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

¹Department of Applied Mathematics, Taiyuan University, Taiyuan 030600, China
²Key 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

Copyright © 2020 Yong-Hong Duan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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

\[
\min_{X \in \mathbb{R}^{m \times n}} \text{rank}(X),
\]

s.t. \( \rho_\Omega (X) = \rho_\Omega (M) \),

where the matrix \( M \in \mathbb{R}^{m \times n} \) is the underlying matrix to be reconstructed and \( \rho_\Omega \) 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 \( \Omega \) 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 Recht [13] switched (1) to another simple convex optimization problem (2) as follows:

\[
\min_{X \in \mathbb{R}^{m \times n}} \|X\|_*
\]

s.t. \( \rho_\Omega (X) = \rho_\Omega (M) \),

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]:

\[
\min_{X \in \mathbb{R}^{m \times n}} r\|X\|_* + \frac{1}{2} \|X\|_F^2
\]

s.t. \( \rho_\Omega (X) = \rho_\Omega (M) \).
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 $X \in \mathbb{R}^{m \times n}, Y \in \mathbb{R}^{n \times r}$ and rank $(M) = r$, and the singular value thresholding (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 algorithm which obtained widespread attention [25–27]. (J)he vari-

\[ \text{Algorithm 1.} \quad \min_{X \in \mathbb{R}^{m \times n}} \left\{ \frac{1}{2} \| (M - X) W \|_F^2 + \tau \| X \|_2 \right\}, \]  

with the nonsingular weight matrix $W \in \mathbb{R}^{m \times n}$ 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. $\mathbb{R}^{m \times n}$ denotes $m \times n$ real matrices set, $\mathbb{R}^{m \times n}$ represents the $m \times n$ nonnegative and real matrices’ set. The nuclear norm $\| X \|_n$ of a matrix $X$ is defined by $\| X \|_n = \sum_{k=1}^r \sigma_k(X)$, $\sigma_k(X)$ denotes the $k$th largest singular value of the real matrix $X \in \mathbb{R}^{m \times n}$ of rank $r$, and the Frobenius norm $\| X \|_F$ of a matrix $X = (x_{ij}) \in \mathbb{R}^{m \times n}$ is $\| X \|_F = (\sum_{i=1}^m \sum_{j=1}^n |x_{ij}|^2)^{1/2} = \sqrt{\text{trace}(X^T X)}$. $X^T$ 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 $\Omega^c$ is the complementary set of $\Omega$. $\rho_\Omega$ is the orthogonal projector onto $\Omega^c$, satisfying,

\[ \rho_\Omega(X) = \begin{cases} X_{ij}, & (i, j) \in \Omega, \\ 0, & (i, j) \notin \Omega. \end{cases} \]  

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 $X \in \mathbb{R}^{m \times n}$ of rank $r$ is

\[ X = U \Sigma V^T, \]

\[ \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r), \]

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ 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”:

\[ \mathcal{D}_\tau(X) = U \mathcal{D}_\tau(\Sigma) V^T, \]

\[ \mathcal{D}_\tau(\Sigma) = \text{diag}(\{\sigma_i - \tau\}_{i}), \]

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

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.
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 \( D = \rho_Ω (X^k) \) for short. The difference with the standard SVT algorithm may seem at \( k \) th step, replacing the iteration matrix \( X^k \) with the diagonal-update \( D W_k \) of its projector \( D \), where \( W_k = \text{diag}(w_1, w_2, \ldots, w_m) \in \mathbb{R}^{m \times m} \) is obtained by

\[
W_k = \arg \min_W \| \rho_Ω (M) - D W \|_F. \tag{11}
\]

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

\[
w_{ij}^{(k)} = \frac{\langle M(:, j), X_k(:, j) \rangle}{\langle X_k(:, j), X_k(:, j) \rangle}, \quad j = 1, 2, \ldots, n. \tag{12}
\]

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 \( \left\{ \rho_Ω (X^k) (I - W_k) \right\}_{k=1}^{\infty} \) is uniformly bounded. Then,

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

---

**Input:** sampled set \( \Omega \) and sampled entries \( \rho_Ω (M) \), step size \( \delta \in (0, 2) \), tolerance \( \varepsilon \), parameter \( \tau > 0 \), increment \( \ell \), and maximum iteration count \( k_{\max} \).

