s_k = s_k + \ell$
 until $\sigma^{k-1}_{s_k - \ell} \le \tau$
 Set $r_k = \max\left\{j \colon \sigma^{k-1}_j > \tau\right\}$
  Set $\widetilde{X}^k = \sum_{j=1}^{r_k} (\sigma^{k-1}_j - \tau) u^{k-1}_j v^{k-1}_j$
  Compute $W_k = \operatorname{argmin}_W \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(\widetilde{X}_k) W_k\|_F$
  Set $X^k = \mathcal{P}_\Omega(\widetilde{X}_k) W_k$
  if $\|\mathcal{P}_\Omega(X^k - M)\|_F < \varepsilon \|\mathcal{P}_\Omega(M)\|_F$ **then break**
  Set $Y^k_{ij} = \begin{cases} Y^{k-1}_{ij} + \delta(M_{ij} - X^k_{ij}), & \text{if } (i,j) \in \Omega, \\ 0, & \text{if } (i,j) \notin \Omega \end{cases}$
end *for* k
 Set $X^{\text{opt}} = X^k$

Algorithm 4: Singular value thresholding with diagonal-update (D-SVT) algorithm, Algorithm 1 of [15]

In term of the assumption of this theorem, $\forall k, \exists \Theta$ such that

$$\left\| \mathcal{P}_\Omega\left(\widetilde{X}^k\right)(I - W_k)\right\|_F \le \Theta \tag{15}$$

holds true.

Let $Z_k = : Y^k - \mathcal{P}_\Omega(M)$. We note that $X^k = \mathcal{P}_\Omega(\widetilde{X}_k) W_k$ from Algorithm 4, and by substituting the following inequalities

$$\begin{aligned} \left\| \mathcal{P}_\Omega\left(Y^{k-1} - \widetilde{X}^k\right)\right\|_F &\le \left\| Y^{k-1} - \widetilde{X}^k \right\|_F \le \sqrt{n}\tau, \\ \left\| \mathcal{P}_\Omega\left(\widetilde{X}^k - X^k\right)\right\|_F &= \left\| \mathcal{P}_\Omega\left(\widetilde{X}^k (I - W_k)\right)\right\|_F \le \Theta, \end{aligned} \tag{16}$$

we have

$$\begin{aligned} \left\| Z_k \right\|_F &\le \left|1 - \delta_k\right| \left\| Z_{k-1} \right\|_F + \delta_k \sqrt{n}\tau + \delta_k \Theta \le \cdots \\ &\le \prod_{i=1}^{k} \left|1 - \delta_i\right| \left\| Z_0 \right\|_F + \sum_{i=1}^{k} \prod_{j=1}^{i} \left|1 - \delta_j\right| \left(\delta_i \sqrt{n}\,\tau + \delta_i \Theta\right) \\ &\le c^k \left\| Z_0 \right\|_F + \sum_{i=1}^{k} c^i \delta_i \left(\sqrt{n}\,\tau + M\right), \quad if \left|1 - \delta_i\right| \\ &\le c < 1 \le c^k \left\| Z_0 \right\|_F + \frac{2}{1-c} (n\tau + M). \end{aligned} \tag{17}$$

Hence, $\{Z_k\}_{k=1}^\infty$ is bounded and so is $\{Y^k\}_{k=1}^\infty$. Moreover, it is obtained that

$$\lim_{i \longrightarrow \infty} \mathcal{P}_\Omega\left(M - X^i\right) = 0, \tag{18}$$

from the equation

$$Y^k = \sum_{i=1}^{k} \delta_i \left(\mathcal{P}_\Omega\left(M - X^i\right)\right). \tag{19}$$

Moreover,

$$\lim_{i \longrightarrow \infty} \mathcal{P}_\Omega\left(M - \widetilde{X}^i\right) = 0, \tag{20}$$

since $\lim_{k \longrightarrow \infty} W_k = I$. The theorem has been proved. □

Theorem 2. *Let $X_{\dagger,\tau}$ be the limiting point of the sequence $\{X^k\}$ generated by Algorithm 4. Then, $X_{\dagger,\tau}$ is the solution of the optimization problem (3).*

Proof. It is obtained that $\widetilde{X}^k$ is the optimal solution of (8) for that value of Y^{k-1} from Algorithm 4.

Hence, for any feasible matrix X, it is yielded that

$$\begin{aligned} \tau \|X\|_* + \frac{1}{2}\|X\|_F^2 &\ge \mathcal{L}\left(\widetilde{X}^k, Y^{k-1}\right) \\ &\ge \lim_{k \longrightarrow \infty} \left(\mathcal{L}\left(\widetilde{X}^k, Y^{k-1}\right)\right) \\ &= \mathcal{L}\left(X_{\dagger,\tau}, Y_{\dagger,\tau}\right) \\ &= \tau \left\| X_{\dagger,\tau} \right\|_* + \frac{1}{2} \left\| X_{\dagger,\tau} \right\|_F^2. \end{aligned} \tag{21}$$

Thus, $X_{\dagger,\tau}$ is the unique solution of (3). □

Table 1: Computational results for small-size problems when $\tau = 5n$.

