SIAM J. OPTIM.
Vol. 20, No. 4, pp. 1956–1982

A SINGULAR VALUE THRESHOLDING ALGORITHM FOR MATRIX COMPLETION*

JIAN-FENG CAI†, EMMANUEL J. CANDÈS‡, AND ZUOWEI SHEN§

Abstract. This paper introduces a novel algorithm to approximate the matrix with minimum nuclear norm among all matrices obeying a set of convex constraints. This problem may be understood as the convex relaxation of a rank minimization problem and arises in many important applications as in the task of recovering a large matrix from a small subset of its entries (the famous Netflix problem). Off-the-shelf algorithms such as interior point methods are not directly amenable to large problems of this kind with over a million unknown entries. This paper develops a simple first-order and easy-to-implement algorithm that is extremely efficient at addressing problems in which the optimal solution has low rank. The algorithm is iterative, produces a sequence of matrices $\{\boldsymbol{X}^k, \boldsymbol{Y}^k\}$, and at each step mainly performs a soft-thresholding operation on the singular values of the matrix $\boldsymbol{Y}^k$. There are two remarkable features making this attractive for low-rank matrix completion problems. The first is that the soft-thresholding operation is applied to a sparse matrix; the second is that the rank of the iterates $\{\boldsymbol{X}^k\}$ is empirically nondecreasing. Both these facts allow the algorithm to make use of very minimal storage space and keep the computational cost of each iteration low. On the theoretical side, we provide a convergence analysis showing that the sequence of iterates converges. On the practical side, we provide numerical examples in which $1,000 \times 1,000$ matrices are recovered in less than a minute on a modest desktop computer. We also demonstrate that our approach is amenable to very large scale problems by recovering matrices of rank about 10 with nearly a billion unknowns from just about 0.4% of their sampled entries. Our methods are connected with the recent literature on linearized Bregman iterations for ℓ_1 minimization, and we develop a framework in which one can understand these algorithms in terms of well-known Lagrange multiplier algorithms.

Key words. nuclear norm minimization, matrix completion, singular value thresholding, Lagrange dual function, Uzawa's algorithm, linearized Bregman iteration

AMS subject classifications. 90C25, 15A83, 65K05

DOI. 10.1137/080738970

1. Introduction.

1.1. Motivation.

There is a rapidly growing interest in the recovery of an unknown low-rank or approximately low-rank matrix from very limited information. This problem occurs in many areas of engineering and applied science such as machine learning [2, 3, 4], control [53], and computer vision; see [61]. As a motivating example, consider the problem of recovering a data matrix from a sampling of its entries. This routinely comes up whenever one collects partially filled out surveys, and one would like to infer the many missing entries. In the area of recommender systems, users submit ratings on a subset of entries in a database, and the vendor provides recommendations based on the user's preferences. Because users only rate a

*Received by the editors October 23, 2008; accepted for publication (in revised form) January 5, 2010; published electronically March 3, 2010.
http://www.siam.org/journals/siopt/20-4/73897.html

†Department of Mathematics, University of California, Los Angeles, CA 90095 (cai@math.ucla.edu). This author is supported by the Wavelets and Information Processing Programme under a grant from DSTA, Singapore.

‡Applied and Computational Mathematics, Caltech, Pasadena, CA 91125 (candes@stanford.edu). This author is partially supported by the Waterman Award from the National Science Foundation and by ONR grant N00014-08-1-0749.

§Department of Mathematics, National University of Singapore, 117543 Singapore (matzuows@nus.edu.sg). This author is supported in part by grant R-146-000-113-112 from the National University of Singapore.

few items, one would like to infer their preference for unrated items; this is the famous Netflix problem [1]. Recovering a rectangular matrix from a sampling of its entries is known as the *matrix completion* problem. The issue is, of course, that this problem is extraordinarily ill posed since with fewer samples than entries we have infinitely many completions. Therefore, it is apparently impossible to identify which of these candidate solutions is indeed the "correct" one without some additional information.

In many instances, however, the matrix we wish to recover has low rank or approximately low rank. For instance, the Netflix data matrix of all user ratings may be approximately low rank because it is commonly believed that only a few factors contribute to anyone's taste or preference. In computer vision, inferring scene geometry and camera motion from a sequence of images is a well-studied problem known as the structure-from-motion problem. This is an ill-conditioned problem for objects that may be distant with respect to their size, or especially for "missing data" which occur because of occlusion or tracking failures. However, when properly stacked and indexed, these images form a matrix which has very low rank (e.g., rank 3 under orthography) [24,61]. Other examples of low-rank matrix fitting abound; e.g., in control (system identification), machine learning (multiclass learning), and so on. Having said this, the premise that the unknown has (approximately) low rank radically changes the problem, making the search for solutions feasible since the lowest-rank solution now tends to be the right one.

In a recent paper [16], Candès and Recht showed that matrix completion is not as ill posed as people thought. Indeed, they proved that most low-rank matrices can be recovered *exactly* from most sets of sampled entries even though these sets have surprisingly small cardinality, and more importantly, they proved that this can be done by solving a simple *convex* optimization problem. To state their results, suppose to simplify that the unknown matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is square and that one has available m sampled entries $\{\boldsymbol{M}_{ij} : (i,j) \in \Omega\}$ where Ω is a random subset of cardinality m. Then [16] proves that most matrices $\boldsymbol{M}$ of rank r can be perfectly recovered by solving the optimization problem

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & X_{ij} = M_{ij}, \quad (i,j) \in \Omega, \end{array} \tag{1.1}$$

provided that the number of samples obeys $m \geq Cn^{6/5} r \log n$ for some positive numerical constant C.[1] In (1.1), the functional $\|\boldsymbol{X}\|_*$ is the nuclear norm of the matrix $\boldsymbol{M}$, which is the sum of its singular values. The optimization problem (1.1) is convex and can be recast as a semidefinite program [34,35]. In some sense, this is the tightest convex relaxation of the NP-hard rank minimization problem

$$\begin{array}{ll} \text{minimize} & \text{rank}(\boldsymbol{X}) \\ \text{subject to} & X_{ij} = M_{ij}, \quad (i,j) \in \Omega, \end{array} \tag{1.2}$$

since the nuclear ball $\{\boldsymbol{X} : \|\boldsymbol{X}\|_* \leq 1\}$ is the convex hull of the set of rank-one matrices with spectral norm bounded by one. Another interpretation of Candès and Recht's result is that under suitable conditions, the rank minimization program (1.2) and the convex program (1.1) are *formally equivalent* in the sense that they have exactly the same unique solution.

1.2. Algorithm outline. Because minimizing the nuclear norm both provably recovers the lowest-rank matrix subject to constraints (see [56] for related results) and

[1] Note that an $n \times n$ matrix of rank r depends upon $r(2n - r)$ degrees of freedom.

gives generally good empirical results in a variety of situations, it is understandably of great interest to develop numerical methods for solving (1.1). In [16], this optimization problem was solved using one of the most advanced semidefinite programming solvers, namely, SDPT3 [59]. This solver and others like SeDuMi are based on interior-point methods and are problematic when the size of the matrix is large because they need to solve huge systems of linear equations to compute the Newton direction. In fact, SDPT3 can only handle $n \times n$ matrices with $n \leq 100$. Presumably, one could resort to iterative solvers such as the method of conjugate gradients to solve for the Newton step, but this is problematic as well since it is well known that the condition number of the Newton system increases rapidly as one gets closer to the solution. In addition, none of these general purpose solvers use the fact that the solution may have low rank. We refer the reader to [50] for some recent progress on interior-point methods concerning some special nuclear norm-minimization problems.

This paper develops the *singular value thresholding* algorithm for approximately solving the nuclear norm minimization problem (1.1) and, by extension, problems of the form

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & \mathcal{A}(\boldsymbol{X}) = \boldsymbol{b}, \end{array} \tag{1.3}$$

where $\mathcal{A}$ is a linear operator acting on the space of $n_1 \times n_2$ matrices and $\boldsymbol{b} \in \mathbb{R}^m$. This algorithm is a simple first-order method and is especially well suited for problems of very large sizes in which the solution has low rank. We sketch this algorithm in the special matrix completion setting and let $\mathcal{P}_\Omega$ be the orthogonal projector onto the span of matrices vanishing outside of Ω so that the (i,j)th component of $\mathcal{P}_\Omega(\boldsymbol{X})$ is equal to X_{ij} if $(i,j) \in \Omega$ and zero otherwise. Our problem may be expressed as

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}), \end{array} \tag{1.4}$$

with optimization variable $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$. Fix $\tau > 0$ and a sequence $\{\delta_k\}_{k \geq 1}$ of scalar step sizes. Then starting with $\boldsymbol{Y}^0 = 0 \in \mathbb{R}^{n_1 \times n_2}$, the algorithm inductively defines

$$\begin{cases} \boldsymbol{X}^k = \text{shrink}(\boldsymbol{Y}^{k-1}, \tau), \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k) \end{cases} \tag{1.5}$$

until a stopping criterion is reached. In (1.5), $\text{shrink}(\boldsymbol{Y}, \tau)$ is a nonlinear function which applies a soft-thresholding rule at level τ to the singular values of the input matrix; see section 2 for details. The key property here is that for large values of τ, the sequence $\{\boldsymbol{X}^k\}$ converges to a solution which very nearly minimizes (1.4). Hence, at each step, one only needs to compute at most one singular value decomposition and perform a few elementary matrix additions. Two important remarks are in order:

1. *Sparsity.* For each $k \geq 0$, $\boldsymbol{Y}^k$ vanishes outside of Ω and is, therefore, sparse, a fact which can be used to evaluate the shrink function rapidly.
2. *Low-rank property.* The matrices $\boldsymbol{X}^k$ turn out to have low rank, and hence the algorithm has a minimum storage requirement since we need to keep only principal factors in memory.

Our numerical experiments demonstrate that the proposed algorithm can solve problems, in Matlab, involving matrices of size $30{,}000 \times 30{,}000$ having close to a billion unknowns in 17 minutes on a standard desktop computer with a 1.86 GHz CPU (dual core with Matlab's multithreading option enabled) and 3 GB of memory. As a consequence, the singular value thresholding algorithm may become a rather powerful computational tool for large scale matrix completion.

1.3. General formulation. The singular value thresholding algorithm can be adapted to deal with other types of convex constraints. For instance, it may address problems of the form

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & f_i(\boldsymbol{X}) \le 0, \quad i = 1, \dots, m, \end{array} \tag{1.6}$$

where each f_i is a Lipschitz convex function (note that one can handle linear equality constraints by considering pairs of affine functionals). In the simpler case where the f_i's are affine functionals, the general algorithm goes through a sequence of iterations which greatly resemble (1.5). This is useful because this enables the development of numerical algorithms which are effective for recovering matrices from a small subset of sampled entries possibly contaminated with noise.

1.4. Contents and notations. The rest of the paper is organized as follows. In section 2, we derive the singular value thresholding (SVT) algorithm for the matrix completion problem and recast it in terms of a well-known Lagrange multiplier algorithm. In section 3, we extend the SVT algorithm and formulate a general iteration which is applicable to general convex constraints. In section 4, we establish the convergence results for the iterations given in sections 2 and 3. We demonstrate the performance and effectiveness of the algorithm through numerical examples in section 5, and review additional implementation details. Finally, we conclude the paper with a short discussion in section 6.

Before continuing, we provide here a brief summary of the notations used throughout the paper. Matrices are bold capital, vectors are bold lowercase, and scalars or entries are not bold. For instance, $\boldsymbol{X}$ is a matrix and X_{ij} its (i,j)th entry. Likewise, $\boldsymbol{x}$ is a vector and x_i its ith component. The nuclear norm of a matrix is denoted by $\|\boldsymbol{X}\|_*$, the Frobenius norm by $\|\boldsymbol{X}\|_F$, and the spectral norm by $\|\boldsymbol{X}\|_2$; note that these are, respectively, the 1-norm, the 2-norm, and the sup-norm of the vector of singular values. The adjoint of a matrix $\boldsymbol{X}$ is $\boldsymbol{X}^*$ and similarly for vectors. The notation $\operatorname{diag}(\boldsymbol{x})$, where $\boldsymbol{x}$ is a vector, stands for the diagonal matrix with $\{x_i\}$ as diagonal elements. We denote by $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \operatorname{trace}(\boldsymbol{X}^*\boldsymbol{Y})$ the standard inner product between two matrices ($\|\boldsymbol{X}\|_F^2 = \langle \boldsymbol{X}, \boldsymbol{X} \rangle$). The Cauchy–Schwarz inequality gives $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \le \|\boldsymbol{X}\|_F \|\boldsymbol{Y}\|_F$, and it is well known that we also have $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \le \|\boldsymbol{X}\|_* \|\boldsymbol{Y}\|_2$ (the spectral and nuclear norms are dual from one another); see, e.g., [16, 56].

2. The singular value thresholding algorithm. This section introduces the singular value thresholding algorithm and discusses some of its basic properties. We begin with the definition of a key building block, namely, the singular value thresholding operator.