**Output:** \( X^{opt} \)

**Description:** recover a low-rank matrix \( M \) from a subset of sampled entries

1. Set \( Y^0 = k_0 \delta \rho_Ω (M) \), \( k_0 \) is an integer with \( (\tau/\delta \rho_Ω (M)) \in (k_0 - 1, k_0) \)
2. Set \( r_0 = 0 \)
3. for \( k = 1 \) to \( k_{\max} \)
4. Set \( s_k = r_{k-1} + 1 \)
5. repeat
6. Compute \([U^{k-1}, \Sigma^{k-1}, V^{k-1}]_\Omega \)
7. Set \( s_k = s_k + \ell \)
8. until \( \sigma_{s_k} \leq \tau \)
9. Set \( r_k = \max \{ j : \sigma_j > \tau \} \)
10. Set \( Z^k = \sum_{j=1}^{r_k} (\sigma_j - \tau) u_j \Sigma_j^{-1} v_j^T \)
11. if \( \| \rho_Ω (Y^k - M) \|_F < \varepsilon \| \rho_Ω (M) \|_F \) then break
12. Set \( Y^k = \begin{cases} \frac{Y^{k-1}}{\delta} + \delta (M - X^k), & \text{if } (i, j) \in \Omega, \\ 0, & \text{if } (i, j) \notin \Omega \end{cases} \)
13. end for \( k \)
14. Set \( X^{opt} = X^k \)

---

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

\[
\mathcal{L} (X, Y) = \tau \| X \|_* + \langle Y, \rho_Ω (M - X) \rangle + \frac{1}{2} \| Y \|_F^2, \tag{8}
\]

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

\[
h(Y) = -f(Y) = -\left( \tau \| D \| (\rho_Ω (Y)) \|_* + \langle Y, \rho_Ω (M - X^k) \rangle - \sum_{i,j} \rho_Ω (Y)) \|_* \right) + \frac{1}{2} \| D (\rho_Ω (Y)) \|_F^2, \tag{9}
\]

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

\[
\min_{Y \in \mathbb{R}^{m \times n}} h(Y). \tag{10}
\]

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 al. [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
Mathematical Problems in Engineering

Input: $\bar{\mu}, \alpha_{-1} = 0.5, Y_0 = Y_{-1} = Y_0, L_{-1} = L_0, \gamma_0 \geq \bar{\mu}, \lambda_0 = 1$

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 $L_k \alpha_k^2 = (1 - \alpha_k)\gamma_k + \alpha_k \bar{\mu}$
(4) Compute $Y_{k+1} = (1 - \alpha_k)Y_k + \alpha_k \bar{\mu}$; $\beta_k = (\gamma_k (1 - \alpha_{k-1})/\alpha_{k-1} (\gamma_k + L_k \alpha_k))$
(5) Compute $S_k = Y_k + \beta_k (Y_k - Y_{k-1})$
(6) Compute $L_{k+1} = S_k = (1/L_k)h' (S_k)$
(7) if $h(Y_{k+1}) \leq h(S_k) - (1/2L_k)\|h' (S_k)\|^2_F$ then
(8) go to Line 13
(9) else
(10) $L_k = 2L_k$
(11) end if
(12) end while
(13) Set $\omega = 2L_k (h(S_k) - h(Y_{k+1})/\|h' (S_k)\|^2_F), L_{k+1} = h(\omega)L_k, h(\omega) = \begin{cases} 1, & \text{if } 1 \leq \omega \leq 5, \\ 0.8, & \text{if } \omega > 5 \end{cases}$
(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: $Y_{N+1}, X_{N+1} = \mathcal{D}_\eta (\rho_\Omega (Y_{N+1}))$

Description: recover a low-rank matrix $Y$

(1) for $k = 1, 2, \ldots, N$ do
(2) Compute $k = 1, 2, \ldots, N$
(3) Compute $S_k = Y_k + Y_k - Y_{k-1}$
(4) while 1 do
(5) Compute $Y_{k+1} = S_k = (1/L_k)h' (S_k)$
(6) if $h(Y_{k+1}) \leq h(S_k) - (1/2L_k)\|h' (S_k)\|^2_F$ 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 $t_k = (1 + \sqrt{1 + 4\lambda_{k-1}^2})/2$
(13) end for

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

\[
\lim_{k \to \infty} \rho_\Omega (M - X^k) = \lim_{k \to \infty} \rho_\Omega (M - \bar{X}^k) = 0. \tag{13}
\]

Proof. It follows from Algorithm 4 that

\[
Y^k - \rho_\Omega (M) = Y^{k-1} - \rho_\Omega (M) + \delta_k \left(\rho_\Omega (M) - \rho_\Omega (X^k)\right)
= (1 - \delta_k) \left(\rho_\Omega (M) - \rho_\Omega (X^k)\right) + \delta_k \left(Y^{k-1} - \rho_\Omega (X^k)\right)
= (1 - \delta_k) \left(Y^{k-1} - \rho_\Omega (M)\right) + \delta_k \left(Y^{k-1} - \rho_\Omega (X^k)\right) + \rho_\Omega (X^k - \rho_\Omega (X^k))
= (1 - \delta_k) \left(Y^{k-1} - \rho_\Omega (M)\right) + \delta_k \left(Y^{k-1} - \rho_\Omega (X^k)\right) + \delta_k \rho_\Omega (X^k - X^k). \tag{14}
\]
In term of the assumption of this theorem, \( \forall k, \exists \Theta \) such that
\[
\left\| \rho_{\Omega}(\tilde{X}^k)(I-W_k) \right\|_F \leq \Theta
\] (15)
holds true.

