

Shiqian Ma · Donald Goldfarb · Lifeng Chen

Fixed point and Bregman iterative methods for matrix rank minimization

October 27, 2008

Abstract The linearly constrained matrix rank minimization problem is widely applicable in many fields such as control, signal processing and system identification. The linearly constrained nuclear norm minimization is a convex relaxation of this problem. Although it can be cast as a semidefinite programming problem, the nuclear norm minimization problem is expensive to solve when the matrices are large. In this paper, we propose fixed point and Bregman iterative algorithms for solving the nuclear norm minimization problem and prove convergence of the first of these algorithms. By using a homotopy approach together with an approximate singular value decomposition procedure, we get a very fast, robust and powerful algorithm that can solve very large matrix rank minimization problems. Our numerical results on randomly generated and real matrix completion problems demonstrate that this algorithm is much faster and provides much better recoverability than semidefinite programming solvers such as SDPT3.

Keywords Matrix Rank Minimization · Matrix Completion Problem · Nuclear Norm Minimization · Fixed Point Iterative Method · Bregman Distances · Singular Value Decomposition

AMS subject classification. 65K05, 90C25, 90C06, 93C41, 68Q32

1 Introduction

In this paper, we are interested in methods for solving the affine constrained matrix rank minimization problem

$$\begin{aligned} \min \operatorname{rank}(X) \\ \text{s.t. } \mathcal{A}(X) = b, \end{aligned} \tag{1.1}$$

where $X \in \mathbb{R}^{m \times n}$ is the decision variable, and the linear map $\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^p$ and vector $b \in \mathbb{R}^p$ are given. This model has many applications such as determining a low-order controller for a plant [19] and a minimum order linear system realization [17], and solving low-dimensional Euclidean embedding problems [25].

Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027. Email: {sm2756, goldfarb, lc2161}@columbia.edu
Research supported in part by NSF Grant DMS 06-06712, ONR Grants N00014-03-0514 and N00014-08-1-1118, and DOE Grants DE-FG01-92ER-25126 and DE-FG02-08ER-58562.

The matrix completion problem

$$\begin{aligned} \min \operatorname{rank}(X) \\ \text{s.t. } X_{ij} = M_{ij}, (i, j) \in \Omega \end{aligned} \quad (1.2)$$

is a special case of (1.1), where X and M are both $m \times n$ matrices and Ω is a subset of index pairs (i, j) . The so called collaborative filtering problem [30; 33] can be cast as a matrix completion problem. Suppose some users in an online survey provide ratings of some movies. This yields a matrix M with users as rows and movies as columns whose (i, j) -th entry M_{ij} is the rating given by the i -th user to the j -th movie. Since most users rate only a small portion of the movies, we typically only know a small subset $\{M_{ij} | (i, j) \in \Omega\}$ of the entries. Based on the known ratings of a user, we want to predict the user's ratings of the movies that the user did not rate. That means we want to fill in the missing entries of the matrix based on the small portion of entries we observed. It is commonly believed that only a few factors contribute to an individual's tastes or preferences for movies. Thus the rating matrix M is likely to be of low rank. Finding this low-rank matrix M corresponds to solving the matrix completion problem (1.2).

When the matrix X is diagonal, problem (1.1) reduces to the cardinality minimization problem

$$\begin{aligned} \min \|x\|_0 \\ \text{s.t. } Ax = b, \end{aligned} \quad (1.3)$$

where $x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ and $\|x\|_0$ denotes the number of nonzeros in the vector x . This problem finds the sparsest solution to an underdetermined system of equations and has a wide range of applications in signal processing. This problem is NP-hard [27]. To get a more computationally tractable problem, we can replace $\|x\|_0$ by its convex envelope.

Definition 1 The convex envelope of a function $f: \mathcal{C} \rightarrow \mathbb{R}$ is defined as the largest convex function g such that $g(x) \leq f(x)$ for all $x \in \mathcal{C}$ (see e.g., [23]).