2.1. The singular value shrinkage operator. Consider the singular value decomposition (SVD) of a matrix $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ of rank r

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*, \quad \boldsymbol{\Sigma} = \operatorname{diag}(\{\sigma_i\}_{1 \le i \le r}), \tag{2.1}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are, respectively, $n_1 \times r$ and $n_2 \times r$ matrices with orthonormal columns, and the singular values σ_i are positive (unless specified otherwise, we will always assume that the SVD of a matrix is given in the reduced form above). For each $\tau \ge 0$, we introduce the soft-thresholding operator $\mathcal{D}_\tau$ defined as follows:

$$\mathcal{D}_\tau(\boldsymbol{X}) := \boldsymbol{U}\mathcal{D}_\tau(\boldsymbol{\Sigma})\boldsymbol{V}^*, \quad \mathcal{D}_\tau(\boldsymbol{\Sigma}) = \operatorname{diag}(\{\sigma_i - \tau)_+\}), \tag{2.2}$$

where t_+ is the positive part of t, namely, $t_+ = \max(0, t)$. In words, this operator simply applies a soft-thresholding rule to the singular values of $\boldsymbol{X}$, effectively shrinking these toward zero. This is the reason why we will also refer to this transformation as the *singular value shrinkage* operator. Even though the SVD may not be unique, it is easy to see that the singular value shrinkage operator is well defined, and we do not elaborate further on this issue. In some sense, this shrinkage operator is a straightforward extension of the soft-thresholding rule for scalars and vectors. In particular, note that if many of the singular values of $\boldsymbol{X}$ are below the threshold τ, the rank of $\mathcal{D}_\tau(\boldsymbol{X})$ may be considerably lower than that of $\boldsymbol{X}$, just like the soft-thresholding rule applied to vectors leads to sparser outputs whenever some entries of the input are below threshold.

The singular value thresholding operator is the proximity operator associated with the nuclear norm. Details about the proximity operator can be found in, e.g., [42].

THEOREM 2.1.[2] *For each $\tau \geq 0$ and $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$, the singular value shrinkage operator* (2.2) *obeys*

$$\mathcal{D}_\tau(\boldsymbol{Y}) = \arg\min_{\boldsymbol{X}} \ \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2 + \tau\|\boldsymbol{X}\|_*. \tag{2.3}$$

Proof. Since the function $h_0(\boldsymbol{X}) := \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_F^2$ is strictly convex, it is easy to see that there exists a unique minimizer, and we thus need to prove that it is equal to $\mathcal{D}_\tau(\boldsymbol{Y})$. To do this, recall the definition of a subgradient of a convex function $f : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}$. We say that $\boldsymbol{Z}$ is a subgradient of f at $\boldsymbol{X}_0$, denoted $\boldsymbol{Z} \in \partial f(\boldsymbol{X}_0)$, if

$$f(\boldsymbol{X}) \geq f(\boldsymbol{X}_0) + \langle \boldsymbol{Z}, \boldsymbol{X} - \boldsymbol{X}_0 \rangle \tag{2.4}$$

for all $\boldsymbol{X}$. Now $\hat{\boldsymbol{X}}$ minimizes h_0 if and only if $\boldsymbol{0}$ is a subgradient of the functional h_0 at the point $\hat{\boldsymbol{X}}$, i.e.,

$$\boldsymbol{0} \in \hat{\boldsymbol{X}} - \boldsymbol{Y} + \tau\partial\|\hat{\boldsymbol{X}}\|_*, \tag{2.5}$$

where $\partial\|\hat{\boldsymbol{X}}\|_*$ is the set of subgradients of the nuclear norm. Let $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ be an arbitrary matrix and $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$ be its SVD. It is known [16, 46, 64] that

$$\partial\|\boldsymbol{X}\|_* = \left\{\boldsymbol{U}\boldsymbol{V}^* + \boldsymbol{W} : \ \boldsymbol{W} \in \mathbb{R}^{n_1 \times n_2}, \ \ \boldsymbol{U}^*\boldsymbol{W} = 0, \ \ \boldsymbol{W}\boldsymbol{V} = 0, \ \ \|\boldsymbol{W}\|_2 \leq 1\right\}. \tag{2.6}$$

Set $\hat{\boldsymbol{X}} := \mathcal{D}_\tau(\boldsymbol{Y})$ for short. In order to show that $\hat{\boldsymbol{X}}$ obeys (2.5), decompose the SVD of $\boldsymbol{Y}$ as $\boldsymbol{Y} = \boldsymbol{U}_0\boldsymbol{\Sigma}_0\boldsymbol{V}_0^* + \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*$, where $\boldsymbol{U}_0$, $\boldsymbol{V}_0$ (resp., $\boldsymbol{U}_1$, $\boldsymbol{V}_1$) are the singular vectors associated with singular values greater than τ (resp., smaller than or equal to τ). With these notations, we have $\hat{\boldsymbol{X}} = \boldsymbol{U}_0(\boldsymbol{\Sigma}_0 - \tau\boldsymbol{I})\boldsymbol{V}_0^*$ and, therefore,

$$\boldsymbol{Y} - \hat{\boldsymbol{X}} = \tau(\boldsymbol{U}_0\boldsymbol{V}_0^* + \boldsymbol{W}), \quad \boldsymbol{W} = \tau^{-1}\boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^*.$$

By definition, $\boldsymbol{U}_0^*\boldsymbol{W} = 0$, $\boldsymbol{W}\boldsymbol{V}_0 = 0$, and since the diagonal elements of $\boldsymbol{\Sigma}_1$ have magnitudes bounded by τ, we also have $\|\boldsymbol{W}\|_2 \leq 1$. Hence $\boldsymbol{Y} - \hat{\boldsymbol{X}} \in \tau\partial\|\hat{\boldsymbol{X}}\|_*$, which concludes the proof. □

[2] One reviewer pointed out that a similar result had been mentioned in a talk given by Donald Goldfarb at the Foundations of Computational Mathematics conference which took place in Hong Kong in June 2008.

2.2. Shrinkage iterations. We are now in the position to introduce the singular value thresholding algorithm. Fix $\tau > 0$ and a sequence $\{\delta_k\}$ of positive step sizes. Starting with $\boldsymbol{Y}_0$, inductively define for $k = 1, 2, \ldots,$

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\boldsymbol{Y}^{k-1}), \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k) \end{cases} \tag{2.7}$$

until a stopping criterion is reached (we postpone the discussion of this stopping criterion and of the choice of step sizes). This shrinkage iteration is very simple to implement. At each step, we only need to compute an SVD and perform elementary matrix operations. With the help of a standard numerical linear algebra package, the whole algorithm can be coded in just a few lines. As we will see later, the iteration (2.7) is the linearized Bregman iteration, which is a special instance of Uzawa's algorithm.

Before addressing further computational issues, we would like to make explicit the relationship between this iteration and the original problem (1.1). In section 4, we will show that the sequence $\{\boldsymbol{X}^k\}$ converges to the unique solution of an optimization problem closely related to (1.1), namely,

$$\begin{array}{ll} \text{minimize} & \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2 \\ \text{subject to} & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}). \end{array} \tag{2.8}$$

Furthermore, it is intuitive that the solution to this modified problem converges to that of (1.4) as $\tau \to \infty$ as shown in section 3. Thus by selecting a large value of the parameter τ, the sequence of iterates converges to a matrix which nearly minimizes (1.1).

As mentioned earlier, there are two crucial properties which make this algorithm ideally suited for matrix completion.

- *Low-rank property.* A remarkable empirical fact is that the matrices in the sequence $\{\boldsymbol{X}^k\}$ have low rank (provided, of course, that the solution to (2.8) has low rank). We use the word "empirical" because all of our numerical experiments have produced low-rank sequences, but we cannot rigorously prove that this is true in general. The reason for this phenomenon is, however, simple: Because we are interested in large values of τ (as to better approximate the solution to (1.1)), the thresholding step happens to "kill" most of the small singular values and produces a low-rank output. In fact, our numerical results show that the rank of $\boldsymbol{X}^k$ is nondecreasing with k, and the maximum rank is reached in the last steps of the algorithm; see section 5.
  Thus, when the rank of the solution is substantially smaller than either dimension of the matrix, the storage requirement is low since we could store each $\boldsymbol{X}^k$ in its SVD form (note that we need to keep only the current iterate and may discard earlier values).
- *Sparsity.* Another important property of the SVT algorithm is that the iteration matrix $\boldsymbol{Y}^k$ is sparse. Since $\boldsymbol{Y}^0 = \boldsymbol{0}$, we have by induction that $\boldsymbol{Y}^k$ vanishes outside of Ω. The fewer entries available, the sparser $\boldsymbol{Y}^k$. Because the sparsity pattern Ω is fixed throughout, one can then apply sparse matrix techniques to save storage. Also, if $|\Omega| = m$, the computational cost of updating $\boldsymbol{Y}^k$ is of order m. Moreover, we can call subroutines supporting sparse matrix computations, which can further reduce computational costs.
  One such subroutine is the SVD. However, note that we do not need to compute the entire SVD of $\boldsymbol{Y}^k$ to apply the singular value thresholding operator.

Only the part corresponding to singular values greater than τ is needed. Hence, a good strategy is to apply the iterative Lanczos algorithm to compute the first few singular values and singular vectors. Because $\boldsymbol{Y}^k$ is sparse, $\boldsymbol{Y}^k$ can be applied to arbitrary vectors rapidly, and this procedure offers a considerable speedup over naive methods.

2.3. Relation with other works. Our algorithm is inspired by recent work in the area of ℓ_1 minimization, and especially by the work on linearized Bregman iterations for compressed sensing; see [11, 12, 13, 27, 55, 67] for linearized Bregman iterations and [17, 19, 20, 21, 30] for some information about the field of compressed sensing. In this line of work, linearized Bregman iterations are used to find the solution to an underdetermined system of linear equations with minimum ℓ_1 norm. In fact, Theorem 2.1 asserts that the singular value thresholding algorithm can be formulated as a linearized Bregman iteration. Bregman iterations were first introduced in [54] as a convenient tool for solving computational problems in the imaging sciences, and a later paper [67] showed that they were useful for solving ℓ_1-norm minimization problems in the area of compressed sensing. Linearized Bregman iterations were proposed in [27] to improve performance of plain Bregman iterations; see also [67]. Additional details together with a technique for improving the speed of convergence called *kicking* are described in [55]. On the practical side, the paper [13] applied Bregman iterations to solve a deblurring problem, while on the theoretical side, the references [11, 12] gave a rigorous analysis of the convergence of such iterations. New developments keep on coming out at a rapid pace, and recently, [39] introduced a new iteration, the *split Bregman iteration*, to extend Bregman-type iterations (such as linearized Bregman iterations) to problems involving the minimization of ℓ_1-like functionals such as total-variation norms, Besov norms, and so forth.

When applied to ℓ_1-minimization problems, linearized Bregman iterations are sequences of soft-thresholding rules operating on vectors. Iterative soft-thresholding algorithms in connection with ℓ_1 or total-variation minimization have quite a bit of history in signal and image processing, and we would like to mention the works [14, 48] for total-variation minimization, [28, 29, 36] for ℓ_1 minimization, and [5, 9, 10, 22, 23, 32, 33, 58] for some recent applications in the area of image inpainting and image restoration. Just as iterative soft-thresholding methods are designed to find sparse solutions, our iterative singular value thresholding scheme is designed to find a sparse vector of singular values. In classical problems arising in the areas of compressed sensing, and signal or image processing, the sparsity is expressed in a known transformed domain and soft-thresholding is applied to transformed coefficients. In contrast, the shrinkage operator $\mathcal{D}_\tau$ is adaptive. The SVT not only discovers a sparse singular vector but also the bases in which we have a sparse representation. In this sense, the SVT algorithm is an extension of earlier iterative soft-thresholding schemes.

Finally, we would like to contrast the SVT iteration (2.7) with the popular iterative soft-thresholding algorithm used in many papers in imaging processing and perhaps best known under the name of proximal forward-backward splitting method (PFBS); see [10, 26, 28, 36, 40, 62, 63], for example. The constrained minimization problem (1.4) may be relaxed into

$$\text{minimize} \quad \lambda\|\boldsymbol{X}\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{X}) - \mathcal{P}_\Omega(\boldsymbol{M})\|_F^2 \tag{2.9}$$

for some $\lambda > 0$. Theorem 2.1 asserts that $\mathcal{D}_\lambda$ is the proximity operator of $\lambda\|\boldsymbol{X}\|_*$ and Proposition 3.1(iii) in [26] gives that the solution to this unconstrained problem is

characterized by the fixed point equation $\boldsymbol{X} = \mathcal{D}_{\lambda\delta}(\boldsymbol{X} + \delta P_\Omega(\boldsymbol{M} - \boldsymbol{X}))$ for each $\delta > 0$. One can then apply a simplified version of the PFBS method (see (3.6) in [26]) to obtain iterations of the form $\boldsymbol{X}^k = \mathcal{D}_{\lambda\delta_{k-1}}(\boldsymbol{X}^{k-1} + \delta_{k-1} P_\Omega(\boldsymbol{M} - \boldsymbol{X}^{k-1}))$. Introducing an intermediate matrix $\boldsymbol{Y}^k$, this algorithm may be expressed as

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_{\lambda\delta_{k-1}}(\boldsymbol{Y}^{k-1}), \\ \boldsymbol{Y}^k = \boldsymbol{X}^k + \delta_k P_\Omega(\boldsymbol{M} - \boldsymbol{X}^k). \end{cases} \tag{2.10}$$

The difference with (2.7) may seem subtle at first—replacing $\boldsymbol{X}^k$ in (2.10) with $\boldsymbol{Y}^{k-1}$ and setting $\delta_k = \delta$ gives (2.7) with $\tau = \lambda\delta$—but has enormous consequences as this gives entirely different algorithms. First, they have different limits: While (2.7) converges to the solution of the constrained minimization (2.8), (2.10) converges to the solution of (2.9) provided that the sequence of step sizes is appropriately selected. Second, selecting a large λ (or a large value of $\tau = \lambda\delta$) in (2.10) gives a low-rank sequence of iterates and a limit with small nuclear norm. The limit, however, does not fit the data, and this is why one has to choose a small or moderate value of λ (or of $\tau = \lambda\delta$). However, when λ is not sufficiently large, the $\boldsymbol{X}^k$'s may not have low rank even though the solution has low rank (and one may need to compute many singular vectors), and thus applying the shrinkage operation accurately to $\boldsymbol{Y}^k$ may be computationally very expensive. Moreover, the limit does not necessary have a small nuclear norm. These are some of the reasons why (2.10) does not seem to be very suitable for very large scale matrix completion problems (in cases where computing the SVD is prohibitive). Since the original submission of this paper, however, we note that several papers proposed some working implementations [51, 60].