Let \( Z_k = Y^k - \rho_{\Omega}(M) \). We note that \( X^k = \rho_{\Omega}(\tilde{X}_k)W_k \)
from Algorithm 4, and by substituting the following inequalities
\[
\left\| \rho_{\Omega}(Y^{k-1} - X^k) \right\|_F \leq \left\| Y^{k-1} - X^k \right\|_F \leq \sqrt{n}r,
\]
\[
\left\| \rho_{\Omega}(\tilde{X}^k - X^k) \right\|_F = \left\| \rho_{\Omega}(\tilde{X}^k (I-W_k)) \right\|_F \leq \Theta,
\] (16)
we have
\[
\|Z_k\|_F \leq |1 - \delta_k| \|Z_k\|_F + \delta_k \sqrt{n}r + \delta_k \Theta \leq \ldots
\]
\[
\leq \sum_{i=1}^{k} |1 - \delta_i| \|Z_i\|_F + \sum_{i=1}^{k} \prod_{j=1}^{i} |1 - \delta_j| (\delta_j \sqrt{n}r + \delta_j \Theta)
\leq c^k \|Z_0\|_F + \sum_{i=1}^{k} c^i \delta_i (\sqrt{n}r + M), \ \text{if} \ |1 - \delta|
\leq c < 1 \leq c^k \|Z_0\|_F + \frac{2}{1-c} (n\tau + M).
\] (17)

Hence, \( \{Z_k\}_{k=1}^{\infty} \) is bounded and so is \( \{Y^k\}_{k=1}^{\infty} \).

Moreover, it is obtained that
\[
\lim_{i \to \infty} \rho_{\Omega}(M - X^i) = 0,
\] (18)
from the equation
\[
Y^k = \sum_{i=1}^{k} \delta_i (\rho_{\Omega}(M - X^i)).
\] (19)

Moreover,
\[
\lim_{i \to \infty} \rho_{\Omega}(M - \tilde{X}^i) = 0,
\] (20)
since \( \lim_{k \to \infty} W_k = I \). The theorem has been proved. \( \square \)

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

**Proof.** It is obtained that \( \tilde{X}_k \) 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
\[
\tau \|X\|_\ast + \frac{1}{2} \|X\|_F^2 \geq \mathscr{L}(\tilde{X}_k, Y^{k-1})
\geq \lim_{k \to \infty} \left( \mathscr{L}(\tilde{X}_k, Y^{k-1}) \right)
= \mathscr{L}(X_{t,\tau}, Y_{t,\tau})
= \tau \|X_{t,\tau}\|_\ast + \frac{1}{2} \|X_{t,\tau}\|_F^2.
\] (21)

Thus, \( X_{t,\tau} \) is the unique solution of (3). \( \square \)

---

**Algorithm 4:** Singular value thresholding with diagonal-update (D-SVT) algorithm, Algorithm 1 of [15]
Table 1: Computational results for small-size problems when $\tau = 5n$.

| Unknown $M$ | Rank ($\tau$) | $p$ | 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  | 1.9793e-04 | 3.9195e-04 |
|            |               |     | SVT       | 255  | 88.2853  | 1.9663e-04 | 3.6080e-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  | 1.9705e-04 | 3.0682e-04 |
|            |               |     | SVT       | 271  | 83.7704  | 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 | 1.9687e-04 | 3.1833e-04 |
|            |               |     | 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 | 1.9892e-04 | 3.1476e-04 |
|            |               |     | 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   | 1.9885e-04 | 3.2650e-04 |
|            |               |     | SVT       | 267  | 2692.5   | 1.9690e-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   | 1.9943e-04 | 3.2126e-04 |
|            |               |     | 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   | 1.9611e-04 | 3.1200e-04 |
|            |               |     | 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   | 1.9823e-04 | 3.2674e-04 |
|            |               |     | SVT       | 271  | 6184.3   | 1.9895e-04 | 3.2622e-04 |
|            |               |     | ASVT      | 276  | 6155     | 1.1012e-04 | 2.1170e-04 |
|            |               |     | M-ASVT    | 269  | 6099     | 1.9863e-04 | 2.0208e-04 |
|            |               |     | D-SVT     | 180  | 5289.6   | 1.9689e-04 | 3.2015e-04 |
|            |               |     | 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   | 1.9802e-04 | 3.1833e-04 |