It is well known that the convex envelope of $\|x\|_0$ is $\|x\|_1$, the ℓ_1 norm of x , which is the sum of the absolute values of all components of x . Replacing the objective function $\|x\|_0$ in (1.3) by $\|x\|_1$ yields the so-called basis pursuit problem

$$\begin{aligned} \min \|x\|_1 \\ \text{s.t. } Ax = b. \end{aligned} \quad (1.4)$$

The basis pursuit problem has received an increasing amount of attention since the emergence of the field of compressed sensing (CS) [8; 12]. Compressed sensing theories connect the NP-hard problem (1.3) to the convex and computationally tractable problem (1.4) and provide guarantees for when an optimal solution to (1.4) gives an optimal solution to (1.3). In the cardinality minimization and basis pursuit problems (1.3) and (1.4), b is a vector of measurements of the signal x obtained using the sampling matrix A . The main result of compressed sensing is that when the signal x is sparse, i.e., $k := \|x\|_0 \ll n$, we can recover the signal by solving (1.4) with a very limited number of measurements, i.e., $m \ll n$, when A is a Gaussian random matrix or when it corresponds to a partial Fourier transformation. Note that if b is contaminated by noise, the constraint $Ax = b$ in (1.4) must be relaxed, resulting in either the problem

$$\begin{aligned} \min \|x\|_1 \\ \text{s.t. } \|Ax - b\|_2 \leq \theta \end{aligned} \quad (1.5)$$

or its Lagrangian version

$$\min \mu \|x\|_1 + \frac{1}{2} \|Ax - b\|_2^2, \quad (1.6)$$

where θ and μ are parameters. Algorithms for solving (1.4) and its variants (1.5) and (1.6) have been widely investigated and many algorithms have been suggested including convex optimization methods ([2; 10; 18; 22; 24]) and heuristic methods ([11; 13; 14; 36; 37]).

The rank of a matrix is the number of its positive singular values. The matrix rank minimization (1.1) is NP-hard in general due to the combinational nature of the function $\text{rank}(\cdot)$. Similar to the cardinality function $\|x\|_0$, we can replace $\text{rank}(X)$ by its convex envelope to get a convex and more computationally tractable approximation to (1.1). It turns out that the convex envelope of $\text{rank}(X)$ on the set $\{X \in \mathbb{R}^{m \times n} : \|X\|_2 \leq 1\}$ is the nuclear norm $\|X\|_*$ [16], i.e., the nuclear norm is the best convex approximation of the rank function over the unit ball of matrices with norm less than one, where $\|X\|_2$ is the operator norm of X . The nuclear norm and operator norm are defined as follows.

Definition 2 Nuclear norm and Operator norm. Assume that the matrix X has r positive singular values of $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$. The nuclear norm of X is defined as the sum of its singular values, i.e.,

$$\|X\|_* := \sum_{i=1}^r \sigma_i(X).$$

The operator norm of matrix X is defined as the largest singular value of X , i.e.,

$$\|X\|_2 := \sigma_1(X).$$

The nuclear norm is also known as Schatten 1-norm or Ky Fan norm. Using it as an approximation to $\text{rank}(X)$ in (1.1) yields the nuclear norm minimization problem

$$\begin{aligned} \min \|X\|_* \\ \text{s.t. } \mathcal{A}(X) = b. \end{aligned} \quad (1.7)$$

As in the basis pursuit problem, if b is contaminated by noise, the constraint $\mathcal{A}(X) = b$ must be relaxed, resulting in either the problem

$$\begin{aligned} \min \|X\|_* \\ \text{s.t. } \|\mathcal{A}(X) - b\|_2 \leq \theta \end{aligned}$$

or its Lagrangian version

$$\min \mu \|X\|_* + \frac{1}{2} \|\mathcal{A}(X) - b\|_2^2, \quad (1.8)$$

where θ and μ are parameters.

Note that if we write X in vector form by stacking the columns of X in a single vector $\text{vec}(X) \in \mathbb{R}^{mn}$, then we get the following equivalent formation of (1.7):

$$\begin{aligned} \min \|X\|_* \\ \text{s.t. } A \text{vec}(X) = b, \end{aligned} \quad (1.9)$$

where $A \in \mathbb{R}^{p \times mn}$ is the matrix corresponding to the linear map \mathcal{A} . An important question is: when will an optimal solution to the nuclear norm minimization problem (1.7) give an optimal solution to matrix rank minimization problem (1.1). In response to this question, Recht et al. [29] proved that if the entries of A are suitably random, e.g., i.i.d. Gaussian, then with very high probability, most $m \times n$ matrices of rank r can be recovered by solving the nuclear norm minimization (1.7) or equivalently, (1.9), whenever $p \geq Cr(m+n) \log(mn)$, where C is a positive constant.

For the matrix completion problem (1.2), the corresponding nuclear norm minimization problem is

$$\begin{aligned} \min \|X\|_* \\ \text{s.t. } X_{ij} = M_{ij}, (i, j) \in \Omega. \end{aligned} \quad (1.10)$$

Candès et al. [9] proved the following result.