2.4. Interpretation as a Lagrange multiplier method. In this section, we recast the SVT algorithm as a type of Lagrange multiplier algorithm known as Uzawa's algorithm. An important consequence is that this will allow us to extend the SVT algorithm to other problems involving the minimization of the nuclear norm under convex constraints; see section 3. Further, another contribution of this paper is that this framework actually recasts linear Bregman iterations as a very special form of Uzawa's algorithm, hence providing fresh and clear insights about these iterations.

In what follows, we set $f_\tau(\boldsymbol{X}) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ for some fixed $\tau > 0$ and recall that we wish to solve (2.8)

$$\begin{array}{ll} \text{minimize} & f_\tau(\boldsymbol{X}) \\ \text{subject to} & \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M}). \end{array}$$

The Lagrangian for this problem is given by $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) = f_\tau(\boldsymbol{X}) + \langle \boldsymbol{Y}, \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X})\rangle$, where $\boldsymbol{Y} \in \mathbb{R}^{n_1 \times n_2}$. Strong duality holds and $\boldsymbol{X}^\star$ and $\boldsymbol{Y}^\star$ are primal-dual optimal if $(\boldsymbol{X}^\star, \boldsymbol{Y}^\star)$ is a saddlepoint of the Lagrangian $\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$, i.e., a pair obeying

$$\sup_{\boldsymbol{Y}} \inf_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}) = \mathcal{L}(\boldsymbol{X}^\star, \boldsymbol{Y}^\star) = \inf_{\boldsymbol{X}} \sup_{\boldsymbol{Y}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}). \tag{2.11}$$

The function $g_0(\boldsymbol{Y}) = \inf_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y})$ is called the dual function. Here, g_0 is continuously differentiable and has a gradient which is Lipschitz with Lipschitz constant at most one, as this is a consequence of well-known results concerning conjugate functions. Uzawa's algorithm approaches the problem of finding a saddlepoint with an iterative procedure. From $\boldsymbol{Y}_0 = \boldsymbol{0}$, say, inductively define

$$\begin{cases} \mathcal{L}(\boldsymbol{X}^k, \boldsymbol{Y}^{k-1}) = \min_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}^{k-1}), \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k), \end{cases} \tag{2.12}$$

where $\{\delta_k\}_{k\geq 1}$ is a sequence of positive step sizes. Uzawa's algorithm is, in fact, a subgradient method applied to the dual problem, where each step moves the current iterate in the direction of the gradient or of a subgradient. Indeed, observe that the gradient of $g_0(\boldsymbol{Y})$ is given by

$$\partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}) = \partial_{\boldsymbol{Y}} \mathcal{L}(\tilde{\boldsymbol{X}}, \boldsymbol{Y}) = \mathcal{P}_\Omega(\boldsymbol{M} - \tilde{\boldsymbol{X}}), \tag{2.13}$$

where $\tilde{\boldsymbol{X}}$ is the minimizer of the Lagrangian for that value of $\boldsymbol{Y}$ so that a gradient descent update for $\boldsymbol{Y}$ is of the form

$$\boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}^{k-1}) = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k).$$

It remains to compute the minimizer of the Lagrangian (2.12), and note that

$$\arg\min\ f_\tau(\boldsymbol{X}) + \langle \boldsymbol{Y}, \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X})\rangle = \arg\min\ \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X} - \mathcal{P}_\Omega \boldsymbol{Y}\|_F^2. \tag{2.14}$$

However, we know that the minimizer is given by $\mathcal{D}_\tau(\mathcal{P}_\Omega(\boldsymbol{Y}))$, and since $\boldsymbol{Y}^k = \mathcal{P}_\Omega(\boldsymbol{Y}^k)$ for all $k \geq 0$, Uzawa's algorithm takes the form

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\boldsymbol{Y}^{k-1}), \\ \boldsymbol{Y}^k = \boldsymbol{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k), \end{cases}$$

which is exactly the update (2.7). This point of view brings to bear many different mathematical tools for proving the convergence of the singular value thresholding iterations. For an early use of Uzawa's algorithm minimizing an ℓ_1-like functional, the total-variation norm, under linear inequality constraints, see [14].

3. General formulation. This section presents a general formulation of the SVT algorithm for approximately minimizing the nuclear norm of a matrix under convex constraints.

3.1. Linear equality constraints. Set the objective functional $f_\tau(\boldsymbol{X}) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ for some fixed $\tau > 0$, and consider the following optimization problem:

$$\begin{array}{ll} \text{minimize} & f_\tau(\boldsymbol{X}) \\ \text{subject to} & \mathcal{A}(\boldsymbol{X}) = \boldsymbol{b}, \end{array} \tag{3.1}$$

where $\mathcal{A}$ is a linear transformation mapping $n_1 \times n_2$ matrices into $\mathbb{R}^m$ ($\mathcal{A}^*$ is the adjoint of $\mathcal{A}$). This more general formulation is considered in [16] and [56] as an extension of the matrix completion problem. Then the Lagrangian for this problem is of the form

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{y}) = f_\tau(\boldsymbol{X}) + \langle \boldsymbol{y}, \boldsymbol{b} - \mathcal{A}(\boldsymbol{X})\rangle, \tag{3.2}$$

where $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ and $\boldsymbol{y} \in \mathbb{R}^m$, and starting with $\boldsymbol{y}^0 = \boldsymbol{0}$, Uzawa's iteration is given by

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\mathcal{A}^*(\boldsymbol{y}^{k-1})), \\ \boldsymbol{y}^k = \boldsymbol{y}^{k-1} + \delta_k(\boldsymbol{b} - \mathcal{A}(\boldsymbol{X}^k)). \end{cases} \tag{3.3}$$

The iteration (3.3) is, of course, the same as (2.7) in the case where $\mathcal{A}$ is a sampling operator extracting m entries with indices in Ω out of an $n_1 \times n_2$ matrix. To verify this claim, observe that in this situation, $\mathcal{A}^*\mathcal{A} = \mathcal{P}_\Omega$, and let $\boldsymbol{M}$ be any matrix obeying $\mathcal{A}(\boldsymbol{M}) = \boldsymbol{b}$. Then defining $\boldsymbol{Y}^k = \mathcal{A}^*(\boldsymbol{y}^k)$ and substituting this expression in (3.3) gives (2.7).

3.2. General convex constraints. One can also adapt the algorithm to handle general convex constraints. Suppose we wish to minimize $f_\tau(\boldsymbol{X})$ defined as before over a convex set $\boldsymbol{X} \in \mathcal{C}$. To simplify, we will assume that this convex set is given by $\mathcal{C} = \{\boldsymbol{X} : f_i(\boldsymbol{X}) \leq 0 \, \forall i = 1, \ldots, m\}$, where the f_i's are convex functionals (note that one can handle linear equality constraints by considering pairs of affine functionals). The problem of interest is then of the form

$$\begin{array}{ll} \text{minimize} & f_\tau(\boldsymbol{X}) \\ \text{subject to} & f_i(\boldsymbol{X}) \leq 0, \quad i = 1, \ldots, m. \end{array} \tag{3.4}$$

Just as before, it is intuitive that as $\tau \to \infty$, the solution to this problem converges to a minimizer of the nuclear norm under the same constraints (1.6) as shown in Theorem 3.1 at the end of this section.

Put $\mathcal{F}(\boldsymbol{X}) := (f_1(\boldsymbol{X}), \ldots, f_m(\boldsymbol{X}))$ for short. Then the Lagrangian for (3.4) is equal to $\mathcal{L}(\boldsymbol{X}, \boldsymbol{y}) = f_\tau(\boldsymbol{X}) + \langle \boldsymbol{y}, \mathcal{F}(\boldsymbol{X}) \rangle$, where $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ and $\boldsymbol{y} \in \mathbb{R}^m$ is now a vector with nonnegative components denoted, as usual, by $\boldsymbol{y} \geq \boldsymbol{0}$. One can apply Uzawa's method just as before with the only modification that we will use a subgradient method with projection to maximize the dual function since we need to make sure that the successive updates $\boldsymbol{y}^k$ belong to the nonnegative orthant. This gives

$$\begin{cases} \boldsymbol{X}^k = \arg\min \{ f_\tau(\boldsymbol{X}) + \langle \boldsymbol{y}^{k-1}, \mathcal{F}(\boldsymbol{X}) \rangle \}, \\ \boldsymbol{y}^k = [\boldsymbol{y}^{k-1} + \delta_k \mathcal{F}(\boldsymbol{X}^k)]_+. \end{cases} \tag{3.5}$$

Above, $\boldsymbol{x}_+$ is of course the vector with entries equal to $\max(x_i, 0)$. When $\mathcal{F}$ is an affine mapping of the form $\boldsymbol{b} - \mathcal{A}(\boldsymbol{X})$, this simplifies to

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\mathcal{A}^*(\boldsymbol{y}^{k-1})), \\ \boldsymbol{y}^k = [\boldsymbol{y}^{k-1} + \delta_k(\boldsymbol{b} - \mathcal{A}(\boldsymbol{X}^k))]_+, \end{cases} \tag{3.6}$$

and thus the extension to linear inequality constraints is straightforward.

3.3. Examples. Suppose we have available linear measurements $\boldsymbol{b}$ about a matrix $\boldsymbol{M}$, which take the form

$$\boldsymbol{b} = \mathcal{A}(\boldsymbol{M}) + \boldsymbol{z}, \tag{3.7}$$

where $\boldsymbol{z} \in \mathbb{R}^m$ is a noise vector. Then under these circumstances, one might want to find the matrix which minimizes the nuclear norm among all matrices which are consistent with the data $\boldsymbol{b}$.

3.3.1. Linear inequality constraints. A possible approach to this problem consists in solving

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & |\mathbf{vec}(\mathcal{A}^*(\boldsymbol{r}))| \leq \mathbf{vec}(\boldsymbol{E}), \quad \boldsymbol{r} := \boldsymbol{b} - \mathcal{A}(\boldsymbol{X}), \end{array} \tag{3.8}$$

where $\boldsymbol{E}$ is an array of tolerances, which is adjusted to fit the noise statistics. Above, $\mathbf{vec}(\boldsymbol{A}) \leq \mathbf{vec}(\boldsymbol{B})$, for any two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, means componentwise inequalities; that is, $A_{ij} \leq B_{ij}$ for all indices i, j. We use this notation so as not to confuse the reader with the positive semidefinite ordering. In the case of the matrix completion problem where $\mathcal{A}$ extracts sampled entries indexed by Ω, one can always see the

data vector as the sampled entries of some matrix $\boldsymbol{B}$ obeying $\mathcal{P}_\Omega(\boldsymbol{B}) = \mathcal{A}^*(\boldsymbol{b})$; the constraint is then natural for it may be expressed as

$$|B_{ij} - X_{ij}| \le E_{ij}, \quad (i,j) \in \Omega.$$

If $\boldsymbol{z}$ is white noise with standard deviation σ, one may want to use a multiple of σ for E_{ij}. In words, we are looking for a matrix with minimum nuclear norm under the constraint that all of its sampled entries do not deviate too much from what has been observed.

Let $\boldsymbol{Y}_+ \in \mathbb{R}^{n_1 \times n_2}$ (resp., $\boldsymbol{Y}_- \in \mathbb{R}^{n_1 \times n_2}$) be the Lagrange multiplier associated with the componentwise linear inequality constraints $\mathbf{vec}(\mathcal{A}^*(\boldsymbol{r})) \le \mathbf{vec}(\boldsymbol{E})$ (resp., $-\mathbf{vec}(\mathcal{A}^*(\boldsymbol{r})) \le \mathbf{vec}(\boldsymbol{E})$). Then starting with $\boldsymbol{Y}_\pm^0 = \boldsymbol{0}$, the SVT iteration for this problem is of the form

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\mathcal{A}^*\mathcal{A}(\boldsymbol{Y}_+^{k-1} - \boldsymbol{Y}_-^{k-1})), \\ \boldsymbol{Y}_\pm^k = [\boldsymbol{Y}_\pm^{k-1} + \delta_k(\pm\mathcal{A}^*(\boldsymbol{r}^k) - \boldsymbol{E})]_+, \quad \boldsymbol{r}^k = \boldsymbol{b}^k - \mathcal{A}(\boldsymbol{X}^k), \end{cases} \tag{3.9}$$

where again $[\cdot]_+$ is applied componentwise (in the matrix completion problem, $\mathcal{A}^*\mathcal{A} = \mathcal{P}_\Omega$).

3.3.2. Quadratic constraints. Another natural solution is to solve the quadratically constrained nuclear norm minimization problem

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & \|\boldsymbol{b} - \mathcal{A}(\boldsymbol{X})\| \le \epsilon. \end{array} \tag{3.10}$$

When z is a stochastic error term, ϵ would typically be adjusted to depend on the noise power.