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

**Proof.** Note that $X_i$ is the optimal solution of the optimization problem (2) if and only if

$$0 \in \mathcal{R}_{\Pi}(\partial \|X_i\|_*).$$

(22)

From Theorem 2, we have

$$0 \in \mathcal{R}_{\Pi}(\tau \partial \|X_{i,\tau}\|_* + X_{i,\tau}),$$

(23)

which implies that

$$0 \in \mathcal{R}_{\Pi}(\partial \|X_{i,\tau}\|_* + \frac{1}{\tau}X_{i,\tau}).$$

(24)

Therefore,

$$\lim_{\tau \to \infty} \frac{1}{\tau}X_{i,\tau} = 0,$$

(25)

since $X_i$ is bounded. Thus,

$$0 \in \mathcal{R}_{\Pi}(\partial \|X_i\|_*).$$

(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 \( M \) | Size (n) | Rank (r) | \( p \) | Algorithm | IT | Time (s) | Error 1 | Error 2 |
|----------------|---------|---------|-------|-----------|----|---------|---------|---------|
|                |         |         |       |           |    |         |         |         |
|                | 1,000   | 20      | 0.20  | SVT       | 223| 76.3553 | 1.9830e-04 | 3.2593e-04 |
|                |         |         |       | ASVT      | 206| 71.2289 | 1.0054e-04 | 1.9965e-04 |
|                |         |         |       | M-ASVT    | 194| 69.5563 | 1.1023  | 1.9877e-04 |
|                |         |         |       | D-SVT     | 156| 66.9487 | 1.9748 e-04 | 3.2191 e-04 |
|                |         |         |       | SVT       | 207| 70.3992 | 1.9988e-04 | 3.1406e-04 |
|                |         |         |       | ASVT      | 203| 68.2241 | 1.1145e-04 | 2.1365e-04 |
|                |         |         |       | M-ASVT    | 192| 63.2289 | 1.0089e-04 | 2.0005e-04 |
|                |         |         |       | D-SVT     | 144| 61.4286 | 1.9333 e-04 | 3.0025 e-04 |
|                |         |         |       | SVT       | 220| 712.9493 | 1.9956e-04 | 3.2718e-04 |
|                |         |         |       | ASVT      | 217| 704.3365 | 1.0203e-04 | 2.1475e-04 |
|                |         |         |       | M-ASVT    | 211| 699.3785 | 1.0057e-04 | 2.0624e-04 |
|                |         |         |       | D-SVT     | 152| 535.2846 | 1.9969 e-04 | 3.2316 e-04 |
|                |         |         |       | SVT       | 212| 644.3027 | 1.9890e-04 | 3.1802e-04 |
|                |         |         |       | ASVT      | 205| 634.9870 | 1.1110e-04 | 2.3875e-04 |
|                |         |         |       | M-ASVT    | 199| 618.9952 | 1.0074e-04 | 2.2271e-04 |
|                | 2,000   | 30      | 0.15  | SVT       | 148| 522.8293 | 1.9669 e-04 | 3.1650 e-04 |
|                |         |         |       | ASVT      | 218| 2213.4  | 1.3354e-04 | 2.6522e-04 |
|                |         |         |       | M-ASVT    | 211| 2168.8  | 1.1532e-04 | 2.3301e-04 |
|                |         |         |       | D-SVT     | 158| 1824.8  | 1.9718 e-04 | 3.2220 e-04 |
|                |         |         |       | SVT       | 216| 2222.4  | 1.9616e-04 | 3.1860e-04 |
|                |         |         |       | ASVT      | 207| 2025.3  | 1.1004e-04 | 2.3110e-04 |
|                |         |         |       | M-ASVT    | 201| 1999.9  | 1.2121e-04 | 2.5403e-04 |
|                |         |         |       | D-SVT     | 152| 1807.7  | 1.9394 e-04 | 3.1228 e-04 |
|                |         |         |       | SVT       | 211| 2234.1  | 1.9597e-04 | 3.1398e-04 |
|                |         |         |       | ASVT      | 204| 2106.4  | 1.1140e-04 | 2.5510e-04 |
|                |         |         |       | M-ASVT    | 198| 1999.9  | 1.0042e-04 | 2.1030e-04 |
|                | 3,000   | 30      | 0.10  | SVT       | 148| 1733.6  | 1.9507 e-04 | 3.1039 e-04 |
|                |         |         |       | ASVT      | 224| 5314.0  | 1.9728e-04 | 3.2816e-04 |
|                |         |         |       | M-ASVT    | 218| 5127.1  | 1.2544e-04 | 2.3365e-04 |
|                |         |         |       | D-SVT     | 160| 4324.1  | 1.9500 e-04 | 3.2075 e-04 |
|                |         |         |       | SVT       | 218| 5209.7  | 1.9845e-04 | 3.2463e-04 |
|                |         |         |       | ASVT      | 211| 5101.0  | 1.1140e-04 | 2.0407e-04 |
|                |         |         |       | M-ASVT    | 205| 4998.2  | 1.0045e-04 | 1.8854e-04 |
|                |         |         |       | D-SVT     | 155| 4237.7  | 1.9460 e-04 | 3.1656 e-04 |
|                |         |         |       | SVT       | 215| 5114.3  | 1.9847e-04 | 3.2178e-04 |
|                |         |         |       | ASVT      | 207| 5003.5  | 1.6652e-04 | 2.3310e-04 |
|                |         |         |       | M-ASVT    | 200| 4889.2  | 1.1143e-04 | 2.1113e-04 |
|                |         |         |       | D-SVT     | 151| 4068.9  | 1.9986 e-04 | 3.2133 e-04 |