Theorem 1 *Let M be an $n_1 \times n_2$ matrix of rank r with SVD*

$$M = \sum_{k=1}^r \sigma_k u_k v_k^\top,$$

where the family $\{u_k\}_{1 \leq k \leq r}$ is selected uniformly at random among all families of r orthonormal vectors, and similarly for the family $\{v_k\}_{1 \leq k \leq r}$. Let $n = \max(n_1, n_2)$. Suppose we observe m entries of M with locations sampled uniformly at random. Then there are constants C and c such that if

$$m \geq Cn^{5/4}r \log n,$$

the minimizer to the problem (1.10) is unique and equal to M with probability at least $1 - cn^{-3}$. In addition, if $r \leq n^{1/5}$, then the recovery is exact with probability at least $1 - cn^{-3}$ provided that

$$m \geq Cn^{6/5}r \log n.$$

This theorem states that a surprisingly small number of entries are sufficient to complete a low-rank matrix with high probability.

The dual problem corresponds to the nuclear norm minimization problem (1.7) is

$$\begin{aligned} \max b^\top z \\ \text{s.t. } \|\mathcal{A}^*(z)\|_2 \leq 1, \end{aligned} \quad (1.11)$$

where \mathcal{A}^* is the adjoint operator of \mathcal{A} . Both (1.7) and (1.11) can each be rewritten as an equivalent semidefinite programming (SDP) problem. The equivalent SDP formulation of (1.7) is:

$$\begin{aligned} \min_{X, W_1, W_2} \frac{1}{2}(\text{Tr}(W_1) + \text{Tr}(W_2)) \\ \text{s.t. } \begin{bmatrix} W_1 & X \\ X^\top & W_2 \end{bmatrix} \succeq 0 \\ \mathcal{A}(X) = b. \end{aligned} \quad (1.12)$$

The equivalent SDP formulation of (1.11) is:

$$\begin{aligned} \max_z b^\top z \\ \text{s.t. } \begin{bmatrix} I_m & \mathcal{A}^*(z) \\ \mathcal{A}^*(z)^\top & I_n \end{bmatrix} \succeq 0. \end{aligned} \quad (1.13)$$

Thus we can use SDP solvers such as SeDuMi [35] and SDPT3 [39] to solve (1.12) and (1.13) and therefore solve (1.7) and (1.11). Note that in (1.12), $X \in \mathbb{R}^{m \times n}$, $W_1 \in \mathbb{R}^{m \times m}$, $W_2 \in \mathbb{R}^{n \times n}$; so the number of variables is $mn + m^2 + n^2$. SDP solvers usually cannot solve a problem when m and n are both much larger than 100.

Recently, Liu and Vandenberghe [26] proposed an interior-point method for another nuclear norm approximation problem

$$\min \|\mathcal{A}(x) - B\|_*, \quad (1.14)$$

where $B \in \mathbb{R}^{m \times n}$ and

$$\mathcal{A}(x) = x_1 A_1 + x_2 A_2 + \cdots + x_p A_p$$

is a linear mapping from \mathbb{R}^p to $\mathbb{R}^{m \times n}$. The equivalent SDP formulation of (1.14) is

$$\begin{aligned} \min_{x, W_1, W_2} & \frac{1}{2} (\text{Tr}(W_1) + \text{Tr}(W_2)) \\ \text{s.t.} & \begin{bmatrix} W_1 & (\mathcal{A}(x) - B)^\top \\ \mathcal{A}(x) - B & W_2 \end{bmatrix} \succeq 0. \end{aligned} \quad (1.15)$$

Liu and Vandenberghe [26] proposed a customized method for computing the scaling direction in an interior point method for solving SDP (1.15). They can reduce the complexity in each iteration from $O(p^6)$ to $O(p^4)$ when $m = O(p)$ and $n = O(p)$ and thus can solve problems up to dimension $m = n = 350$. But solving larger problems with $m = n = 1000$ is prohibitive due to the large dimensionality.

Another algorithm for solving (1.7) is due to Burer and Monteiro [5; 6], (see also Rennie and Srebro [30; 33]). This algorithm uses the low-rank factorization $X = LR^\top$ of the matrix $X \in \mathbb{R}^{m \times n}$, where $L \in \mathbb{R}^{m \times r}$, $R \in \mathbb{R}^{n \times r}$, $r \leq \min\{m, n\}$, and solves the optimization problem

$$\begin{aligned} \min_{L, R} & \frac{1}{2} (\|L\|_F^2 + \|R\|_F^2) \\ \text{s.t.} & \mathcal{A}(LR^\top) = b. \end{aligned} \quad (1.16)$$

It is known that as long as r is chosen to be sufficiently larger than the rank of the optimal solution matrix of the nuclear norm problem (1.7), this low-rank factorization problem is equivalent to the nuclear norm problem (1.7) (see e.g., [29]). The advantage of this low-rank factorization formulation is that both the objective function and the constraints are differentiable. Thus gradient-based optimization algorithms such as conjugate gradient algorithms and augmented Lagrangian algorithms can be used to solve this problem. However, the constraints in this problem are nonconvex, so one can only be assured of obtaining a local minimizer. Also, how to choose r is still an open question.