Unknown M			Computational results				
Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
1,000	20	0.20	SVT	275	101.6901	$1.9826e-04$	$3.2515e-04$
			ASVT	267	97.7211	$1.0021e-04$	$1.8735e-04$
			M-ASVT	251	86.5238	$1.2541e-04$	$2.3361e-04$
			D-SVT	**186**	**71.0987**	$\mathbf{1.9793e-04}$	$\mathbf{3.1915e-04}$
1,000	30	0.30	SVT	255	88.2853	$1.9663e-04$	$3.0680e-04$
			ASVT	248	83.2158	$1.0002e-04$	$2.1714e-04$
			M-ASVT	229	80.2655	$1.3326e-04$	$1.9855e-04$
			D-SVT	**170**	**70.0549**	$\mathbf{1.9705e-04}$	$\mathbf{3.0682e-04}$
2,000	30	0.15	SVT	271	837.7041	$1.9803e-04$	$3.2390e-04$
			ASVT	266	821.4554	$1.0204e-04$	$2.1444e-04$
			M-ASVT	258	799.8985	$1.2130e-04$	$2.2225e-04$
			D-SVT	**182**	**675.1372**	$\mathbf{1.9687e-04}$	$\mathbf{3.1833e-04}$
2,000	40	0.20	SVT	262	783.2010	$1.9843e-04$	$3.1685e-04$
			ASVT	248	766.1247	$1.0203e-04$	$1.5682e-04$
			M-ASVT	233	758.1213	$1.8860e-04$	$2.3310e-04$
			D-SVT	**165**	**619.6730**	$\mathbf{1.9892e-04}$	$\mathbf{3.1476e-04}$
3,000	30	0.10	SVT	276	2758.2	$1.9699e-04$	$3.2527e-04$
			ASVT	265	2710.2236	$1.0007e-04$	$1.9985e-04$
			M-ASVT	260	2681.2874	$1.5623e-04$	$2.1414e-04$
			D-SVT	**185**	**2076.1**	$\mathbf{1.9885e-04}$	$\mathbf{3.2650e-04}$
3,000	40	0.13	SVT	267	2692.5	$1.9609e-04$	$3.1899e-04$
			ASVT	261	2551.4478	$1.0014e-04$	$1.8569e-04$
			M-ASVT	255	2432.1278	$1.7784e-04$	$2.3369e-04$
			D-SVT	**174**	**2105.3**	$\mathbf{1.9943e-04}$	$\mathbf{3.2126e-04}$
3,000	50	0.17	SVT	259	2670.6	$1.9688e-04$	$3.1486e-04$
			ASVT	254	2641.2587	$1.0012e-04$	$2.3331e-04$
			M-ASVT	239	2598.5623	$1.7456e-04$	$2.2114e-04$
			D-SVT	**162**	**2092.8**	$\mathbf{1.9611e-04}$	$\mathbf{3.1200e-04}$
4,000	30	0.07	SVT	276	6793	$1.9824e-04$	$3.2954e-04$
			ASVT	270	6659	$1.0040e-04$	$2.3322e-04$
			M-ASVT	261	6555	$1.9998e-04$	$2.2288e-04$
			D-SVT	**186**	**5409.1**	$\mathbf{1.9823e-04}$	$\mathbf{3.2674e-04}$
4,000	40	0.10	SVT	271	6184.3	$1.9895e-04$	$3.2622e-04$
			ASVT	276	6155	$1.1012e-04$	$2.1170e-04$
			M-ASVT	269	6099	$1.9963e-04$	$2.0208e-04$
			D-SVT	**180**	**5289.6**	$\mathbf{1.9689e-04}$	$\mathbf{3.2015e-04}$
4,000	50	0.12	SVT	269	6296.3	$1.9889e-04$	$3.2247e-04$
			ASVT	263	6241.2	$1.0500e-04$	$1.5589e-04$
			M-ASVT	260	6258.2	$1.1010e-04$	$1.6601e-04$
			D-SVT	**186**	**4687.2**	$\mathbf{1.9802e-04}$	$\mathbf{3.1833e-04}$

Theorem 3. *Suppose that* $\lim_{\tau\longrightarrow\infty} X_{\dagger,\tau} = X_\dagger$. *Then,* $X_\dagger$ *is the optimal solution of the optimization problem (2).*

Proof. Note that $X_\dagger$ is the optimal solution of the optimization problem (2) if and only if

$$0 \in \mathcal{P}_{\overline{\Omega}}\left(\partial \left\| X_\dagger \right\|_*\right). \tag{22}$$

From Theorem 2, we have

$$0 \in \mathcal{P}_{\overline{\Omega}}\left(\tau \partial \left\| X_{\dagger,\tau} \right\|_* + X_{\dagger,\tau}\right), \tag{23}$$

which implies that

$$0 \in \mathcal{P}_{\overline{\Omega}}\left(\partial \left\| X_{\dagger,\tau} \right\|_* + \frac{1}{\tau} X_{\dagger,\tau}\right). \tag{24}$$

Therefore,

$$\lim_{\tau\longrightarrow\infty} \frac{1}{\tau} X_{\dagger,\tau} = 0, \tag{25}$$

since $X_\dagger$ is bounded. Thus,

$$0 \in \mathcal{P}_{\overline{\Omega}}\left(\partial \left\| X_\dagger \right\|_*\right). \tag{26}$$

□

4. Numerical Experiments

In this section, we provide the performance of our D-SVT algorithm in comparison with the SVT, ASVT, and M-ASVT algorithms mentioned in Section 2 and report the running time in seconds (denoted by "time (s)"), the numbers of iterations (denoted by "IT") it takes to reach convergence, and the relative errors of the reconstruction (denoted by" Error 1 and Error 2″) as follows:

Table 2: Computational results for small-size problems when $\tau = 4n$.