\[
\text{Error 1} = \left\| X_k - \rho_\Omega (M) \right\|_F, \\
\text{Error 2} = \left\| X_k W_k - M \right\|_F.
\]

(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 \( M \in \mathbb{R}^{n \times n} \) is square and that one has available \( s \) sampled entries \( \{ M_{ij}; (i, j) \in \Omega \} \), where \( \Omega \) 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], \( \delta = 1.2 p^{-1} \) and the step sizes are constant and the parameter \( \tau \) is chosen empirically. And as presented earlier [23], the parameters \( \tau_1 = 1 \) and \( \eta = 0.8 \). As for D-SVT algorithm, we choose \( \delta = 1.68 p^{-1} \), and \( \tau \) 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 $M$ | Rank (r) | $p$ | Algorithm | IT | Computational results |
|-------------|----------|-----|-----------|----|-----------------------|
|             |          |     |           |    | Time (s) | Error 1 | Error 2 |
| 1,000       | 20       | 0.198 | SVT       | 172 | 58.9324 | 1.9708e-04 | 3.2399e-04 |
|             |          |     | ASVT      | 165 | 55.3211 | 1.2114e-04 | 2.1140e-04 |
|             |          |     | M-ASVT    | 160 | 51.3333 | 1.1002e-04 | 2.0047e-04 |
|             |          |     | D-SVT     | 119 | 50.8026 | 1.9333e-04 | 3.1028e-04 |
|             |          |     | SVT       | 158 | 55.6121 | 1.9651e-04 | 3.0783e-04 |
|             |          |     | ASVT      | 150 | 51.3361 | 1.2254e-04 | 2.5512e-04 |
|             |          |     | M-ASVT    | 143 | 48.2210 | 1.1102e-04 | 2.0005e-04 |
| 1,000       | 30       | 0.2955 | D-SVT     | 110 | 48.3159 | 1.9430e-04 | 3.0057e-04 |
|             |          |     | SVT       | 169 | 509.3663 | 1.9555e-04 | 3.1915e-04 |
|             |          |     | ASVT      | 161 | 500.2369 | 1.2323e-04 | 1.9985e-04 |
|             |          |     | M-ASVT    | 156 | 498.2323 | 1.1110e-04 | 1.8541e-04 |
| 2,000       | 30       | 0.1489 | D-SVT     | 117 | 410.1959 | 1.9399e-04 | 3.1251e-04 |
|             |          |     | SVT       | 163 | 491.4569 | 1.9544e-04 | 3.1199e-04 |
|             |          |     | ASVT      | 157 | 482.3145 | 1.3324e-04 | 2.5666e-04 |
|             |          |     | M-ASVT    | 152 | 469.2221 | 1.1142e-04 | 2.1104e-04 |
| 2,000       | 40       | 0.198 | D-SVT     | 113 | 399.2143 | 1.9981e-04 | 3.1611e-04 |
|             |          |     | SVT       | 173 | 1810.3 | 1.9523e-04 | 3.2420e-04 |
|             |          |     | ASVT      | 166 | 1796.2 | 1.5562e-04 | 2.0004e-04 |
|             |          |     | M-ASVT    | 159 | 1652.8 | 1.1124e-04 | 1.8989e-04 |
| 3,000       | 30       | 0.0995 | D-SVT     | 119 | 1386.7 | 1.9967e-04 | 3.2635e-04 |
|             |          |     | SVT       | 166 | 1713.3 | 1.9533e-04 | 3.1671e-04 |
|             |          |     | ASVT      | 159 | 1695.2 | 1.5252e-04 | 2.8485e-04 |
|             |          |     | M-ASVT    | 154 | 1666.3 | 1.2235e-04 | 2.1103e-04 |
| 3,000       | 40       | 0.1324 | D-SVT     | 116 | 1355.3 | 1.9953e-04 | 3.2249e-04 |
|             |          |     | SVT       | 162 | 1664.9 | 1.9418e-04 | 3.1127e-04 |
|             |          |     | ASVT      | 157 | 1662.3 | 1.6523e-04 | 2.1104e-04 |
|             |          |     | M-ASVT    | 157 | 1660.2 | 1.0023e-04 | 2.1001e-04 |
| 3,000       | 50       | 0.1653 | D-SVT     | 113 | 1333.3 | 1.9641e-04 | 3.1282e-04 |
|             |          |     | SVT       | 174 | 4112.2 | 1.9442e-04 | 3.2322e-04 |
|             |          |     | ASVT      | 170 | 4000.3 | 1.1115e-04 | 2.1104e-04 |
|             |          |     | M-ASVT    | 168 | 3995.2 | 1.0002e-04 | 1.9952e-04 |
| 4,000       | 30       | 0.0747 | D-SVT     | 120 | 3247.9 | 1.9820e-04 | 3.2760e-04 |
|             |          |     | SVT       | 168 | 4023.7 | 1.9513e-04 | 3.1937e-04 |
|             |          |     | ASVT      | 163 | 3998.2 | 1.6674e-04 | 2.5510e-04 |
|             |          |     | M-ASVT    | 159 | 3885.1 | 1.3114e-04 | 2.0047e-04 |
| 4,000       | 40       | 0.0995 | D-SVT     | 117 | 3056.1 | 1.9684e-04 | 3.1983e-04 |
|             |          |     | SVT       | 165 | 3803.5 | 1.9604e-04 | 3.1727e-04 |
|             |          |     | ASVT      | 160 | 3685.2 | 1.2246e-04 | 2.8525e-04 |
|             |          |     | M-ASVT    | 156 | 3459.2 | 1.1006e-04 | 2.1139e-04 |
| 4,000       | 50       | 0.1242 | D-SVT     | 115 | 3104.3 | 1.9875e-04 | 3.1965e-04 |