Outline. The rest of this paper is organized as follows. In Section 2 we review the fixed-point continuation algorithm for ℓ_1 -regularized problems. In Section 3 we give an analogous fixed-point iterative algorithm for the nuclear norm minimization problem and prove that it converges to an optimal solution. In Section 4 we discuss a continuation technique for accelerating the convergence of our algorithm. In Section 5 we propose Bregman iterative algorithm for nuclear norm minimization extending the approach in [41] for compressed sensing to the rank minimization problem. In Section 6 we incorporate a Monte-Carlo approximate SVD procedure into our fixed-point continuation algorithm to speed it up and improve its ability to recover low-rank matrices. Numerical results for both synthesized matrices and real problems are given in Section 7. We present further directions for extending the approach proposed in [7; 21] for compressed sensing to the rank minimization problem and give conclusions in Section 8.

Notation. We adopt the following notation in this paper. $g^k = g(X^k) = \mathcal{A}^*(\mathcal{A}(X^k) - b)$ is the gradient of function $\frac{1}{2} \|\mathcal{A}(X) - b\|_2^2$ at point X^k , where \mathcal{A}^* is the adjoint operator of \mathcal{A} . $\|X\|_2$ denotes the largest singular value

of a matrix X . $\|x\|_2$ denotes the Euclidean norm of a vector x . $\|X\|_F$ denotes the Frobenius norm of the matrix X :

$$\|X\|_F := \left(\sum_{i=1}^r \sigma_i^2 \right)^{1/2} = \left(\sum_{i,j} X_{ij}^2 \right)^{1/2} = (\text{Tr}(XX^\top))^{1/2},$$

where r is the rank of X and σ_i is the i -th singular value of X and $\text{Tr}(X)$ denotes the trace of the square matrix X . Throughout this paper, we always assume that the singular values are arranged in nonincreasing order, i.e., $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0 = \sigma_{r+1} = \dots = \sigma_{\min\{m,n\}}$. ∂f denotes the subgradient of the function f . $\text{Diag}(s)$ denotes the diagonal matrix whose diagonal elements are the elements of the vector s . $\text{sgn}(t)$ is the signum function of $t \in \mathbb{R}$, i.e.,

$$\text{sgn}(t) := \begin{cases} +1 & \text{if } t > 0, \\ 0 & \text{if } t = 0, \\ -1 & \text{if } t < 0. \end{cases}$$

While the signum multifunction of $t \in \mathbb{R}$ is

$$\text{SGN}(t) := \partial|t| = \begin{cases} \{+1\} & \text{if } t > 0, \\ [-1, 1] & \text{if } t = 0, \\ \{-1\} & \text{if } t < 0. \end{cases}$$

We use $a \odot b$ to denote the elementwise multiplication of two vectors a and b . We use $X(k:l)$ to denote the submatrix of X consisting of the k -th to l -th column of X . We use \mathbb{R}_+^n to denote the nonnegative orthant in n -dimensional real vector space.

2 Fixed point iterative algorithm

A Fixed point iterative algorithm is proposed in [22] for the ℓ_1 -regularized problem (1.6). The idea behind this algorithm is an operator splitting technique. Note that x^* is an optimal solution to (1.6) if and only if

$$\mathbf{0} \in \mu \text{SGN}(x^*) + g^*, \quad (2.1)$$

where $g^* = A^\top(Ax^* - b)$. For any $\tau > 0$, (2.1) is equivalent to

$$\mathbf{0} \in \tau \mu \text{SGN}(x^*) + \tau g(x^*). \quad (2.2)$$

Note that the operator $T(\cdot) := \tau \mu \text{SGN}(\cdot) + \tau g(\cdot)$ on the right hand side of (2.2) can be split into two parts: $T(\cdot) = T_1(\cdot) - T_2(\cdot)$, where $T_1(\cdot) = \tau \mu \text{SGN}(\cdot) + I(\cdot)$ and $T_2(\cdot) = I(\cdot) - \tau g(\cdot)$.