Unknown M			Computational results				
Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
1,000	20	0.20	SVT	223	76.3553	1.9830$e-04$	3.2593$e-04$
			ASVT	206	71.2289	1.0054$e-04$	1.9965$e-04$
			M-ASVT	194	69.5563	1.1023	1.9877$e-04$
			D-SVT	**156**	**66.9487**	**1.9748 $e-04$**	**3.2191 $e-04$**
1,000	30	0.30	SVT	207	70.3992	1.9988$e-04$	3.1406$e-04$
			ASVT	200	68.2241	1.1145$e-04$	2.1365$e-04$
			M-ASVT	192	63.2289	1.0089$e-04$	2.0005$e-04$
			D-SVT	**144**	**61.4286**	**1.9333 $e-04$**	**3.0025 $e-04$**
2,000	30	0.15	SVT	220	712.9493	1.9956$e-04$	3.2718$e-04$
			ASVT	217	704.3365	1.0203$e-04$	2.1475$e-04$
			M-ASVT	211	699.3785	1.0057$e-04$	2.0624$e-04$
			D-SVT	**152**	**535.2846**	**1.9969 $e-04$**	**3.2316 $e-04$**
2,000	40	0.20	SVT	212	644.3027	1.9890$e-04$	3.1802$e-04$
			ASVT	205	634.9870	1.1110$e-04$	2.3875$e-04$
			M-ASVT	199	618.9952	1.0074$e-04$	2.2271$e-04$
			D-SVT	**148**	**522.8293**	**1.9969 $e-04$**	**3.1650 $e-04$**
3,000	30	0.10	SVT	224	2248.9	1.9831$e-04$	3.2891$e-04$
			ASVT	218	2213.4	1.3354$e-04$	2.6522$e-04$
			M-ASVT	211	2168.8	1.1532$e-04$	2.3301$e-04$
			D-SVT	**158**	**1824.8**	**1.9718 $e-04$**	**3.2220 $e-04$**
3,000	40	0.13	SVT	216	2222.4	1.9616$e-04$	3.1860$e-04$
			ASVT	207	2025.3	1.1004$e-04$	2.3110$e-04$
			M-ASVT	201	1999.2	1.2121$e-04$	2.5403$e-04$
			D-SVT	**152**	**1807.7**	**1.9394 $e-04$**	**3.1228 $e-04$**
3,000	50	0.17	SVT	211	2234.1	1.9597$e-04$	3.1398$e-04$
			ASVT	204	2106.4	1.1140$e-04$	2.5510$e-04$
			M-ASVT	198	1999.4	1.0042$e-04$	2.1030$e-04$
			D-SVT	**148**	**1733.6**	**1.9507 $e-04$**	**3.1039 $e-04$**
4,000	30	0.07	SVT	224	5314.0	1.9728$e-04$	3.2816$e-04$
			ASVT	218	5127.1	1.2544$e-04$	2.3365$e-04$
			M-ASVT	209	5001.2	1.1004$e-04$	1.9952$e-04$
			D-SVT	**160**	**4324.1**	**1.9500 $e-04$**	**3.2075 $e-04$**
4,000	40	0.10	SVT	218	5209.7	1.9845$e-04$	3.2463$e-04$
			ASVT	211	5101.0	1.1140$e-04$	2.0407$e-04$
			M-ASVT	205	4998.2	1.0045$e-04$	1.8854$e-04$
			D-SVT	**155**	**4237.7**	**1.9460 $e-04$**	**3.1656 $e-04$**
4,000	50	0.12	SVT	215	5114.3	1.9847$e-04$	3.2178$e-04$
			ASVT	207	5003.5	1.6652$e-04$	2.3310$e-04$
			M-ASVT	200	4889.2	1.1143$e-04$	2.1113$e-04$
			D-SVT	**151**	**4068.9**	**1.9986 $e-04$**	**3.2133 $e-04$**

$$\text{Error 1} = \frac{\left\|X_k - \mathcal{P}_\Omega(M)\right\|_F}{\left\|\mathcal{P}_\Omega(M)\right\|_F},$$
$$\text{Error 2} = \frac{\left\|X_k W_k - M\right\|_F}{\|M\|_F}. \tag{27}$$

All the experiments are conducted on the same workstation with an Intel (R) Core (TM) i7-6700 CPU @3.40GHZ that has 16 GB memory and 64 bit operating system, running Windows 7, and Matlab (vision 2016a). For conciseness, the tests presented consider square matrices as is typical in the study. That is to say, suppose to simplify that the unknown matrix $M \in \mathfrak{R}^{n\times n}$ is square and that one has available s sampled entries $\left\{M_{ij}\colon (i,j)\in\Omega\right\}$, where Ω is a random subset of cardinality s. By the way, the iteration fails if the number of iterations is up to 1000.

In our implement, we generate $n \times n$ matrices of rank r by sampling uniformly at random among all sets of cardinality s; then, $p = s/n^2$ denotes the observation ratio. Let $\varepsilon = 2 \times 10^{-4}$. As discussed earlier [15], $\delta = 1.2\,\mathrm{p}^{-1}$ and the step sizes are constant and the parameter τ is chosen empirically. And as presented earlier [23], the parameters $L_1 = 1$ and $\eta = 0.8$. As for D-SVT algorithm, we choose $\delta = 1.68\,\mathrm{p}^{-1}$, and τ is the same as the SVT algorithm.

The tested matrix dimensions (denoted by "size (n)") are from 1,000 to 12,000. The experimental results are shown in Tables 1–6. Our experiments suggest that Algorithm 4 is fast and significantly outperforms the other algorithms in terms of both number of iteration steps and computing time. The

Table 3: Computational results for small-size problems when $\tau = 3n$.