To see how we can adapt our ideas in this setting, we work with the approximate objective functional $\tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ as before, and rewrite our program in the conic form

$$\begin{array}{ll} \text{minimize} & \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2 \\ \text{subject to} & \begin{bmatrix} \boldsymbol{b} - \mathcal{A}(\boldsymbol{X}) \\ \epsilon \end{bmatrix} \in \mathcal{K}, \end{array} \tag{3.11}$$

where $\mathcal{K}$ is the second-order cone $\mathcal{K} = \{(\boldsymbol{x}, t) \in \mathbb{R}^{m+1} \ : \ \|\boldsymbol{x}\| \le t\}$. This model has also been considered in [49]. The cone $\mathcal{K}$ is self-dual. The Lagrangian is then given by

$$\mathcal{L}(\boldsymbol{X}; \boldsymbol{y}, s) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2 + \langle \boldsymbol{y}, \boldsymbol{b} - \mathcal{A}(\boldsymbol{X})\rangle - s\epsilon,$$

where $(\boldsymbol{y}, s) \in \mathbb{R}^{m+1} \in \mathcal{K}^* = \mathcal{K}$; that is, $\|\boldsymbol{y}\| \le s$. Letting $P_\mathcal{K}$ be the orthogonal projection onto $\mathcal{K}$, this leads to the simple iteration

$$\begin{cases} \boldsymbol{X}^k = \mathcal{D}_\tau(\mathcal{A}^*(\boldsymbol{y}^k)), \\ \begin{bmatrix} \boldsymbol{y}^k \\ s^k \end{bmatrix} = P_\mathcal{K}\left(\begin{bmatrix} \boldsymbol{y}^{k-1} \\ s^{k-1} \end{bmatrix} + \delta_k \begin{bmatrix} \boldsymbol{b} - \mathcal{A}(\boldsymbol{X}^k) \\ -\epsilon \end{bmatrix} \right). \end{cases} \tag{3.12}$$

This is an explicit algorithm since the projection is given by (see also [37, Proposition 3.3])

$$P_\mathcal{K} : (x,t) \mapsto \begin{cases} (x,t), & \|x\| \le t, \\ \frac{\|x\|+t}{2\|x\|}(x, \|x\|), & -\|x\| \le t \le \|x\|, \\ (0,0), & t \le -\|x\|. \end{cases}$$

3.3.3. General conic constraints. Clearly, one could apply this methodology with general cone constraints of the form $\mathcal{F}(\boldsymbol{X}) + \boldsymbol{d} \in \mathcal{K}$, where $\mathcal{K}$ is some closed and pointed convex cone. Inspired by the work on the Dantzig selector [18], which was originally developed for estimating sparse parameter vectors from noisy data, another approach is to set a constraint on the spectral norm of $\mathcal{A}^*(\boldsymbol{r})$—recall that $\boldsymbol{r}$ is the residual vector $\boldsymbol{b} - \mathcal{A}(\boldsymbol{X})$—and solve

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{X}\|_* \\ \text{subject to} & \|\mathcal{A}^*(\boldsymbol{r})\| \le \epsilon. \end{array} \tag{3.13}$$

Developing our approach in this setting is straightforward and involves projections of the dual variable onto the positive semidefinite cone.

3.4. When the proximal problem gets close. We now show that minimizing the proximal objective $f_\tau(\boldsymbol{X}) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ is the same as minimizing the nuclear norm in the limit of large τ's. The theorem below is general and covers the special case of linear equality constraints as in (2.8).

THEOREM 3.1. *Let $\boldsymbol{X}_\tau^\star$ be the solution to* (3.4) *and $\boldsymbol{X}_\infty$ be the minimum Frobenius norm solution to* (1.6) *defined as*

$$\boldsymbol{X}_\infty := \arg\min_{\boldsymbol{X}} \{\|\boldsymbol{X}\|_F^2 \ : \ \boldsymbol{X} \text{ is a solution of } (1.6)\}. \tag{3.14}$$

Assume that the $f_i(\boldsymbol{X})$'s, $1 \le i \le m$, are convex and lower semicontinuous. Then

$$\lim_{\tau\to\infty} \|\boldsymbol{X}_\tau^\star - \boldsymbol{X}_\infty\|_F = 0. \tag{3.15}$$

Proof. It follows from the definition of $\boldsymbol{X}_\tau^\star$ and $\boldsymbol{X}_\infty$ that

$$\|\boldsymbol{X}_\tau^\star\|_* + \frac{1}{2\tau}\|\boldsymbol{X}_\tau^\star\|_F^2 \le \|\boldsymbol{X}_\infty\|_* + \frac{1}{2\tau}\|\boldsymbol{X}_\infty\|_F^2, \quad \text{and} \quad \|\boldsymbol{X}_\infty\|_* \le \|\boldsymbol{X}_\tau^\star\|_*. \tag{3.16}$$

Summing these two inequalities gives

$$\|\boldsymbol{X}_\tau^\star\|_F^2 \le \|\boldsymbol{X}_\infty\|_F^2, \tag{3.17}$$

which implies that $\|\boldsymbol{X}_\tau^\star\|_F^2$ is bounded uniformly in τ. Thus, we would prove the theorem if we could establish that any convergent subsequence $\{\boldsymbol{X}_{\tau_k}^\star\}_{k\ge1}$ must converge to $\boldsymbol{X}_\infty$.

Consider an arbitrary converging subsequence $\{\boldsymbol{X}_{\tau_k}^\star\}$ and set $\boldsymbol{X}_c := \lim_{k\to\infty} \boldsymbol{X}_{\tau_k}^\star$. Since for each $1 \le i \le m$, $f_i(\boldsymbol{X}_{\tau_k}^\star) \le 0$ and f_i is lower semicontinuous, $\boldsymbol{X}_c$ obeys $f_i(\boldsymbol{X}_c) \le 0$ for $i = 1, \dots, m$. Furthermore, since $\|\boldsymbol{X}_\tau^\star\|_F^2$ is bounded, (3.16) yields

$$\limsup_{\tau\to\infty} \|\boldsymbol{X}_\tau^\star\|_* \le \|\boldsymbol{X}_\infty\|_*, \quad \|\boldsymbol{X}_\infty\|_* \le \liminf_{\tau\to\infty} \|\boldsymbol{X}_\tau^\star\|_*.$$

An immediate consequence is $\lim_{\tau\to\infty} \|\boldsymbol{X}_\tau^\star\|_* = \|\boldsymbol{X}_\infty\|_*$ and, therefore, $\|\boldsymbol{X}_c\|_* = \|\boldsymbol{X}_\infty\|_*$. This shows that $\boldsymbol{X}_c$ is a solution to (1.1). Now it follows from the definition of $\boldsymbol{X}_\infty$ that $\|\boldsymbol{X}_c\|_F \ge \|\boldsymbol{X}_\infty\|_F$, while we also have $\|\boldsymbol{X}_c\|_F \le \|\boldsymbol{X}_\infty\|_F$ because of (3.17). We conclude that $\|\boldsymbol{X}_c\|_F = \|\boldsymbol{X}_\infty\|_F$ and thus $\boldsymbol{X}_c = \boldsymbol{X}_\infty$ since $\boldsymbol{X}_\infty$ is unique. ∎

4. Convergence analysis. This section establishes the convergence of the SVT iterations. We begin with the simpler proof of the convergence of (2.7) in the special case of the matrix completion problem, and then present the argument for the more general constraints (3.5). We hope that this progression will make the second and more general proof more transparent. We have seen that SVT iterations are projected gradient-descent algorithms applied to the dual problems. The convergence of projected gradient-descent algorithms has been well studied; see [6, 25, 38, 43, 45, 52, 68], for example.

4.1. Convergence for matrix completion. We begin by recording a lemma which establishes the strong convexity of the objective f_τ.

LEMMA 4.1. *Let* $\boldsymbol{Z} \in \partial f_\tau(\boldsymbol{X})$ *and* $\boldsymbol{Z}' \in \partial f_\tau(\boldsymbol{X}')$. *Then*

$$\langle \boldsymbol{Z} - \boldsymbol{Z}', \boldsymbol{X} - \boldsymbol{X}' \rangle \geq \|\boldsymbol{X} - \boldsymbol{X}'\|_F^2. \tag{4.1}$$

The proof may be found in [57, page 240], but we sketch it for convenience. We have $\boldsymbol{Z} \in \partial f_\tau(\boldsymbol{X})$ if and only if $\boldsymbol{Z} = \tau \boldsymbol{Z}_0 + \boldsymbol{X}$, where $\boldsymbol{Z}_0 \in \partial \|\boldsymbol{X}\|_*$. This gives

$$\langle \boldsymbol{Z} - \boldsymbol{Z}', \boldsymbol{X} - \boldsymbol{X}' \rangle = \tau \langle \boldsymbol{Z}_0 - \boldsymbol{Z}_0', \boldsymbol{X} - \boldsymbol{X}' \rangle + \|\boldsymbol{X} - \boldsymbol{X}'\|_F^2,$$

and it suffices to show that $\langle \boldsymbol{Z}_0 - \boldsymbol{Z}_0', \boldsymbol{X} - \boldsymbol{X}' \rangle \geq 0$. From (2.6), we have that any subgradient of the nuclear norm at $\boldsymbol{X}$ obeys $\|\boldsymbol{Z}_0\|_2 \leq 1$ and $\langle \boldsymbol{Z}_0, \boldsymbol{X} \rangle = \|\boldsymbol{X}\|_*$. In particular, this gives $|\langle \boldsymbol{Z}_0, \boldsymbol{X}' \rangle| \leq \|\boldsymbol{Z}_0\|_2 \|\boldsymbol{X}'\|_* \leq \|\boldsymbol{X}'\|_*$ and, likewise, $|\langle \boldsymbol{Z}_0', \boldsymbol{X} \rangle| \leq \|\boldsymbol{X}\|_*$. Then the lemma follows from

$$\begin{aligned} \langle \boldsymbol{Z}_0 - \boldsymbol{Z}_0', \boldsymbol{X} - \boldsymbol{X}' \rangle &= \langle \boldsymbol{Z}_0, \boldsymbol{X} \rangle + \langle \boldsymbol{Z}_0', \boldsymbol{X}' \rangle - \langle \boldsymbol{Z}_0, \boldsymbol{X}' \rangle - \langle \boldsymbol{Z}_0', \boldsymbol{X} \rangle \\ &= \|\boldsymbol{X}\|_* + \|\boldsymbol{X}'\|_* - \langle \boldsymbol{Z}_0, \boldsymbol{X}' \rangle - \langle \boldsymbol{Z}_0', \boldsymbol{X} \rangle \geq 0. \end{aligned}$$

This lemma is key in showing that the SVT algorithm (2.7) converges. Indeed, applying [25, Theorem 2.1] gives the theorem below.

THEOREM 4.2. *Suppose the step sizes obey* $0 < \inf \delta_k \leq \sup \delta_k < 2/\|\mathcal{A}\|^2$. *Then the sequence* $\{\boldsymbol{X}^k\}$ *obtained via* (3.3) *converges to the unique solution to* (3.1). *In particular, the sequence* $\{\boldsymbol{X}^k\}$ *obtained via* (2.7) *converges to the unique solution of* (2.8) *provided that* $0 < \inf \delta_k \leq \sup \delta_k < 2$.

4.2. General convergence theorem. Our second result is more general and establishes the convergence of the SVT iterations to the solution of (3.4) under general convex constraints. From now on, we will assume that the function $\mathcal{F}(\boldsymbol{X})$ is Lipschitz only in the sense that

$$\|\mathcal{F}(\boldsymbol{X}) - \mathcal{F}(\boldsymbol{Y}\| \leq L(\mathcal{F})\|\boldsymbol{X} - \boldsymbol{Y}\|_F, \tag{4.2}$$

for some nonnegative constant $L(\mathcal{F})$. Note that if $\mathcal{F}$ is affine, $\mathcal{F}(\boldsymbol{X}) = \boldsymbol{b} - \mathcal{A}(\boldsymbol{X})$, we have $L(\mathcal{F}) = \|\mathcal{A}\|_2$ where $\|\mathcal{A}\|_2$ is the spectrum norm of the linear transformation $\mathcal{A}$ defined as $\|\mathcal{A}\|_2 := \sup\{\|\mathcal{A}(\boldsymbol{X})\|_{\ell_2} : \|\boldsymbol{X}\|_F = 1\}$. We also recall that $\mathcal{F}(\boldsymbol{X}) = (f_1(\boldsymbol{X}), \ldots, f_m(\boldsymbol{X}))$ where each f_i is convex and that the Lagrangian for the problem (3.4) is given by

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{y}) = f_\tau(\boldsymbol{X}) + \langle \boldsymbol{y}, \mathcal{F}(\boldsymbol{X}) \rangle, \quad \boldsymbol{y} \geq \boldsymbol{0}.$$

To simplify, we will assume that strong duality holds which is automatically true if the constraints obey constraint qualifications such as Slater's condition [7].

We first establish a preparatory lemma, whose proof can be found in [31].

LEMMA 4.3. *Let* $(\boldsymbol{X}^\star, \boldsymbol{y}^\star)$ *be a primal-dual optimal pair for* (3.4). *Then for each* $\delta > 0$, $\boldsymbol{y}^\star$ *obeys*

$$\boldsymbol{y}^\star = [\boldsymbol{y}^\star + \delta \mathcal{F}(\boldsymbol{X}^\star)]_+. \tag{4.3}$$

We are now in the position to state our general convergence result; see also [25, Theorem 2.1].

THEOREM 4.4. *Suppose that the step sizes obey* $0 < \inf \delta_k \le \sup \delta_k < 2/\|L(\mathcal{F})\|^2$, *where* $L(\mathcal{F})$ *is the Lipschitz constant in* (4.2). *Then assuming strong duality, the sequence* $\{\boldsymbol{X}^k\}$ *obtained via* (3.5) *converges to the unique solution of* (3.4).