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

| Unknown $M$ | Rank (r) | $p$ | Algorithm | IT | Computational results |
|-------------|----------|-----|-----------|----|-----------------------|
|             |          |     |           |    | Time (s) | Error 1 | Error 2 |
| 5,000       | 40       | 0.08 | SVT       | 271 | 12334 | 1.9826e-04 | 3.2645e-04 |
|             |          |     | ASVT      | 266 | 12301 | 1.2225e-04 | 2.1147e-04 |
|             |          |     | M-ASVT    | 264 | 12292 | 1.0558e-04 | 2.0036e-04 |
|             |          |     | D-SVT     | 190 | 9845.8 | 1.9556e-04 | 3.1916e-04 |
|             |          |     | SVT       | 265 | 11505 | 1.9898e-04 | 3.2434e-04 |
|             |          |     | ASVT      | 263 | 11427 | 1.2558e-04 | 3.0001e-04 |
|             |          |     | M-ASVT    | 260 | 11582 | 1.1010e-04 | 2.5563e-04 |
| 5,000       | 50       | 0.1 | D-SVT     | 186 | 9312.2 | 1.9976e-04 | 3.2374e-04 |
### Table 4: Continued.

| Unknown $M$ | Rank ($r$) | $p$ | Algorithm | $\tau$ | Time (s) | Error 1 | Error 2 |
|-------------|------------|-----|-----------|-------|----------|---------|--------|
| Size ($n$)  |            |     |           |       |          |         |        |
| 5,000       | 60         | 0.12| SVT       | 262   | 11983    | 1.9795e–04 | 3.1961e–04 |
|             |            |     | ASVT      | 259   | 11886    | 1.5252e–04 | 2.5474e–04 |
|             |            |     | M-ASVT    | 256   | 11963    | 1.1023e–04 | 2.1336e–04 |
|             |            |     | D-SVT     | 185   | 9656.4   | 1.9581e–04 | 3.1450e–04 |
|             |            |     | SVT       | 269   | 48699    | 1.9902e–04 | 3.2707e–04 |
|             |            |     | ASVT      | 263   | 49987    | 1.1298e–04 | 2.6634e–04 |
|             |            |     | M-ASVT    | 256   | 50000    | 1.0002e–04 | 1.8859e–04 |
|             |            |     | D-SVT     | 191   | 40793    | 1.9714e–04 | 3.2171e–04 |
| 8,000       | 50         | 0.06| SVT       | 266   | 46717    | 1.9926e–04 | 3.2572e–04 |
|             |            |     | ASVT      | 259   | 47558    | 1.3320e–04 | 2.0045e–04 |
|             |            |     | M-ASVT    | 251   | 48800    | 1.0223e–04 | 1.8879e–04 |
|             |            |     | D-SVT     | 189   | 37324    | 1.9591e–04 | 3.1842e–04 |
|             |            |     | SVT       | 258   | 48230    | 1.9750e–04 | 3.1624e–04 |
|             |            |     | ASVT      | 254   | 47889    | 1.2220e–04 | 2.1145e–04 |
|             |            |     | M-ASVT    | 250   | 47556    | 1.0023e–04 | 1.9987e–04 |
|             |            |     | D-SVT     | 183   | 38534    | 1.9695e–04 | 3.1436e–04 |
|             |            |     | SVT       | 250   | 43758    | 1.9707e–04 | 3.0923e–04 |
|             |            |     | ASVT      | 248   | 43665    | 1.3365e–04 | 2.1104e–04 |
|             |            |     | M-ASVT    | 245   | 43666    | 1.1102e–04 | 1.9965e–04 |
|             |            |     | D-SVT     | 177   | 35029    | 1.9794e–04 | 3.1001e–04 |
|             |            |     | SVT       | 247   | 85840    | 1.9976e–04 | 3.1183e–04 |
|             |            |     | ASVT      | 241   | 85662    | 1.2003e–04 | 2.0014e–04 |