Unknown M Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
1,000	20	0.198	SVT	172	58.9324	1.9708$e-04$	3.2399$e-04$
			ASVT	165	55.3211	1.2114$e-04$	2.1140$e-04$
			M-ASVT	160	51.3333	1.1002$e-04$	2.0047$e-04$
			D-SVT	**119**	**50.8026**	**1.9333 $e-04$**	**3.1028 $e-04$**
1,000	30	0.2955	SVT	158	55.6121	1.9651$e-04$	3.0783$e-04$
			ASVT	150	51.3361	1.2254$e-04$	2.5512$e-04$
			M-ASVT	143	48.2210	1.1102$e-04$	2.0005$e-04$
			D-SVT	**110**	**48.3159**	**1.9430 $e-04$**	**3.0057 $e-04$**
2,000	30	0.1489	SVT	169	509.3663	1.9555$e-04$	3.1915$e-04$
			ASVT	161	500.2369	1.2323$e-04$	1.9985$e-04$
			M-ASVT	156	498.2323	1.1110$e-04$	1.8541$e-04$
			D-SVT	**117**	**410.1959**	**1.9399 $e-04$**	**3.1251 $e-04$**
2,000	40	0.198	SVT	163	491.4569	1.9544$e-04$	3.1199$e-04$
			ASVT	157	482.3145	1.3324$e-04$	2.5666$e-04$
			M-ASVT	152	469.2221	1.1142$e-04$	2.1104$e-04$
			D-SVT	**113**	**399.2143**	**1.9981 $e-04$**	**3.1611 $e-04$**
3,000	30	0.0995	SVT	173	1810.3	1.9523$e-04$	3.2420$e-04$
			ASVT	166	1796.2	1.5562$e-04$	2.0004$e-04$
			M-ASVT	159	1652.8	1.1124$e-04$	1.8989$e-04$
			D-SVT	**119**	**1386.7**	**1.9967 $e-04$**	**3.2635 $e-04$**
3,000	40	0.1324	SVT	166	1713.3	1.9533$e-04$	3.1671$e-04$
			ASVT	159	1695.2	1.5252$e-04$	2.8485$e-04$
			M-ASVT	154	1666.3	1.2235$e-04$	2.1103$e-04$
			D-SVT	**116**	**1355.3**	**1.9953 $e-04$**	**3.2249 $e-04$**
3,000	50	0.1653	SVT	162	1664.9	1.9418$e-04$	3.1127$e-04$
			ASVT	157	1662.3	1.6523$e-04$	2.1104$e-04$
			M-ASVT	157	1660.2	1.0023$e-04$	2.1001$e-04$
			D-SVT	**113**	**1333.3**	**1.9641 $e-04$**	**3.1282 $e-04$**
4,000	30	0.0747	SVT	174	4112.2	1.9442$e-04$	3.2322$e-04$
			ASVT	170	4000.3	1.1115$e-04$	2.1104$e-04$
			M-ASVT	168	3995.2	1.0002$e-04$	1.9952$e-04$
			D-SVT	**120**	**3247.9**	**1.9820 $e-04$**	**3.2760 $e-04$**
4,000	40	0.0995	SVT	168	4023.7	1.9513$e-04$	3.1937$e-04$
			ASVT	163	3998.2	1.6674$e-04$	2.5510$e-04$
			M-ASVT	159	3885.1	1.3114$e-04$	2.0047$e-04$
			D-SVT	**117**	**3056.1**	**1.9684 $e-04$**	**3.1983 $e-04$**
4,000	50	0.1242	SVT	165	3803.5	1.9604$e-04$	3.1727$e-04$
			ASVT	160	3685.2	1.2246$e-04$	2.8525$e-04$
			M-ASVT	156	3459.2	1.1006$e-04$	2.1139$e-04$
			D-SVT	**115**	**3104.3**	**1.9875 $e-04$**	**3.1965 $e-04$**

Table 4: Computational results for large-size problems when $\tau = 5n$.

Unknown M Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
5,000	40	0.08	SVT	271	12334	1.9826$e-04$	3.2645$e-04$
			ASVT	266	12301	1.2225$e-04$	2.1147$e-04$
			M-ASVT	264	12295	1.0558$e-04$	2.0036$e-04$
			D-SVT	**190**	**9845.8**	**1.9556 $e-04$**	**3.1916 $e-04$**
5,000	50	0.10	SVT	265	11505	1.9898$e-04$	3.2434$e-04$
			ASVT	263	11427	1.2558$e-04$	3.0001$e-04$
			M-ASVT	260	11582	1.1010$e-04$	2.5563$e-04$
			D-SVT	**186**	**9312.2**	**1.9976 $e-04$**	**3.2374 $e-04$**

Table 4: Continued.

Unknown M			Computational results				
Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
5,000	60	0.12	SVT	262	11983	1.9795$e-04$	3.1961$e-04$
			ASVT	259	11886	1.5252$e-04$	2.5474$e-04$
			M-ASVT	256	11963	1.1023$e-04$	2.1136$e-04$
			D-SVT	**185**	**9656.4**	**1.9581 $e-04$**	**3.1450 $e-04$**
8,000	50	0.06	SVT	269	48699	1.9902$e-04$	3.2707$e-04$
			ASVT	263	49987	1.1298$e-04$	2.6634$e-04$
			M-ASVT	256	50000	1.0002$e-04$	1.8859$e-04$
			D-SVT	**191**	**40793**	**1.9714 $e-04$**	**3.2171 $e-04$**
8,000	60	0.07	SVT	266	46717	1.9926$e-04$	3.2572$e-04$
			ASVT	259	47558	1.3320$e-04$	2.0045$e-04$
			M-ASVT	251	48800	1.0223$e-04$	1.8879$e-04$
			D-SVT	**189**	**37324**	**1.9591 $e-04$**	**3.1842 $e-04$**
8,000	100	0.12	SVT	258	48230	1.9750$e-04$	3.1624$e-04$
			ASVT	254	47889	1.2220$e-04$	2.1145$e-04$
			M-ASVT	250	47556	1.0023$e-04$	1.9987$e-04$
			D-SVT	**183**	**38534**	**1.9695 $e-04$**	**3.1436 $e-04$**
8,000	150	0.19	SVT	250	43758	1.9707$e-04$	3.0923$e-04$
			ASVT	248	43665	1.3365$e-04$	2.1104$e-04$
			M-ASVT	245	43666	1.1102$e-04$	1.9965$e-04$
			D-SVT	**177**	**35029**	**1.9794 $e-04$**	**3.1001 $e-04$**
10,000	200	0.20	SVT	247	85840	1.9976$e-04$	3.1183$e-04$
			ASVT	241	85662	1.2003$e-04$	2.0014$e-04$
			M-ASVT	238	85441	1.0041$e-04$	1.9941$e-04$
			D-SVT	**176**	**69674**	**1.9795 $e-04$**	**3.0853 $e-04$**
10,000	500	0.49	SVT	215	71656	1.9760$e-04$	2.81173$e-04$
			ASVT	211	71221	1.2258$e-04$	2.0041$e-04$
			M-ASVT	207	71234	1.0005$e-04$	1.9595$e-04$
			D-SVT	**155**	**59264**	**1.9531 $e-04$**	**2.7785 $e-04$**
12,000	300	0.25	SVT	242	139650	1.9701$e-04$	3.0300$e-04$
			ASVT	239	139666	1.2254$e-04$	2.3314$e-04$
			M-ASVT	234	139556	1.0005$e-04$	2.0101$e-04$
			D-SVT	**173**	**128650**	**1.9664 $e-04$**	**3.0193 $e-04$**
12,000	800	0.64	SVT	195	110750	1.9964$e-04$	2.6726$e-04$
			ASVT	191	115544	1.2023$e-04$	1.8856$e-04$
			M-ASVT	187	115531	1.0006$e-04$	1.5654$e-04$
			D-SVT	**141**	**97547**	**1.9785 $e-04$**	**2.6492 $e-04$**