Proof. Let $(\boldsymbol{X}^\star, \boldsymbol{y}^\star)$ be primal-dual optimal for the problem (3.4). We claim that the optimality conditions give that for all $\boldsymbol{X}$

$$\begin{aligned} \langle \boldsymbol{Z}^k, \boldsymbol{X} - \boldsymbol{X}^k \rangle + \langle \boldsymbol{y}^{k-1}, \mathcal{F}(\boldsymbol{X}) - \mathcal{F}(\boldsymbol{X}^k) \rangle \ge 0, \\ \langle \boldsymbol{Z}^\star, \boldsymbol{X} - \boldsymbol{X}^\star \rangle + \langle \boldsymbol{y}^\star, \mathcal{F}(\boldsymbol{X}) - \mathcal{F}(\boldsymbol{X}^\star) \rangle \ge 0, \end{aligned} \tag{4.4}$$

for some $\boldsymbol{Z}^k \in \partial f_\tau(\boldsymbol{X}^k)$ and some $\boldsymbol{Z}^\star \in \partial f_\tau(\boldsymbol{X}^\star)$. We justify this assertion by proving one of the two inequalities since the other is exactly similar. For the first, $\boldsymbol{X}^k$ minimizes $\mathcal{L}(\boldsymbol{X}, \boldsymbol{y}^{k-1})$ over all $\boldsymbol{X}$ and, therefore, there exist $\boldsymbol{Z}^k \in \partial f_\tau(\boldsymbol{X}^k)$ and $\boldsymbol{Z}_i^k \in \partial f_i(\boldsymbol{X}^k)$, $1 \le i \le m$, such that

$$\boldsymbol{Z}^k + \sum_{i=1}^m y_i^{k-1} \boldsymbol{Z}_i^k = 0.$$

Now because each f_i is convex,

$$f_i(\boldsymbol{X}) - f_i(\boldsymbol{X}^k) \ge \langle \boldsymbol{Z}_i^k, \boldsymbol{X} - \boldsymbol{X}^k \rangle$$

and, therefore,

$$\langle \boldsymbol{Z}^k, \boldsymbol{X} - \boldsymbol{X}^k \rangle + \sum_{i=1}^m y_i^{k-1} (f_i(\boldsymbol{X}) - f_i(\boldsymbol{X}^k)) \ge \left\langle \boldsymbol{Z}^k + \sum_{i=1}^m y_i^{k-1} \boldsymbol{Z}_i^k, \boldsymbol{X} - \boldsymbol{X}^k \right\rangle = 0.$$

This is (4.4).

Now write the first inequality in (4.4) for $\boldsymbol{X}^\star$, write the second for $\boldsymbol{X}^k$, and sum the two inequalities. This gives

$$\langle \boldsymbol{Z}^k - \boldsymbol{Z}^\star, \boldsymbol{X}^k - \boldsymbol{X}^\star \rangle + \langle \boldsymbol{y}^{k-1} - \boldsymbol{y}^\star, \mathcal{F}(\boldsymbol{X}^k) - \mathcal{F}(\boldsymbol{X}^\star) \rangle \le 0.$$

It follows from Lemma 4.1 that

$$\langle \boldsymbol{y}^{k-1} - \boldsymbol{y}^\star, \mathcal{F}(\boldsymbol{X}^k) - \mathcal{F}(\boldsymbol{X}^\star) \rangle \le -\langle \boldsymbol{Z}^k - \boldsymbol{Z}^\star, \boldsymbol{X}^k - \boldsymbol{X}^\star \rangle \le -\|\boldsymbol{X}^k - \boldsymbol{X}^\star\|_F^2. \tag{4.5}$$

We continue and observe that because $\boldsymbol{y}^\star = [\boldsymbol{y}^\star + \delta_k \mathcal{F}(\boldsymbol{X})]_+$ by Lemma 4.3, we have

$$\begin{aligned} \|\boldsymbol{y}^k - \boldsymbol{y}^\star\| &= \|[\boldsymbol{y}^{k-1} + \delta_k \mathcal{F}(\boldsymbol{X}^k)]_+ - [\boldsymbol{y}^\star + \delta_k \mathcal{F}(\boldsymbol{X}^\star)]_+\| \\ &\le \|\boldsymbol{y}^{k-1} - \boldsymbol{y}^\star + \delta_k (\mathcal{F}(\boldsymbol{X}^k) - \mathcal{F}(\boldsymbol{X}^\star))\| \end{aligned}$$

since the projection onto the convex set $\mathbb{R}_+^m$ is a contraction. Therefore,

$$\begin{aligned} \|\boldsymbol{y}^k - \boldsymbol{y}^\star\|^2 &= \|\boldsymbol{y}^{k-1} - \boldsymbol{y}^\star\|^2 + 2\delta_k \langle \boldsymbol{y}^{k-1} - \boldsymbol{y}^\star, \mathcal{F}(\boldsymbol{X}^k) - \mathcal{F}(\boldsymbol{X}^\star) \rangle \\ &\quad + \delta_k^2 \|\mathcal{F}(\boldsymbol{X}^k) - \mathcal{F}(\boldsymbol{X}^\star)\|^2 \\ &\le \|\boldsymbol{y}^{k-1} - \boldsymbol{y}^\star\|^2 - 2\delta_k \|\boldsymbol{X}^k - \boldsymbol{X}^\star\|_F^2 + \delta_k^2 L^2 \|\boldsymbol{X}^k - \boldsymbol{X}^\star\|_F^2, \end{aligned}$$

where we have put L instead of $L(\mathcal{F})$ for short. Under our assumptions about the size of δ_k, we have $2\delta_k - \delta_k^2 L^2 \geq \beta$ for all $k \geq 1$ and some $\beta > 0$. Then

$$\|\boldsymbol{y}^k - \boldsymbol{y}^\star\|^2 \leq \|\boldsymbol{y}^{k-1} - \boldsymbol{y}^\star\|^2 - \beta\|\boldsymbol{X}^k - \boldsymbol{X}^\star\|_F^2, \tag{4.6}$$

and the conclusion is as before. □

5. Implementation and numerical results. This section provides implementation details of the SVT algorithm—as to make it practically effective for matrix completion—such as the numerical evaluation of the singular value thresholding operator, the selection of the step size δ_k, the selection of a stopping criterion, and so on. This section also introduces several numerical simulation results which demonstrate the performance and effectiveness of the SVT algorithm. We show that $30{,}000 \times 30{,}000$ matrices of rank 10 are recovered from just about 0.4% of their sampled entries in a matter of a few minutes on a modest desktop computer with a 1.86 GHz CPU (dual core with Matlab's multithreading option enabled) and 3 GB of memory.

5.1. Implementation details.

5.1.1. Evaluation of the singular value thresholding operator. To apply the singular value thresholding operator at level τ to an input matrix, it suffices to know those singular values and corresponding singular vectors above the threshold τ. In the matrix completion problem, the singular value thresholding operator is applied to sparse matrices $\{\boldsymbol{Y}^k\}$ since the number of sampled entries is typically much lower than the number of entries in the unknown matrix $\boldsymbol{M}$, and we are hence interested in numerical methods for computing the dominant singular values and singular vectors of large sparse matrices. The development of such methods is a relatively mature area in scientific computing and numerical linear algebra in particular. In fact, many high-quality packages are readily available. Our implementation uses PROPACK; see [44] for documentation and availability. One reason for this choice is convenience: PROPACK comes in a Matlab and a Fortran version, and we find it convenient to use the well-documented Matlab version. More importantly, PROPACK uses the iterative Lanczos algorithm to compute the singular values and singular vectors directly, by using the Lanczos bidiagonalization algorithm with partial reorthogonalization. In particular, PROPACK does not compute the eigenvalues and eigenvectors of $(\boldsymbol{Y}^k)^*\boldsymbol{Y}^k$ and $\boldsymbol{Y}^k(\boldsymbol{Y}^k)^*$, or of an augmented matrix as in the Matlab built-in function "`svds`," for example. Consequently, PROPACK is an efficient—in terms of both number of flops and storage requirement—and stable package for computing the dominant singular values and singular vectors of a large sparse matrix. For information, the available documentation [44] reports a speedup factor of about ten over Matlab's "`svds`." Furthermore, the Fortran version of PROPACK is about 3 to 4 times faster than the Matlab version. Despite this significant speedup, we have used only the Matlab version, but since the singular value shrinkage operator is by and large the dominant cost in the SVT algorithm, we expect that a Fortran implementation would run about 3 to 4 times faster.

As for most SVD packages, though one can specify the number of singular values to compute, PROPACK cannot automatically compute only those singular values exceeding the threshold τ. One must instead specify the number s of singular values ahead of time, and the software will compute the s largest singular values and corresponding singular vectors. To use this package, we must then determine the number s_k of singular values of $\boldsymbol{Y}^{k-1}$ to be computed at the kth iteration. We use the following simple method. Let $r_{k-1} = \operatorname{rank}(\boldsymbol{X}^{k-1})$ be the number of nonzero singular

values of $\boldsymbol{X}^{k-1}$ at the previous iteration. Set $s_k = r_{k-1} + 1$ and compute the first s_k singular values of $\boldsymbol{Y}^{k-1}$. If some of the computed singular values are already smaller than τ, then s_k is the right choice. Otherwise, increment s_k by a predefined integer ℓ repeatedly until some of the singular values fall below τ. In the experiments, we choose $\ell = 5$. Another rule might be to repeatedly multiply s_k by a positive number—e.g., 2—until our criterion is met. Incrementing s_k by a fixed integer works very well in practice; in our experiments, we very rarely need more than one update.

We note that it is not necessary to rerun the Lanczos iterations for the first s_k vectors since they have already been computed; only a few new singular values (ℓ of them) need to be numerically evaluated. This can be done by modifying the PROPACK routines. We have not yet modified PROPACK, however. Had we done so, our run times would be decreased.

5.1.2. Step sizes. There is a large literature on ways of selecting a step size, but for simplicity, we shall use step sizes that are independent of the iteration count; that is, $\delta_k = \delta$ for $k = 1, 2, \ldots$. From Theorem 4.2, convergence for the completion problem is guaranteed (2.7) provided that $0 < \delta < 2$. This choice is, however, too conservative and the convergence is typically slow. In our experiments, we use instead

$$\delta = 1.2\,\frac{n_1 n_2}{m}, \tag{5.1}$$

i.e., 1.2 times the undersampling ratio. We give a heuristic justification below.

Consider a fixed matrix $\boldsymbol{A} \in \mathbb{R}^{n_1 \times n_2}$. Under the assumption that the column and row spaces of $\boldsymbol{A}$ are not well aligned with the vectors taken from the canonical basis of $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$, respectively—the *incoherence assumption* in [16]—then with very large probability over the choices of Ω, we have

$$(1-\epsilon)p\,\|\boldsymbol{A}\|_F^2 \le \|\mathcal{P}_\Omega(\boldsymbol{A})\|_F^2 \le (1+\epsilon)p\,\|\boldsymbol{A}\|_F^2, \quad p := m/(n_1 n_2), \tag{5.2}$$

provided that the rank of $\boldsymbol{A}$ is not too large. The probability model is that Ω is a set of sampled entries of cardinality m sampled uniformly at random so that all the choices are equally likely. In (5.2), we want to think of ϵ as a small constant, e.g., smaller than $1/2$. In other words, the "energy" of $\boldsymbol{A}$ on Ω (the set of sampled entries) is just about proportional to the size of Ω. The near isometry (5.2) is a consequence of Theorem 4.1 in [16], and we omit the details.

Now returning to the proof of Theorem 4.2, one sees that a sufficient condition for the convergence of (2.7) is

$$\exists \beta > 0, \quad -2\delta\|\boldsymbol{X}^\star - \boldsymbol{X}^k\|_F^2 + \delta^2\|\mathcal{P}_\Omega(\boldsymbol{X}^\star - \boldsymbol{X}^k)\|_F^2 \le -\beta\|\boldsymbol{X}^\star - \boldsymbol{X}^k\|_F^2,$$

which is equivalent to

$$0 < \delta < 2\frac{\|\boldsymbol{X}^\star - \boldsymbol{X}^k\|_F^2}{\|\mathcal{P}_\Omega(\boldsymbol{X}^\star - \boldsymbol{X}^k)\|_F^2}.$$

Since $\|\mathcal{P}_\Omega(\boldsymbol{X})\|_F \le \|\boldsymbol{X}\|_F$ for any matrix $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$, it is safe to select $\delta < 2$. But suppose that we could apply (5.2) to the matrix $\boldsymbol{A} = \boldsymbol{X}^\star - \boldsymbol{X}^k$. Then we could take δ inversely proportional to p; e.g., with $\epsilon = 1/4$, we could take $\delta \le 1.6p^{-1}$. Below, we shall use the value $\delta = 1.2p^{-1}$ which allows us to take large steps and still provides convergence, at least empirically.

The reason why this is not a rigorous argument is that (5.2) cannot be applied to $\boldsymbol{A} = \boldsymbol{X}^\star - \boldsymbol{X}^k$ even though this matrix difference may obey the incoherence assumption. The issue here is that $\boldsymbol{X}^\star - \boldsymbol{X}^k$ is not a fixed matrix, but rather depends on Ω since the iterates $\{\boldsymbol{X}^k\}$ are computed with the knowledge of the sampled set.

5.1.3. Initial steps. The SVT algorithm starts with $\boldsymbol{Y}^0 = \boldsymbol{0}$, and we want to choose a large τ to make sure that the solution of (2.8) is close enough to a solution of (1.1). Define k_0 as that integer obeying

$$\frac{\tau}{\delta \|\mathcal{P}_\Omega(\boldsymbol{M})\|_2} \in (k_0 - 1, k_0]. \tag{5.3}$$

Since $\boldsymbol{Y}^0 = \boldsymbol{0}$, it is not difficult to see that $\boldsymbol{X}^k = \boldsymbol{0}$ and $\boldsymbol{Y}^k = k\delta\,\mathcal{P}_\Omega(\boldsymbol{M})$ for $k = 1, \ldots, k_0$. To save work, we may simply skip the computations of $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^{k_0}$ and start the iteration by computing $\boldsymbol{X}^{k_0+1}$ from $\boldsymbol{Y}^{k_0}$.