|             |            |     | M-ASVT    | 238   | 85441    | 1.0041e–04 | 1.9941e–04 |
|             |            |     | D-SVT     | 176   | 69674    | 1.9795e–04 | 3.0853e–04 |
|             |            |     | SVT       | 215   | 71656    | 1.9760e–04 | 2.8117e–04 |
|             |            |     | ASVT      | 211   | 71221    | 1.2258e–04 | 2.0041e–04 |
|             |            |     | M-ASVT    | 207   | 71234    | 1.0005e–04 | 1.9595e–04 |
|             |            |     | D-SVT     | 155   | 59264    | 1.9531e–04 | 2.7785e–04 |
|             |            |     | SVT       | 242   | 139650   | 1.9701e–04 | 3.0300e–04 |
|             |            |     | ASVT      | 239   | 139666   | 1.2254e–04 | 2.3314e–04 |
|             |            |     | M-ASVT    | 234   | 139556   | 1.0005e–04 | 2.0101e–04 |
|             |            |     | D-SVT     | 173   | 128650   | 1.9664e–04 | 3.0193e–04 |
|             |            |     | SVT       | 195   | 110750   | 1.9964e–04 | 2.6726e–04 |
|             |            |     | ASVT      | 191   | 115544   | 1.2023e–04 | 1.8856e–04 |
|             |            |     | M-ASVT    | 187   | 115531   | 1.0006e–04 | 1.5654e–04 |
|             |            |     | D-SVT     | 141   | 97547    | 1.9785e–04 | 2.6492e–04 |

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

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

| Unknown $M$ | Rank ($r$) | $p$ | Algorithm | IT | Time (s) | Error 1 | Error 2 |
|-------------|------------|-----|-----------|----|----------|---------|---------|
| Size $(n)$  |            |     |           |    |          |         |         |
| 8,000       | 60         | 0.07| SVT       | 217| 38236    | 1.956e-04 | 3.1953e-04 |
|             |            |     | ASVT      | 214| 38001    | 1.2238e-04 | 2.3131e-04 |
|             |            |     | M-ASVT    | 208| 37996    | 1.0101e-04 | 2.0045e-04 |
|             |            |     | D-SVT     | 153| 30619    | 1.9923e-04 | 3.2345e-04 |
|             |            |     | 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    | 1.9913e-04 | 3.1791e-04 |
|             |            |     | 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    | 1.9832e-04 | 3.1065e-04 |
|             |            |     | 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    | 1.9315e-04 | 3.0102e-04 |
|             |            |     | 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 | 1.5611e-04 |
|             |            |     | D-SVT     | 126| 48758    | 1.9709e-04 | 2.8051e-04 |
|             |            |     | SVT       | 196| 11284    | 1.9603e-04 | 3.0148e-04 |
|             |            |     | ASVT      | -  | -        | -        | -        |
|             |            |     | M-ASVT    | -  | -        | -        | -        |
|             |            |     | D-SVT     | 131| 10375    | 1.9734e-04 | 3.0320e-04 |
|             |            |     | SVT       | 158| 88478    | 1.9661e-04 | 2.6335e-04 |
|             |            |     | ASVT      | -  | -        | -        | -        |
|             |            |     | M-ASVT    | -  | -        | -        | -        |
|             |            |     | D-SVT     | 115| 78069    | 1.9509e-04 | 2.6139e-04 |

Note: The mark '-' indicates that the iteration is failing.