Table 5: Computational results for large-size problems when $\tau = 4n$.

Unknown M			Computational results				
Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
5,000	40	0.08	SVT	219	9835.9	1.9870$e-04$	3.2720$e-04$
			ASVT	215	9755.2	1.4451$e-04$	2.5656$e-04$
			M-ASVT	209	9741.3	1.0907$e-04$	2.1036$e-04$
			D-SVT	**156**	**8041.4**	**1.9393 $e-04$**	**3.1676 $e-04$**
5,000	50	0.10	SVT	216	9453.7	1.9655$e-04$	3.2005$e-04$
			ASVT	211	9431.2	1.2001$e-04$	2.5543$e-04$
			M-ASVT	206	9388.2	1.1007$e-04$	2.1016$e-04$
			D-SVT	**152**	**7509.2**	**1.9991 $e-04$**	**2.2336 $e-04$**
5,000	60	0.12	SVT	213	9383.4	1.9786$e-04$	3.1958$e-04$
			ASVT	209	9321.0	1.3357$e-04$	2.5151$e-04$
			M-ASVT	201	9289.1	1.1110$e-04$	2.1910$e-04$
			D-SVT	**151**	**7576.3**	**1.9481 $e-04$**	**3.1298 $e-04$**
8,000	50	0.06	SVT	220	39280	1.9652$e-04$	3.2339$e-04$
			ASVT	218	38923	1.5001$e-04$	2.4410$e-04$
			M-ASVT	211	38009	1.0944$e-04$	2.1106$e-04$
			D-SVT	**156**	**31915**	**1.9525 $e-04$**	**3.1915 $e-04$**

Table 5: Continued.

Unknown M				Computational results			
Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
8,000	60	0.07	SVT	217	38236	$1.9566e-04$	$3.1953e-04$
			ASVT	214	38001	$1.2238e-04$	$2.3131e-04$
			M-ASVT	208	37996	$1.1010e-04$	$2.0045e-04$
			D-SVT	**153**	**30619**	**1.9923 $e-04$**	**3.2345 $e-04$**
8,000	100	0.12	SVT	209	37745	$1.9771e-04$	$3.1676e-04$
			ASVT	206	37512	$1.2325e-04$	$2.0109e-04$
			M-ASVT	200	37008	$1.0056e-04$	$1.9221e-04$
			D-SVT	**148**	**30537**	**1.9913 $e-04$**	**3.1791 $e-04$**
8,000	150	0.19	SVT	203	37414	$1.9488e-04$	$3.0607e-04$
			ASVT	199	37002	$1.1125e-04$	$2.0087e-04$
			M-ASVT	196	36912	$1.0023e-04$	$1.8989e-04$
			D-SVT	**144**	**29507**	**1.9832 $e-04$**	**3.1065 $e-04$**
10,000	200	0.20	SVT	201	67135	$1.9474e-04$	$3.0422e-04$
			ASVT	198	68852	$1.2365e-04$	$2.5620e-04$
			M-ASVT	195	69592	$1.0101e-04$	$1.9962e-04$
			D-SVT	**144**	**56015**	**1.9315 $e-04$**	**3.0102 $e-04$**
10,000	500	0.49	SVT	174	58453	$1.9652e-04$	$2.7972e-04$
			ASVT	171	60033	$1.2020e-04$	$1.9963e-04$
			M-ASVT	167	60114	$1.0023e-04$	$1.5651e-04$
			D-SVT	**126**	**48758**	**1.9709 $e-04$**	**2.8051 $e-04$**
12,000	300	0.25	SVT	196	11284	$1.9603e-04$	$3.0148e-04$
			ASVT	-	-	-	-
			M-ASVT	-	-	-	-
			D-SVT	**131**	**10375**	**1.9734 $e-04$**	**3.0320 $e-04$**
12,000	800	0.64	SVT	158	88478	$1.9661e-04$	$2.6335e-04$
			ASVT	-	-	-	-
			M-ASVT	-	-	-	-
			D-SVT	**115**	**78069**	**1.9509 $e-04$**	**2.6139 $e-04$**

Note. The mark "-" indicates that the iteration is failing.

Table 6: Computational results for large-size problems when $\tau = 3n$.