This strategy is a special case of a *kicking device* introduced in [55]; the main idea of such a kicking scheme is that one can "jump over" a few steps whenever possible. Just as in the aforementioned reference, we can develop similar kicking strategies here as well. Because in our numerical experiments the kicking is rarely triggered, we forgo the description of such strategies.

5.1.4. Stopping criteria. Here, we discuss stopping criteria for the sequence of SVT iterations (2.7) and present two possibilities.

The first is motivated by the first-order optimality conditions or KKT conditions tailored to the minimization problem (2.8). By (2.14) and letting $\partial_{\boldsymbol{Y}} g_0(\boldsymbol{Y}) = \boldsymbol{0}$ in (2.13), we see that the solution $\boldsymbol{X}_\tau^\star$ to (2.8) must also verify

$$\begin{cases} \boldsymbol{X} = \mathcal{D}_\tau(\boldsymbol{Y}), \\ \mathcal{P}_\Omega(\boldsymbol{X} - \boldsymbol{M}) = \boldsymbol{0}, \end{cases} \tag{5.4}$$

where $\boldsymbol{Y}$ is a matrix vanishing outside of Ω^c. Therefore, to make sure that $\boldsymbol{X}^k$ is close to $\boldsymbol{X}_\tau^\star$, it is sufficient to check how close $(\boldsymbol{X}^k, \boldsymbol{Y}^{k-1})$ is to obeying (5.4). By definition, the first equation in (5.4) is always true. Therefore, it is natural to stop (2.7) when the error in the second equation is below a specified tolerance. We suggest stopping the algorithm when

$$\frac{\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F}{\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F} \le \epsilon, \tag{5.5}$$

where ϵ is a fixed tolerance, e.g., 10^{-4}. We provide a short heuristic argument justifying this choice below.

In the matrix completion problem, we know that under suitable assumptions

$$\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F^2 \asymp p\,\|\boldsymbol{M}\|_F^2,$$

which is just (5.2) applied to the fixed matrix $\boldsymbol{M}$ (the symbol "$\asymp$" here means that there is a constant ϵ as in (5.2)). Suppose we could also apply (5.2) to the matrix $\boldsymbol{X}^k - \boldsymbol{M}$ (which we rigorously cannot since $\boldsymbol{X}^k$ depends on Ω); then we would have

$$\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F^2 \asymp p\,\|\boldsymbol{X}^k - \boldsymbol{M}\|_F^2, \tag{5.6}$$

and thus

$$\frac{\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F}{\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F} \asymp \frac{\|\boldsymbol{X}^k - \boldsymbol{M}\|_F}{\|\boldsymbol{M}\|_F}.$$

In words, one would control the relative reconstruction error by controlling the relative error on the set of sampled locations.

A second stopping criterion comes from duality theory. First, the iterates $\boldsymbol{X}^k$ are generally not feasible for (2.8) although they become asymptotically feasible. One can construct a feasible point from $\boldsymbol{X}^k$ by projecting it onto the affine space $\{\boldsymbol{X} : \mathcal{P}_\Omega(\boldsymbol{X}) = \mathcal{P}_\Omega(\boldsymbol{M})\}$ as $\tilde{\boldsymbol{X}}^k = \boldsymbol{X}^k + \mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k)$. As usual let $f_\tau(\boldsymbol{X}) = \tau\|\boldsymbol{X}\|_* + \frac{1}{2}\|\boldsymbol{X}\|_F^2$ and denote by $p^\star$ the optimal value of (2.8). Since $\tilde{\boldsymbol{X}}^k$ is feasible, we have $p^\star \le f_\tau(\tilde{\boldsymbol{X}}^k) := b_k$. Second, using the notations of section 2.4, duality theory gives that $a_k := g_0(\boldsymbol{Y}^{k-1}) = \mathcal{L}(\boldsymbol{X}^k, \boldsymbol{Y}^{k-1}) \le p^\star$. Therefore, $b_k - a_k$ is an upper bound on the duality gap, and one can stop the algorithm when this quantity falls below a given tolerance.

For very large problems in which one holds $\boldsymbol{X}^k$ in reduced SVD form, one may not want to compute the projection $\tilde{\boldsymbol{X}}^k$ since this matrix would not have low rank and would require significant storage space (presumably, one would not want to spend much time computing this projection either). Hence, the second method makes practical sense only when the dimensions are not prohibitively large, or when the iterates do not have low rank.

Similarly, one can derive stopping criteria for all the iterations (3.3), (3.5), and (3.6). For example, we can stop (3.3) for general linear constraints when $\|\mathcal{A}(\boldsymbol{X}^k) - \boldsymbol{b}\|/\|\boldsymbol{b}\| \le \epsilon$. We omit the detailed discussions here.

5.1.5. Algorithm. We conclude this section by summarizing the implementation details and give the SVT algorithm for matrix completion below (Algorithm 1). Of course, one would obtain a very similar structure for the more general problems of the form (3.1) and (3.4) with linear inequality constraints. For convenience, define for each nonnegative integer $s \le \min\{n_1, n_2\}$,

$$[\boldsymbol{U}^k, \boldsymbol{\Sigma}^k, \boldsymbol{V}^k]_s, \quad k = 1, 2, \ldots,$$

where $\boldsymbol{U}^k = [\boldsymbol{u}_1^k, \ldots, \boldsymbol{u}_s^k]$ and $\boldsymbol{V}^k = [\boldsymbol{v}_1^k, \ldots, \boldsymbol{v}_s^k]$ are the first s singular vectors of the matrix $\boldsymbol{Y}^k$, and $\boldsymbol{\Sigma}^k$ is a diagonal matrix with the first s singular values $\sigma_1^k, \ldots, \sigma_s^k$ on the diagonal.

5.2. Numerical results.

5.2.1. Linear equality constraints. Our implementation is in Matlab, and all the computational results we are about to report were obtained on a desktop computer with a 1.86 GHz CPU (dual core with Matlab's multithreading option enabled) and 3 GB of memory. In our simulations, we generate $n \times n$ matrices of rank r by sampling two $n \times r$ factors $\boldsymbol{M}_L$ and $\boldsymbol{M}_R$ independently, each having independently and identically distributed Gaussian entries, and setting $\boldsymbol{M} = \boldsymbol{M}_L \boldsymbol{M}_R^*$ as it is suggested in [16]. The set of observed entries Ω is sampled uniformly at random among all sets of cardinality m.

The recovery is performed via the SVT algorithm (Algorithm 1), and we use

$$\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F / \|\mathcal{P}_\Omega \boldsymbol{M}\|_F < 10^{-4} \tag{5.7}$$

as a stopping criterion. As discussed earlier, the step sizes are constant and we set $\delta = 1.2p^{-1}$. Throughout this section, we denote the output of the SVT algorithm by $\boldsymbol{X}^{\mathrm{opt}}$. The parameter τ is chosen empirically and set to $\tau = 5n$. A heuristic argument is as follows. Clearly, we would like the term $\tau\|\boldsymbol{M}\|_*$ to dominate the other, namely, $\frac{1}{2}\|\boldsymbol{M}\|_F^2$. For products of Gaussian matrices as above, standard random matrix theory asserts that the Frobenius norm of $\boldsymbol{M}$ concentrates around $n\sqrt{r}$ and that the nuclear norm concentrates around about nr (this should be clear in the simple case where

ALGORITHM 1. SINGULAR VALUE THRESHOLDING (SVT) ALGORITHM.

Input: sampled set Ω and sampled entries $\mathcal{P}_\Omega(\boldsymbol{M})$, step size δ, tolerance ϵ, parameter τ, increment ℓ, and maximum iteration count $k_{\max}$
Output: $\boldsymbol{X}^{\text{opt}}$
Description: Recover a low-rank matrix $\boldsymbol{M}$ from a subset of sampled entries

Set $\boldsymbol{Y}^0 = k_0\delta\,\mathcal{P}_\Omega(\boldsymbol{M})$ (k_0 is defined in (5.3))
Set $r_0 = 0$
for $k = 1$ **to** $k_{\max}$
  Set $s_k = r_{k-1} + 1$
  repeat
    Compute $[\boldsymbol{U}^{k-1}, \boldsymbol{\Sigma}^{k-1}, \boldsymbol{V}^{k-1}]_{s_k}$
    Set $s_k = s_k + \ell$
  until $\sigma^{k-1}_{s_k-\ell} \le \tau$
  Set $r_k = \max\{j : \sigma^{k-1}_j > \tau\}$
  Set $\boldsymbol{X}^k = \sum_{j=1}^{r_k}(\sigma^{k-1}_j - \tau)\boldsymbol{u}^{k-1}_j\boldsymbol{v}^{k-1}_j$
  if $\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F / \|\mathcal{P}_\Omega\boldsymbol{M}\|_F \le \epsilon$ **then break**
  Set $Y^k_{ij} = \begin{cases} 0 & \text{if } (i,j) \notin \Omega, \\ Y^{k-1}_{ij} + \delta(M_{ij} - X^k_{ij}) & \text{if } (i,j) \in \Omega \end{cases}$
end *for* k
Set $\boldsymbol{X}^{\text{opt}} = \boldsymbol{X}^k$

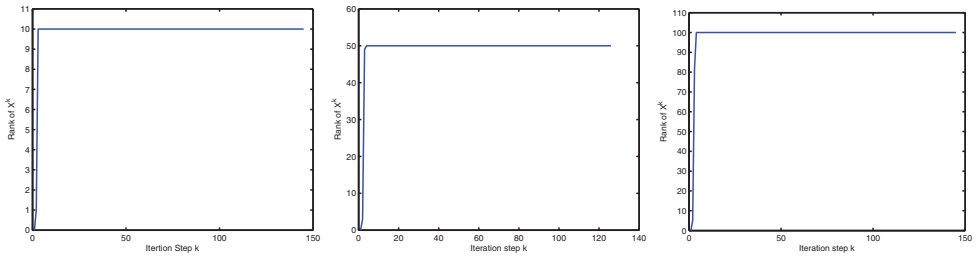
TABLE 5.1

Experimental results for matrix completion. The rank r is the rank of the unknown matrix $\boldsymbol{M}$, m/d_r is the ratio between the number of sampled entries and the number of degrees of freedom in an $n \times n$ matrix of rank r (oversampling ratio), and m/n^2 is the fraction of observed entries. All the computational results on the right are averaged over five runs.

Unknown $\boldsymbol{M}$				Computational results		
Size ($n \times n$)	Rank (r)	m/d_r	m/n^2	Time(s)	# Iters	Relative error
1,000 × 1,000	10	6	0.12	23	117	1.64×10^{-4}
	50	4	0.39	196	114	1.59×10^{-4}
	100	3	0.57	501	129	1.68×10^{-4}
5,000 × 5,000	10	6	0.024	147	123	1.73×10^{-4}
	50	5	0.10	950	108	1.61×10^{-4}
	100	4	0.158	3,339	123	1.72×10^{-4}
10,000 × 10,000	10	6	0.012	281	123	1.73×10^{-4}
	50	5	0.050	2,096	110	1.65×10^{-4}
	100	4	0.080	7,059	127	1.79×10^{-4}
20,000 × 20,000	10	6	0.006	588	124	1.73×10^{-4}
	50	5	0.025	4,581	111	1.66×10^{-4}
30,000 × 30,000	10	6	0.004	1,030	125	1.73×10^{-4}

$r = 1$ and is generally valid). The value $\tau = 5n$ makes sure that on the average, the value of $\tau\|\boldsymbol{M}\|_*$ is about 10 times that of $\frac{1}{2}\|\boldsymbol{M}\|_F^2$ as long as the rank is bounded away from the dimension n.

Our computational results are displayed in Table 5.1. There, we report the run time in seconds, the number of iterations it takes to reach convergence (5.7), and the



(a) $r = 10$ (b) $r = 50$ (c) $r = 100$

FIG. 5.1. *Rank of* $\boldsymbol{X}^k$ *as a function* k *when the unknown matrix* $\boldsymbol{M}$ *is of size* $5{,}000 \times 5{,}000$ *and of rank* r.

relative error of the reconstruction

$$\text{relative error} = \|\boldsymbol{X}^{\text{opt}} - \boldsymbol{M}\|_F / \|\boldsymbol{M}\|_F, \tag{5.8}$$

where $\boldsymbol{M}$ is the real unknown matrix. All of these quantities are averaged over five runs. Table 5.1 also gives the percentage of entries that are observed, namely, m/n^2 together with a quantity that we may want to think of the information oversampling ratio. Recall that an $n \times n$ matrix of rank r depends upon $d_r := r(2n - r)$ degrees of freedom. Then m/d_r is the ratio between the number of sampled entries and the "true dimensionality" of an $n \times n$ matrix of rank r.

The first observation is that the SVT algorithm performs extremely well in these experiments. In all of our experiments, it takes fewer than 200 SVT iterations to reach convergence. As a consequence, the run times are short. As indicated in Table 5.1, we note that one recovers a $1{,}000 \times 1{,}000$ matrix of rank 10 in less than a minute. The algorithm also recovers $30{,}000 \times 30{,}000$ matrices of rank 10 from about 0.4% of their sampled entries in just about 17 minutes. In addition, higher-rank matrices are also efficiently completed: For example, it takes between one and two hours to recover $10{,}000 \times 10{,}000$ matrices of rank 100 and $20{,}000 \times 20{,}000$ matrices of rank 50. We would like to stress that these numbers were obtained on a modest CPU (1.86GHz). Furthermore, a Fortran implementation is likely to cut down on these numbers by a multiplicative factor typically between three and four.

We also check the validity of the stopping criterion (5.7) by inspecting the relative error defined in (5.8). Table 5.1 shows that the heuristic and nonrigorous analysis of section 5.1 holds in practice since the relative reconstruction error is of the same order as $\|\mathcal{P}_\Omega(\boldsymbol{X}^{\text{opt}} - \boldsymbol{M})\|_F / \|\mathcal{P}_\Omega \boldsymbol{M}\|_F \sim 10^{-4}$. Indeed, the overall relative errors reported in Table 5.1 are all less than 2×10^{-4}.