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

| Unknown $M$ | Rank ($r$) | $p$ | Algorithm | IT | Time (s) | Error 1 | Error 2 |
|-------------|------------|-----|-----------|----|----------|---------|---------|
| Size $(n)$  |            |     |           |    |          |         |         |
| 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   | 1.9422e-04 | 3.1763e-04 |
|             |            |     | 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   | 1.9280e-04 | 3.1234e-04 |
|             |            |     | 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   | 1.9562e-04 | 3.1424e-04 |
|             |            |     | 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    | 1.9267e-04 | 3.1490e-04 |
|             |            |     | 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 | 1.5959e-04 |
|             |            |     | D-SVT     | 117| 23791    | 1.9287e-04 | 3.1360e-04 |
Table 6: Continued.

| Unknown M | Rank (r) | \( p \) | Algorithm | IT | Time (s) | Error 1 | Error 2 |
|-----------|----------|---------|-----------|-----|----------|---------|---------|
| Size (n)  |          |         |           |     |          |         |         |
| 8,000     | 100      | 0.1242  | SVT       | 160 | 30315    | 1.988e-04 | 3.185e-04 |
|           |          |         | ASVT      | 156 | 31256    | 1.200e-04 | 2.113e-04 |
|           |          |         | M-ASVT    | 149 | 33000    | 1.010e-04 | 1.858e-04 |
|           |          |         | D-SVT     | 115 | 23863    | 1.912e-04 | 3.056e-04 |
|           |          |         | SVT       | 155 | 27204    | 1.964e-04 | 3.084e-04 |
|           |          |         | ASVT      | 150 | 28693    | 1.220e-04 | 2.333e-04 |
|           |          |         | M-ASVT    | 144 | 29003    | 1.004e-04 | 2.014e-04 |
|           |          |         | D-SVT     | 112 | 22002    | 1.925e-04 | 3.015e-04 |
| 8,000     | 150      | 0.1857  | SVT       | 154 | 52658    | 1.936e-04 | 3.025e-04 |
|           |          |         | ASVT      | 153 | 53321    | 1.141e-04 | 2.300e-04 |
|           |          |         | M-ASVT    | 150 | 53662    | 1.005e-04 | 2.000e-04 |
|           |          |         | D-SVT     | 111 | 44680    | 1.997e-04 | 3.115e-04 |
| 10,000    | 200      | 0.1980  | SVT       | 133 | 44718    | 1.955e-04 | 2.785e-04 |
|           |          |         | ASVT      | 131 | 45512    | 1.322e-04 | 2.510e-04 |
|           |          |         | M-ASVT    | 126 | 45510    | 1.002e-04 | 1.588e-04 |
|           |          |         | D-SVT     | 109 | 38655    | 1.905e-04 | 2.713e-04 |
|           |          |         | SVT       | 150 | 86110    | 1.958e-04 | 3.013e-04 |
|           |          |         | ASVT      | -   | -        | -        | -        |
|           |          |         | M-ASVT    | -   | -        | -        | -        |
|           |          |         | D-SVT     | 109 | 81902    | 1.999e-04 | 3.073e-04 |
| 10,000    | 500      | 0.1980  | SVT       | 121 | 68906    | 1.923e-04 | 2.578e-04 |
|           |          |         | ASVT      | -   | -        | -        | -        |
|           |          |         | M-ASVT    | -   | -        | -        | -        |
|           |          |         | D-SVT     | 89  | 63517    | 1.912e-04 | 2.564e-04 |
| 12,000    | 300      | 0.2469  | SVT       | 121 | 68906    | 1.923e-04 | 2.578e-04 |
|           |          |         | ASVT      | -   | -        | -        | -        |
|           |          |         | M-ASVT    | -   | -        | -        | -        |
|           |          |         | D-SVT     | 89  | 63517    | 1.912e-04 | 2.564e-04 |
| 12,000    | 800      | 0.6444  | SVT       | 121 | 68906    | 1.923e-04 | 2.578e-04 |
|           |          |         | ASVT      | -   | -        | -        | -        |
|           |          |         | M-ASVT    | -   | -        | -        | -        |
|           |          |         | D-SVT     | 89  | 63517    | 1.912e-04 | 2.564e-04 |

Note. The mark ‘-’ indicates that the iteration is failing.
new algorithm is especially well suited for problems of very large sizes.

In order to show convergence behavior 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 matrix 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, “Uncovering shared structures in multiclass classification,” in *Proceedings 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. Mesbah 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 Recognition*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[7] M. BertiAmio, G. Sapiro, V. Caselles, and C. Ballester, “Multitask feature learning, image inpainting,” *GR Computer & Stationary*, vol. 34, pp. 417–424, 2000.

[8] J.-F. Cai, S. Liu, and W. Xu, “A fast algorithm for reconstruction of spectrally sparse signals in super-resolution,” in *Proceedings of the SPIE Optical Engineering + Applications*, p. 95970A, 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 Technical Report UIUC-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 algorithm 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. Matsuishi, 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
[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, Quebec, Canada, July 2014.