Unknown M				Computational results			
Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
5,000	40	0.0797	SVT	169	7838.0	$1.9546e-04$	$3.2133e-04$
			ASVT	161	7775.1	$1.2230e-04$	$2.3232e-04$
			M-ASVT	156	7701.2	$1.0023e-04$	$2.0045e-04$
			D-SVT	**118**	**6406.5**	**1.9422 $e-04$**	**3.1763 $e-04$**
5,000	50	0.0995	SVT	166	7343.0	$1.9364e-04$	$3.1582e-04$
			ASVT	160	7265.2	$1.2325e-04$	$2.4789e-04$
			M-ASVT	155	7189.2	$1.0089e-04$	$2.1004e-04$
			D-SVT	**116**	**5780.4**	**1.9280 $e-04$**	**3.1234 $e-04$**
5,000	60	0.1193	SVT	164	7485.6	$1.9450e-04$	$3.1432e-04$
			ASVT	160	7401.2	$1.5002e-04$	$2.3004e-04$
			M-ASVT	154	7356.3	$1.0980e-04$	$2.0152e-04$
			D-SVT	**115**	**5833.1**	**1.9562 $e-04$**	**3.1424 $e-04$**
8,000	50	0.0623	SVT	168	31774	$1.9968e-04$	$3.2825e-04$
			ASVT	165	32589	$1.2223e-04$	$2.0058e-04$
			M-ASVT	160	32441	$1.0041e-04$	$1.8856e-04$
			D-SVT	**118**	**24511**	**1.9267 $e-04$**	**3.1490 $e-04$**
8,000	60	0.0747	SVT	166	29378	$1.9945e-04$	$3.2418e-04$
			ASVT	161	29654	$1.2412e-04$	$2.0306e-04$
			M-ASVT	157	30001	$1.0002e-04$	$1.5859e-04$
			D-SVT	**117**	**23791**	**1.9287 $e-04$**	**3.1360 $e-04$**

Table 6: Continued.

Unknown M Size (n)	Rank (r)	p	Algorithm	IT	Time (s)	Error 1	Error 2
8,000	100	0.1242	SVT	160	30315	$1.9888e-04$	$3.1854e-04$
			ASVT	156	31256	$1.2004e-04$	$2.1113e-04$
			M-ASVT	149	33000	$1.0100e-04$	$1.8588e-04$
			D-SVT	**115**	**23863**	**1.9124 $e-04$**	**3.0560 $e-04$**
8,000	150	0.1857	SVT	155	27204	$1.9643e-04$	$3.0842e-04$
			ASVT	150	28693	$1.2203e-04$	$2.3334e-04$
			M-ASVT	144	29003	$1.0045e-04$	$201414e-04$
			D-SVT	**112**	**22002**	**1.9250 $e-04$**	**3.0152 $e-04$**
10,000	200	0.1980	SVT	154	52658	$1.9363e-04$	$3.0252e-04$
			ASVT	153	53321	$1.1414e-04$	$2.3006e-04$
			M-ASVT	150	53662	$1.0051e-04$	$2.0001e-04$
			D-SVT	**111**	**44680**	**1.9978 $e-04$**	**3.1153 $e-04$**
10,000	500	0.1980	SVT	133	44718	$1.9555e-04$	$2.7855e-04$
			ASVT	131	45512	$1.3224e-04$	$2.5101e-04$
			M-ASVT	126	45510	$1.0021e-04$	$1.5880e-04$
			D-SVT	**98**	**38655**	**1.9057 $e-04$**	**2.7139 $e-04$**
12,000	300	0.2469	SVT	150	86110	$1.9587e-04$	$3.0132e-04$
			ASVT	-	-	-	-
			M-ASVT	-	-	-	-
			D-SVT	**109**	**81902**	**1.9999 $e-04$**	**3.0734 $e-04$**
12,000	800	0.6444	SVT	121	68906	$1.9236e-04$	$2.5786e-04$
			ASVT	-	-	-	-
			M-ASVT	-	-	-	-
			D-SVT	**89**	**63517**	**1.9120 $e-04$**	**2.5640 $e-04$**

Note. The mark "-"indicates that the iteration is failing.

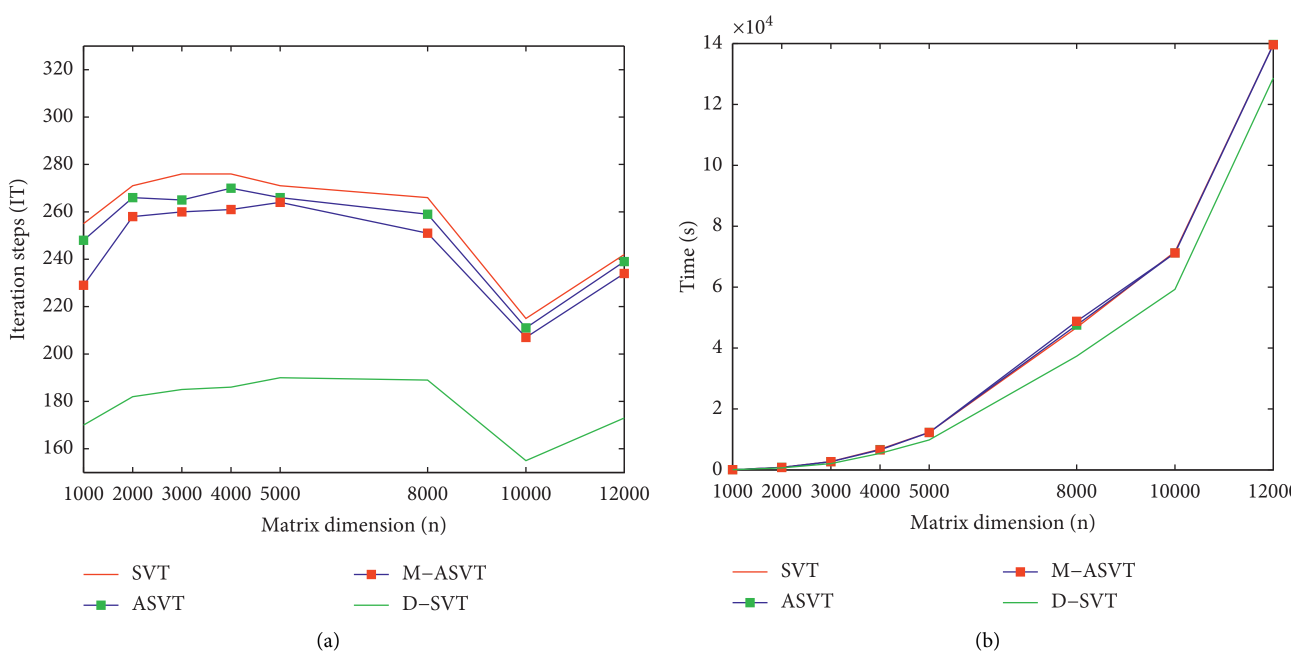

Figure 1: Continued.