We emphasized all along an important feature of the SVT algorithm, which is that the matrices $\boldsymbol{X}^k$ have low rank. We demonstrate this fact empirically in Figure 5.1, which plots the rank of $\boldsymbol{X}^k$ versus the iteration count k, and does this for unknown matrices of size $5{,}000 \times 5{,}000$ with different ranks. The plots reveal an interesting phenomenon: In our experiments, the rank of $\boldsymbol{X}^k$ is nondecreasing so that the maximum rank is reached in the final steps of the algorithm. In fact, the rank of the iterates quickly reaches the value r of the true rank. After these few initial steps, the SVT iterations search for that matrix with rank r minimizing the objective functional. As mentioned earlier, the low-rank property is crucial for making the algorithm run fast.

We now present a limited study examining the role of the parameters δ and τ in the convergence. We consider a square $1{,}000 \times 1{,}000$ matrix of rank 10, and select

TABLE 5.2

Mean and standard deviation over five runs of the number of iterations needed to achieve (5.7) *for different values of the parameters* δ *and* τ*, together with the average ranks of* $\boldsymbol{X}^k$*. The test example is a random* $1{,}000 \times 1{,}000$ *matrix of rank* 10*, and the number of sampled entries is* $m = 6d_r$*. We also report "DNC" when none of the five runs obeys* (5.7) *after* 1,000 *iterations.*

	$\delta = 0.8p^{-1}$			$\delta = 1.2p^{-1}$			$\delta = 1.6p^{-1}$		
	# of iters		Rank	# of iters		Rank	# of iters		Rank
	Mean	Std	Mean	Mean	Std	Mean	Mean	Std	Mean
$\tau = 2n$	322	192	15.4	764	1246	11.9	DNC	DNC	DNC
$\tau = 3n$	117	2.6	10.0	77	1.8	10.0	1310	2194	10.0
$\tau = 4n$	146	3.1	10.0	97	2.0	9.9	266	435	10.0
$\tau = 5n$	177	4.1	10.0	117	2.8	10.0	87	2.3	10.0
$\tau = 6n$	207	6.2	10.0	136	2.7	10.0	102	1.9	10.0

a number m of entries equal to 6 times the number of degrees of freedom; that is, $m = 6d_r$. Numerical results are reported in Table 5.2, which gives the number of iterations needed to achieve convergence (5.7) and the average rank of each iteration for different values of δ and τ. Table 5.2 suggests that for each value of δ, there exists an optimal τ for which the SVT algorithm performs best. In more details, when τ is smaller than this optimal value, the number of iterations needed to achieve convergence is larger (and also more variable). In addition, the average rank of each iteration is also larger, and thus the computational cost is higher. When τ is close to the optimal value, the SVT algorithm exhibits a rapid convergence, and there is little variability in the number of iterations needed to achieve convergence. When τ is too large, the SVT algorithm may overshrink $\boldsymbol{Y}^k$ at each iterate which, in turn, leads to slow convergence. Table 5.2 also indicates that the convergence of the SVT algorithm depends on the step size δ.

Finally, we demonstrate the results of the SVT algorithm for matrix completion from noisy sampled entries. Suppose we observe data from the model

$$B_{ij} = M_{ij} + Z_{ij}, \qquad (i,j) \in \Omega, \tag{5.9}$$

where $\boldsymbol{Z}$ is a zero-mean Gaussian white noise with standard deviation σ. We run the SVT algorithm but stop early, as soon as $\boldsymbol{X}^k$ is consistent with the data and obeys

$$\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{B})\|_F^2 \le (1+\epsilon)\, m\sigma^2, \tag{5.10}$$

where ϵ is a small parameter. Since $\|\mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{B})\|_F^2$ is very close to $m\sigma^2$ for large values of m, we set $\epsilon = 0$. Our reconstruction $\hat{\boldsymbol{M}}$ is the first $\boldsymbol{X}^k$ obeying (5.10). The results are shown in Table 5.3 (the quantities are averages of 5 runs). Define the noise ratio as $\|\mathcal{P}_\Omega(\boldsymbol{Z})\|_F / \|\mathcal{P}_\Omega(\boldsymbol{M})\|_F$, and the relative error by (5.8). From Table 5.3, we see that the SVT algorithm works well as the relative error between the recovered and the true data matrix is just about equal to the noise ratio.

The theory of low-rank matrix recovery from noisy data is nonexistent at the moment and is obviously beyond the scope of this paper. Having said this, we would like to conclude this section with an intuitive and nonrigorous discussion, which may explain why the observed recovery error is within the noise level. Suppose again that $\hat{\boldsymbol{M}}$ obeys (5.6), namely,

$$\|\mathcal{P}_\Omega(\hat{\boldsymbol{M}} - \boldsymbol{M})\|_F^2 \asymp p\|\hat{\boldsymbol{M}} - \boldsymbol{M}\|_F^2. \tag{5.11}$$

As mentioned earlier, one condition for this to happen is that $\boldsymbol{M}$ and $\hat{\boldsymbol{M}}$ have low rank. This is the reason why it is important to stop the algorithm early as we hope to

TABLE 5.3

Simulation results for noisy data. The computational results are averaged over five runs. For each test, the table shows the results of Algorithm 1 *applied with an early stopping criterion.*

Noise ratio	Unknown matrix $\boldsymbol{M}$				Computational results		
	Size ($n \times n$)	Rank (r)	m/d_r	m/n^2	Time(s)	# Iters	Relative error
10^{-2}	1,000 × 1,000	10	6	0.12	10.8	51	0.78×10^{-2}
		50	4	0.39	87.7	48	0.95×10^{-2}
		100	3	0.57	216	50	1.13×10^{-2}
10^{-1}	1,000 × 1,000	10	6	0.12	4.0	19	0.72×10^{-1}
		50	4	0.39	33.2	17	0.89×10^{-1}
		100	3	0.57	85.2	17	1.01×10^{-1}
1	1,000 × 1,000	10	6	0.12	0.9	3	0.52
		50	4	0.39	7.8	3	0.63
		100	3	0.57	34.8	3	0.69

obtain a solution which is both consistent with the data and has low rank (the limit of the SVT iterations, $\lim_{k\to\infty} \boldsymbol{X}^k$, will not generally have low rank since there may be no low-rank matrix matching the noisy data). From

$$\|\mathcal{P}_\Omega(\hat{\boldsymbol{M}} - \boldsymbol{M})\|_F \le \|\mathcal{P}_\Omega(\hat{\boldsymbol{M}} - \boldsymbol{B})\|_F + \|\mathcal{P}_\Omega(\boldsymbol{B} - \boldsymbol{M})\|_F,$$

and the fact that both terms on the right-hand side are on the order of $\sqrt{m\sigma^2}$, we would have $p\|\hat{\boldsymbol{M}} - \boldsymbol{M}\|_F^2 = O(m\sigma^2)$ by (5.11). In particular, this would give that the relative reconstruction error is on the order of the noise ratio since $\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F^2 \asymp p\|\boldsymbol{M}\|_F^2$—as observed experimentally.

5.2.2. Inequality constraints. We now examine the speed at which one can solve similar problems with inequality constraints instead of linear equality constraints. We assume the model (5.9), where the matrix $\boldsymbol{M}$ of rank r is sampled as before.

We use the noise-aware variant with quadratic constraints (3.10)–(3.11). We set ϵ to $\epsilon^2 = \sigma^2(m + 2\sqrt{2m})$ as this provides a likely upper bound on $\|\boldsymbol{z}\|$ so that the true matrix $\boldsymbol{M}$ is in the feasible set with high probability. The step size is as before and set to $1.2/p$. As a stopping criterion, we stop the iterations (3.12) when the quadratic constraint is very nearly satisfied; in details, we terminate the algorithm when

$$\|\boldsymbol{b} - \mathcal{A}(\boldsymbol{X}^k)\|_F \le (1 + \text{tol})\,\epsilon$$

where tol is some small scalar, typically 0.05, so that the constraint is nearly enforced.

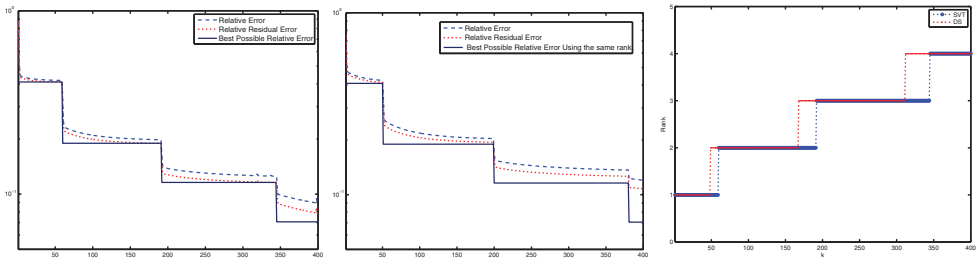
The experimental results are shown in Table 5.4. Our experiments suggest that the algorithm (3.12) is fast and provides statistically accurate answers since it predicts the unseen entries with an accuracy which is about equal to the standard deviation of the noise. In fact, very recent work [15] performed after the original submission of this paper suggests that even with considerable side information about the unknown matrix, one would not be able to do much better.

As seen in the Table 5.4, although the reconstruction is accurate, the ranks of the iterates $\boldsymbol{X}^k$ seem to increase with the iteration count k. This is unlike the case with equality constraints, and we have witnessed this phenomenon in other settings as well such as in the case of linear inequality constraints, e.g., with the iteration (3.9) for solving (3.8). Because a higher rank slows down each iteration, it would be of interest to find methods which stabilize the rank and keep it low in general settings. We leave this important issue for future research.

TABLE 5.4

Simulation results for the noise-aware variant (3.12)*, which solves* (3.11)*. The unknown matrix* $\boldsymbol{M}$ *is* $1{,}000 \times 1{,}000$ *and of rank* $r = 10$*. We get to see* 6 *entries per degree of freedom; i.e.,* $m = 6d_r$*. The noise ratio added is* 0.1*. The averaged true nuclear norm is* 9961*. We choose* $\tau = 5n$ *and* $\delta = 1.2p^{-1}$*. The computational results are averaged over five runs. The computer here is a quad-core* 2.30*GHz AMD Phenom running Matlab* 7.6.0 *with* 3 *threads.*

tol	Time(s)	# Iters	$\|\hat{\boldsymbol{M}} - \boldsymbol{M}\|_F/(n\sigma)$	$\|\hat{\boldsymbol{M}}\|_*$	Rank($\hat{\boldsymbol{M}}$)
0.25	32.8	126	1.11	9034	10
0.2	45.1	158	1.06	9119	15
0.15	94.2	192	1.04	9212	26
0.1	248	232	1.04	9308	39
0.05	447	257	1.03	9415	45



(a) SVT (b) Noise aware variant (3.6) (c) Rank vs. iteration count

FIG. 5.2. *Computational results for the city-to-city distance data set.* (a) *Plot of the reconstruction errors from the SVT algorithm. The blue dashed line is the relative error* $\|\boldsymbol{X}^k - \boldsymbol{M}\|_F/\|\boldsymbol{M}\|_F$*, the red dotted line is the relative residual error* $\|\mathcal{P}_\Omega(\boldsymbol{X}^k - \boldsymbol{M})\|_F/\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F$*, and the black line is the best possible relative error achieved by truncating the SVD of* $\boldsymbol{M}$ *and keeping a number of terms equal to the rank of* $\boldsymbol{X}^k$*.* (b) *Same as* (a) *but with the iteration* (3.6)*.* (c) *Rank of the successive iterates* $\boldsymbol{X}^k$*; the SVT algorithm is in blue and the noise aware variant* (3.6) *is in red.*

5.3. An example with real data. We conclude the numerical section by applying our algorithms to a real data set. We downloaded from the Web site [8] a matrix of geodesic distances (in miles) between 312 cities located in the United States and Canada. The geodesic distances were computed from latitude and longitude information, and rounded to the nearest integer. It is well known that the squared Euclidean distance matrix is a low-rank matrix. With geodesic distances, however, a numerical test suggests that the geodesic-distance matrix $\boldsymbol{M}$ can be well approximated by a low-rank matrix. Indeed, letting $\boldsymbol{M}_3$ be the best rank-3 approximation, we have $\|\boldsymbol{M}_3\|_F/\|\boldsymbol{M}\|_F = 0.9933$ or, equivalently, $\|\boldsymbol{M}_3 - \boldsymbol{M}\|_F/\|\boldsymbol{M}\|_F = 0.1159$. Now sample 30% of the entries of $\boldsymbol{M}$ and obtain and estimate $\hat{\boldsymbol{M}}$ by the SVT algorithm and its noise aware variant (3.6). Here, we set $\tau = 10^7$ which happens to be about 100 times the largest singular value of $\boldsymbol{M}$, and set $\delta = 2$. For completion, we use the SVT algorithm and the iteration (3.6), which solves (3.8) with $E_{ij} = 0.01|M|_{ij}$. In Figure 5.2, we plot the relative error $\|\boldsymbol{M} - \boldsymbol{X}^k\|_F/\|\boldsymbol{M}\|_F$, the relative residual error $\|\mathcal{P}_\Omega(\boldsymbol{M} - \boldsymbol{X}^k)\|_F/\|\mathcal{P}_\Omega(\boldsymbol{M})\|_F$, and the error of the best approximation with the same rank. Let k_i be the smallest integer such that the rank of $\boldsymbol{X}^{k_i}$ is i and the rank of $\boldsymbol{X}^{k_i+1}$ is $i+1$. The computational times needed to reach the k_ith iteration are shown in Table 5.5. Table 5.5 indicates that in a few seconds and in a few iterations, both the SVT algorithm and the iteration (3.6) give a completion, which is nearly as accurate as the best possible low-rank approximation to the unknown matrix $\boldsymbol{M}$.

TABLE 5.5
Speed and accuracy of the completion of the city-to-city distance matrix. Here, $\|\boldsymbol{M} - \boldsymbol{M}_i\|_F/\|\boldsymbol{M}\|_F$ is the best possible relative error achieved by a matrix of rank i.

Algorithm	Rank	k_i	Time	$\|\boldsymbol{M} - \boldsymbol{M}_i\|_F/\|\boldsymbol{M}\|_F$	$\|\boldsymbol{M} - \boldsymbol{X}^{k_i}\|_F/\|\boldsymbol{M}\|_F$
SVT	1	58	1.4	0.4091	0.4170
	2	190	4.8	0.1895	0.1980
	3	343	8.9	0.1159	0.1252
(3.6)	1	47	2.6	0.4091	0.4234
	2	166	7.2	0.1895	0.1998
	3	310	13.3	0.1159	0.1270

6. Discussion. This paper introduced a novel algorithm, namely, the singular value thresholding algorithm for matrix completion and related nuclear norm minimization problems. This algorithm is easy to implement and surprisingly effective both in terms of computational cost and storage requirement when the minimum nuclear norm solution is also the lowest-rank solution. We would like to close this paper by discussing a few open problems and research directions related to this work.

Our algorithm exploits the fact that the sequence of iterates $\{\boldsymbol{X}^k\}$ has low rank when the minimum nuclear solution has low rank. An interesting question is whether one can prove (or disprove) that in a majority of the cases, this is indeed the case.

It would be interesting to explore other ways of computing $\mathcal{D}_\tau(\boldsymbol{Y})$—in words, the action of the singular value shrinkage operator. Our approach uses the Lanczos bidiagonalization algorithm with partial reorthogonalization which takes advantage of sparse inputs, but other approaches are possible. We mention two of them.

1. A series of papers have proposed the use of randomized procedures for the approximation of a matrix $\boldsymbol{Y}$ with a matrix $\boldsymbol{Z}$ of rank r [47, 65]. When this approximation consists of the truncated SVD retaining the part of the expansion corresponding to singular values greater than τ, this can be used to evaluate $\mathcal{D}_\tau(\boldsymbol{Y})$. Some of these algorithms are efficient when the input $\boldsymbol{Y}$ is sparse [65], and it would be interesting to know whether these methods are fast and accurate enough to be used in the SVT iteration (2.7).
2. A wide range of iterative methods for computing matrix functions of the general form $f(\boldsymbol{Y})$ are available today; see [41] for a survey. A valuable research direction is to investigate whether some of these iterative methods, or others to be developed, would provide powerful ways for computing $\mathcal{D}_\tau(\boldsymbol{Y})$.

In practice, one would like to solve (2.8) for large values of τ. However, a larger value of τ generally means a slower rate of convergence. A good strategy might be to start with a value of τ, which is large enough so that (2.8) admits a low-rank solution, and at the same time for which the algorithm converges rapidly. One could then use a continuation method as in [66] to increase the value of τ sequentially according to a schedule $\tau_0, \tau_1, \ldots$, and use the solution to the previous problem with $\tau = \tau_{i-1}$ as an initial guess for the solution to the current problem with $\tau = \tau_i$ (warm starting). We hope to report on this in a separate paper.

Acknowledgments. The second author would like to thank Benjamin Recht and Joel Tropp for fruitful conversations related to this project, and Stephen Becker for his help in preparing the computational results of section 5.2.2.

REFERENCES

[1] ACM SIGKDD and Netflix, *Proceedings of kdd cup and workshop*, http://www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html.

[2] J. Abernethy, F. Bach, T. Evgeniou, and J. P. Vert, *Low-rank matrix factorization with attributes*, Arxiv preprint cs/0611124, 2006.

[3] 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, ACM, Providence, RI, 2007, pp. 17–24.

[4] A. Argyriou, T. Evgeniou, and M. Pontil, *Multi-task feature learning*, Adv. Neural Inform. Process. Syst., 19 (2007), pp. 41–48.

[5] J. Bect, L. Blanc-Feraud, G. Aubert, and A. Chambolle, *A-unified variational framework for image restoration*, in Proc. ECCV, Vol. 3024, Springer, New York, 2004, pp. 1–13.

[6] D. Bertsekas, *On the Goldstein-Levitin-Polyak gradient projection method*, IEEE Trans. Automat. Control, 21 (1976), pp. 174–184.

[7] S. P. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, London, 2004.

[8] J. Burkardt, *Cities—City distance datasets*, http://people.sc.fsu.edu/~burkardt/datasets/cities/cities.html.

[9] J.-F. Cai, R. H. Chan, L. Shen, and Z. Shen, *Restoration of chopped and nodded images by framelets*, SIAM J. Sci. Comput., 30 (2008), pp. 1205–1227.

[10] J.-F. Cai, R. H. Chan, and Z. Shen, *A framelet-based image inpainting algorithm*, Appl. Comput. Harmon. Anal., 24 (2008), pp. 131–149.

[11] J.-F. Cai, S. Osher, and Z. Shen, *Convergence of the linearized Bregman iteration for ℓ_1-norm minimization*, Math. Comp., 78 (2009), pp. 2127–2136.

[12] J.-F. Cai, S. Osher, and Z. Shen, *Linearized Bregman iterations for compressed sensing*, Math. Comp., 78 (2009), pp. 1515–1536.

[13] J.-F. Cai, S. Osher, and Z. Shen, *Linearized Bregman iterations for frame-based image deblurring*, SIAM J. Imaging Sci., 2 (2009), pp. 226–252.

[14] E. J. Candès and F. Guo, *New multiscale transforms, minimum total variation synthesis: Applications to edge-preserving image reconstruction*, Signal Process., 82 (2002), pp. 1519–1543.

[15] E. J. Candès and Y. Plan, *Matrix completion with noise*, Proc. IEEE, to appear.

[16] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2009), pp. 717–772.

[17] E. Candès and J. Romberg, *Sparsity and incoherence in compressive sampling*, Inverse Problems, 23 (2007), pp. 969–985.

[18] E. Candès and T. Tao, *The Dantzig selector: Statistical estimation when p is much larger than n*, Ann. Statist., 35 (2007), pp. 2313–2351.

[19] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[20] E. J. Candès and T. Tao, *Decoding by linear programming*, IEEE Trans. Inform. Theory, 51 (2005), pp. 4203–4215.

[21] E. J. Candès and T. Tao, *Near-optimal signal recovery from random projections: Universal encoding strategies?*, IEEE Trans. Inform. Theory, 52 (2006), pp. 5406–5425.

[22] A. Chai and Z. Shen, *Deconvolution: A wavelet frame approach*, Numer. Math., 106 (2007), pp. 529–587.

[23] R. H. Chan, T. F. Chan, L. Shen, and Z. Shen, *Wavelet algorithms for high-resolution image reconstruction*, SIAM J. Sci. Comput., 24 (2003), pp. 1408–1432.

[24] P. Chen and D. Suter, *Recovering the missing components in a large noisy low-rank matrix: Application to SFM*, IEEE Trans. Pattern Anal. Machine Intelligence, 26 (2004), pp. 1051–1063.

[25] Y. C. Cheng, *On the gradient-projection method for solving the nonsymmetric linear complementarity problem*, J. Optim. Theory Appl., 43 (1984), pp. 527–541.

[26] P. L. Combettes and V. R. Wajs, *Signal recovery by proximal forward-backward splitting*, Multiscale Model. Simul., 4 (2005), pp. 1168–1200.

[27] J. Darbon and S. Osher, *Fast discrete optimization for sparse approximations and deconvolutions*, Department of Mathematics, UCLA, preprint, 2007.

[28] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Comm. Pure Appl. Math., 57 (2004), pp. 1413–1457.

[29] I. Daubechies, G. Teschke, and L. Vese, *Iteratively solving linear inverse problems under general convex constraints*, Inverse Probl. Imaging, 1 (2007), pp. 29–46.
[30] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289–1306.
[31] B.C. Eaves, *On the basic theorem of complementarity*, Math. Program., 1 (1971), pp. 68–75.
[32] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho, *Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)*, Appl. Comput. Harmon. Anal., 19 (2005), pp. 340–358.
[33] M. J. Fadili, J.-L. Starck, and F. Murtagh, *Inpainting and zooming using sparse representations*, Comput. J., 52 (2009), pp. 64–79.
[34] M. Fazel, *Matrix rank minimization with applications*, Ph.D. thesis, Stanford University, Stanford, CA, 2002.
[35] M. Fazel, H. Hindi, and S. P. Boyd, *Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices*, Proc. Amer. Control Conf., 3 (2003), pp. 2156–2162.
[36] M. A. T. Figueiredo and R. D. Nowak, *An EM algorithm for wavelet-based image restoration*, IEEE Trans. Image Process., 12 (2003), pp. 906–916.
[37] M. Fukushima, Z.-Q. Luo, and P. Tseng, *Smoothing functions for second-order-cone complementarity problems*, SIAM J. Optim., 12 (2001/02), pp. 436–460.
[38] A. A. Goldstein, *Convex programming in Hilbert space*, Bull. Amer. Math. Soc., 70 (1964), pp. 709–710.
[39] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.
[40] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for l_1-minimization: Methodology and convergence*, SIAM J. Optim., 19 (2008), pp. 1107–1130.
[41] N. J. Higham, *Functions of matrices, Theory and computation*, SIAM, Philadelphia, 2008.
[42] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms.* I, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 305, Springer-Verlag, Berlin, 1993.
[43] A. N. Iusem, *On the convergence properties of the projected gradient method for convex optimization*, Comput. Appl. Math., 22 (2003), pp. 37–52.
[44] R. M. Larsen, *PROPACK—Software for large and sparse SVD calculations*, http://sun.stanford.edu/~rmunk/PROPACK/.
[45] E. S. Levitin and B. T. Poliak, *Constrained minimization methods (Extremum problems from functional-analytic point of view, discussing methods of solving and convergence conditions)*, USSR Comput. Math. Math. Phys., 6 (1966), pp. 1–50.
[46] A. S. Lewis, *The mathematics of eigenvalue optimization*, Math. Program., 97 (2003), pp. 155–176.
[47] E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Nat. Acad. Sci. USA, 104 (2007), pp. 20167–20172.
[48] S. Lintner and F. Malgouyres, *Solving a variational image restoration model which involves L8 constraints*, Inverse Prob., 20 (2004), pp. 815–831.
[49] Y.-J. Liu, D. F. Sun, and K. C. Toh, *An Implementable Proximal Point Algorithmic Framework for Nuclear Norm Minimization*, Department of Mathematics, National University of Singapore, preprint, 2009, http://www.optimization-online.org/DB_HTML/2009/07/2340.html.
[50] Z. Liu and L. Vandenberghe, *Interior-point method for nuclear norm approximation with application to system identification*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1235–1256.
[51] S. Ma, D. Goldfarb, and L. Chen, *Fixed point and Bregman iterative methods for matrix rank minimization*, Math. Program., to appear.
[52] P. Marcotte and J. H. Wu, *On the convergence of projection methods: Application to the decomposition of affine variational inequalities*, J. Optim. Theory Appl., 85 (1995), pp. 347–362.
[53] M. Mesbahi and G. P. Papavassilopoulos, *On the rank minimization problem over a positive semidefinitelinear matrix inequality*, IEEE Trans. Automat. Control, 42 (1997), pp. 239–243.
[54] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, *An iterative regularization method for total variation-based image restoration*, Multiscale Model. Simul., 4 (2005), pp. 460–489.
[55] S. Osher, Y. Mao, B. Dong, and W. Yin, *Fast linearized Bregman iteration for compressive sensing and sparse denoising*, Commun. Math. Sci., 8 (2010), pp. 93–111.

[56] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., to appear.
[57] R. T. Rockafellar, *Convex analysis*, Princeton Mathematical Series 28, Princeton University Press, Princeton, NJ, 1970.
[58] J. L. Starck, D. L. Donoho, and E. J. Candès, *Astronomical image representation by the curvelet transform*, Astronom. Astrophys., 398 (2003), pp. 785–800.
[59] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3—A Matlab software package for semidefinite-quadratic-linear programming, version* 3.0, 2001, http://www.math.nus.edu.sg/mattohkc/sdpt3.html.
[60] K.-C. Toh and S. Yun, *An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems*, Pacific J. Optim., to appear.
[61] C. Tomasi and T. Kanade, *Shape and motion from image streams under orthography: A factorization method*, Int. J. Comput. Vision, 9 (1992), pp. 137–154.
[62] P. Tseng, *Applications of a splitting algorithm to decomposition in convex programming and variational inequalities.*, SIAM J. Control Optim., 29 (1991), pp. 119–138.
[63] P. Tseng, *A modified forward-backward splitting method for maximal monotone mappings*, SIAM J. Control Optim., 38 (2000), pp. 431–446.
[64] G.A. Watson, *Characterization of the subdifferential of some matrix norms*, Linear Algebra Appl., 170 (1992), pp. 33–45.
[65] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comput. Harmonic Anal., 25 (2008), pp. 335–366.
[66] S. J. Wright, R. Nowak, and M. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process., 57 (2009), pp. 2479–2493.
[67] W. Yin, S. Osher, D. Goldfarb, and J. Darbon, *Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing*, SIAM J. Imaging Sci., 1 (2008), pp. 143–168.
[68] D. L. Zhu and P. Marcotte, *Co-coercivity and its role in the convergence of iterative schemes for solving variational inequalities*, SIAM J. Optim., 6 (1996), pp. 714–726.