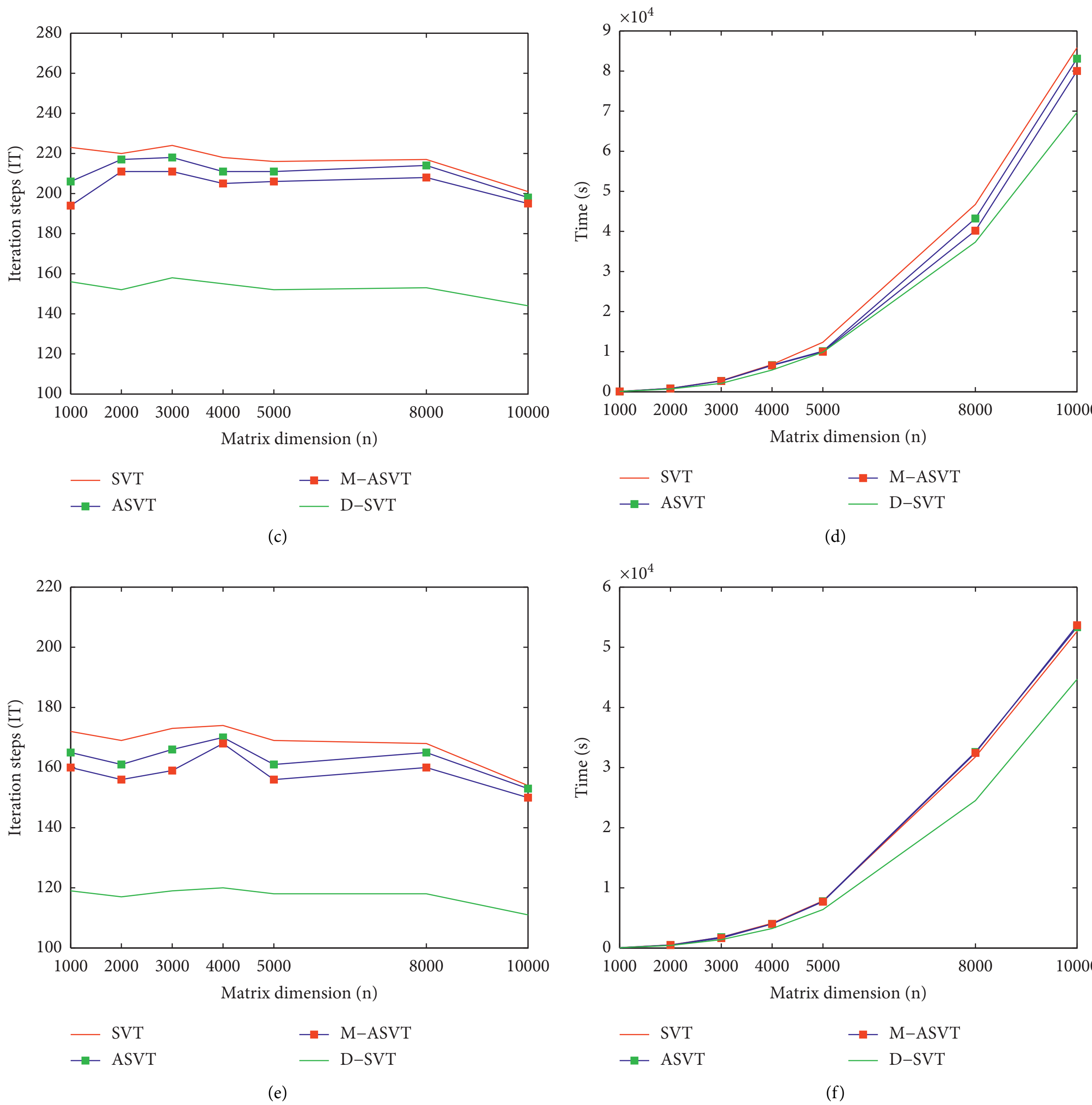

FIGURE 1: Convergence curves of several algorithms for the different parameters. (a) The iteration steps of several algorithms for $t = 5$ n. (b) The time of several algorithms for $t = 5$ n. (c) The iteration steps of several algorithms for $t = 4$ n. (d) The time of several algorithms for $t = 4$ n. (e) The iteration steps of several algorithms for $t = 3$ n. (f) The time of several algorithms for $t = 3$ n.

new algorithm is especially well suited for problems of very large sizes.

In order to show convergence behave of the algorithms briefly, convergence curves of several algorithms are clearly given, which are shown in Figure 1 for the different parameters. It is easy to see that our algorithm takes much less computational cost from iteration steps and computing time. That is to say, the D-SVT algorithm is more efficient than the other algorithms especially when the size of matrix M is large.

5. Concluding Remarks

In this paper, we focus on the problem of completing a low-rank matrix from a small subset of its entries. This model can characterize many applications arising from the areas of signal and image processing, statistical inference, and machine learning. We have proposed a modification of the SVT algorithm for solving the low-rank matrix sparse model. The key step of the algorithm is to update each iteration matrix by a weighting diagonal matrix, without the extra cost. The weighting matrix W was determined adaptively in iteration process. This algorithm is easy to implement and surprisingly effective both in terms of computational cost and storage requirement. Consequently, the matrix would be completed well.

Data Availability

We generate a matrice of rank r by sampling randomly in our implement. The readers can access the data supporting the conclusions of the study by MATLAB codes.

Conflicts of Interest

The authors declare that they have no conflicts of interests.

Authors' Contributions

All authors contributed equally to the writing of this paper. All authors read and approved the final manuscript.

Acknowledgments

This work was supported by the NSF of China (11371275), NSF of Shanxi Province (201901D211423), STIP of Shanxi Provincial Department of Education (2020L0719), and the CSREP in Shanxi (no. 2019KJ035).

References

[1] Y. Amit, M. Fink, N. Srebro, and S. Ullman, "Unconvering shared structures in malticalass classification," in *Processing of the 24th International Conference on Machine Learning*, pp. 17–24, ACM, Guangzhou, China, November 2007.

[2] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," *Advances in Neural Information Processing Systems*, vol. 19, pp. 41–48, 2007.

[3] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, pp. 1235–1256, 2009.

[4] J.-F. Cai, T. Wang, and K. Wei, "Spectral compressed sensing via projected gradient descent," *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 2625–2653, 2018.

[5] M. Mesbahi and G. P. Papavassilopoulos, "On the rank minimization problem over a positive semidefinite linear matrix inequality," *Institute of Electrical and Electronics Engineers Transactions on Automatic Control*, vol. 42, no. 2, pp. 239–243, 1997.

[6] L. Eldén, *Matrix Methods in Data Mining and Pattern Recognization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[7] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Multi-task feature learing, image inpainting," *GR Computer & Stationary*, vol. 34, pp. 417–424, 2000.

[8] J. -F. Cai, S. Liu, and W. Xu, "A fast algorithm for restruction of spectrally sparse signals in super-resolution," in *Proceedings of the SPIE Optical Engineering + Applications*, p. 95970Ap. 95970A, International Society for Optics and Photonics, San Francisco, CA, USA, April 2015.

[9] J.-F. Cai, X. Qu, W. Xu, and G.-B. Ye, "Robust recovery of complex exponential signals from random Gaussian projections via low rank Hankel matrix reconstruction," *Applied and Computational Harmonic Analysis*, vol. 41, no. 2, pp. 470–490, 2016.

[10] J.-F. Cai, T. Wang, and K. Wei, "Fast and provable algorithms for spectrally sparse signal reconstruction via low-rank Hankel matrix completion," *Applied and Computational Harmonic Analysis*, vol. 46, no. 1, pp. 94–121, 2019.

[11] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.

[12] M. Fazel, *Matrix Rank Minimization with Applications*, Ph.D. Dissertation, Stanford University, Stanford, CA 94305, USA, 2002.

[13] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[14] N. J. Harvey, D. R. Karger, and S. Yekhanin, "The complexity of matrix completion," in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pp. 1103–1111, Philadelphia, PA; USA, January 2006.

[15] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[16] K. C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific Journal of Optimization*, vol. 6, pp. 615–640, 2010.

[17] Z. Lin, M. Chen, L. Wu, and Y. Ma, *The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices*, UIUC Technicial Report UIUL-ENG-09-2214, Champaign, IL, USA, 2010.

[18] C. Chen, B. He, and X. Yuan, "Matrix completion via an alternating direction method," *IMA Journal of Numerical Analysis*, vol. 32, no. 1, pp. 227–245, 2012.

[19] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 665–674, Palo Alto, CA, USA, June 2013.

[20] J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 417–429, 2016.

[21] R.-P. Wen and L.-X. Liu, "The two-stage iteration algorithms based on the shortest distance for low-rank matrix completion," *Applied Mathematics and Computation*, vol. 314, pp. 133–141, 2017.

[22] Y. Hu, D. B. Zhang, J. Liu, J. P. Ye, and X. F. He, "Accelerated singular value thresholding for matrix completion," in *Proceedings of the Eighteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, August 2012.

[23] L. Wang, J. Hu, and C. Chen, "On accelerated singular value thresholding algorithm for matrix completion," *Applied Mathematics*, vol. 05, no. 21, pp. 3445–3451, 2014.

[24] R.-P. Wen and X.-H. Yan, "A new gradient projection method for matrix completion," *Applied Mathematics and Computation*, vol. 258, pp. 537–544, 2015.

[25] T. H. Oh, Y. Matsushita, Y. W. Tai, and I. S. Kweon, "Fast randomized singular value thresholding for low-rank optimization," *Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, pp. 376–391, 2015.

[26] Y. P. Song, J. A. Westerhuis, and A. K. Smilde, "Logistic Principal Component Analysis via Non-convex Singular

Value Thresholding," 2019, https://arxiv.org/abs/1902.09486.

[27] S. F. Yeganli and R. Yu, "Image Inpainting via Singular Value Thresholding," in *Proceedings of the IEEE: Signal Processing and Communications Applications Conference*, Calgary, Canada, April 2013.

[28] E. J. Candès, C. A. Sing-Long, and J. D. Trzasko, "Unbiased risk estimates for singular value thresholding and spectral estimators," *Institute of Electrical and Electronics Engineers Transactions on Signal Processing*, vol. 61, no. 19, pp. 4643–4657, 2013.

[29] S. Chatterjee, "Matrix estimation by universal singular value thresholding," *The Annals of Statistics*, vol. 43, no. 1, pp. 177–214, 2015.

[30] D. Donoho and M. Gavish, "Minimax risk of matrix denoising by singular value thresholding," *The Annals of Statistics*, vol. 42, no. 6, pp. 2413–2440, 2014.

[31] A. Dutta, B. Gong, X. Li, and M. Shah, "Weighted singular value thresholding and its application to background estimation," *Journal of Machine Learning Research*, pp. 1–22, 2017.

[32] O. Klopp, "Matrix completion by singular value thresholding: sharp bounds," *Electronic Journal of Statistics*, vol. 9, no. 2, pp. 2348–2369, 2015.

[33] L. Ma and Y. Xu, "Received signal strength recovery in green WLAN indoor positioning system using singular value thresholding," *Sensors*, vol. 15, no. 1, pp. 1292–1311, 2015.

[34] H. Zhang, L. Z. Cheng, and W. Zhu, "A lower bound guaranteeing exact matrix completion via singular value thresholding algorithm," *Applied and Computational Harmonic Analysis*, vol. 31, no. 3, pp. 454–459, 2011.

[35] X. Q. Zhang, Z. Y. Zhou, D. Wang, and Y. Ma, "Hybrid singular value thresholding for tensor completion," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1362–1368, Québec, Canada, July 2014.