Letting $y = T_2(x^*) = x^* - \tau A^\top(Ax^* - b)$, (2.2) is equivalent to

$$\mathbf{0} \in T_1(x^*) - y = \tau \mu \text{SGN}(x^*) + x^* - y. \quad (2.3)$$

Note that (2.3) is actually the optimality condition to the following convex problem

$$\min_{x^*} \tau \mu \|x^*\|_1 + \frac{1}{2} \|x^* - y\|_2^2. \quad (2.4)$$

This problem has a closed form optimal solution given by the so called shrinkage operator:

$$x^* = \tilde{s}_v(y),$$

where $\mathbf{v} = \tau\mu$, and shrinkage operator $\tilde{s}_{\mathbf{v}}(\cdot)$ is given by

$$\tilde{s}_{\mathbf{v}}(\cdot) = \text{sgn}(\cdot) \odot \max\{|\cdot| - \mathbf{v}, \mathbf{0}\}. \quad (2.5)$$

Thus, the fixed point iterative algorithm is given by

$$\mathbf{x}^{k+1} = \tilde{s}_{\tau\mu}(\mathbf{x}^k - \tau\mathbf{g}^k). \quad (2.6)$$

Hale et al. [22] proved global and finite convergence of this algorithm to the optimal solution of the ℓ_1 -regularized problem (1.6).

Motivated by this work, we can also develop a fixed point iterative algorithm for (1.7). Since the objective function in (1.8) is convex, X^* is the optimal solution to (1.8) if and only if

$$\mathbf{0} \in \mu\partial\|X^*\|_* + g(X^*), \quad (2.7)$$

where $g(X^*) = \mathcal{A}^*(\mathcal{A}(X^*) - b)$. Note that if the Singular Value Decomposition (SVD) of X is $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{m \times r}$, $\Sigma = \text{Diag}(\boldsymbol{\sigma}) \in \mathbb{R}^{r \times r}$, $V \in \mathbb{R}^{n \times r}$, then (see e.g., [1; 3])

$$\partial\|X\|_* = \{UV^\top + W : U^\top W = 0, WV = 0, \|W\| \leq 1\}.$$

Hence, we get the following optimality conditions for (1.8):

Theorem 2 *The matrix $X \in \mathbb{R}^{m \times n}$ with singular value decomposition $X = U\Sigma V^\top$, $U \in \mathbb{R}^{m \times r}$, $\Sigma = \text{Diag}(\boldsymbol{\sigma}) \in \mathbb{R}^{r \times r}$, $V \in \mathbb{R}^{n \times r}$, is optimal for the problem (1.8) if and only if there exists a matrix $W \in \mathbb{R}^{m \times n}$ such that*

$$\mu(UV^\top + W) + g(X) = 0, \quad (2.8a)$$

$$U^\top W = 0, WV = 0, \|W\|_2 \leq 1. \quad (2.8b)$$

Now based on the optimality condition (2.7), we can develop a fixed point iterative scheme for solving (1.8) by adopting the operator splitting technique described at the beginning of this section. Note that (2.7) is equivalent to

$$\mathbf{0} \in \tau\mu\partial\|X^*\|_* + X^* - (X^* - \tau\mathbf{g}(X^*)) \quad (2.9)$$

for any $\tau > 0$. If we let

$$Y^* = X^* - \tau\mathbf{g}(X^*),$$

then (2.9) is reduced to

$$\mathbf{0} \in \tau\mu\partial\|X^*\|_* + X^* - Y^*, \quad (2.10)$$

i.e., X^* is the optimal solution to

$$\min_{X \in \mathbb{R}^{m \times n}} \tau\mu\|X\|_* + \frac{1}{2}\|X - Y^*\|_F^2 \quad (2.11)$$

In the following we will prove that the matrix shrinkage operator applied to Y^* gives the optimal solution to (2.11).

First, we need the following definitions.

Definition 3 Nonnegative Vector Shrinkage Operator. Assume $x \in \mathbb{R}_+^n$. For any $\nu > 0$, the nonnegative vector shrinkage operator $s_\nu(\cdot)$ is defined as

$$s_\nu(x) := \bar{x}, \text{ with } \bar{x}_i = \begin{cases} x_i - \nu, & \text{if } x_i - \nu > 0 \\ 0, & \text{o.w.} \end{cases}$$

Definition 4 Matrix Shrinkage Operator. Assume $X \in \mathbb{R}^{m \times n}$ and the SVD of X is given by $X = U \text{Diag}(\sigma) V^\top$, $U \in \mathbb{R}^{m \times r}$, $\sigma \in \mathbb{R}_+^r$, $V \in \mathbb{R}^{n \times r}$. For any $\nu > 0$, the matrix shrinkage operator $S_\nu(\cdot)$ is defined as

$$S_\nu(X) := U \text{Diag}(\bar{\sigma}) V^\top, \quad \text{with } \bar{\sigma} = s_\nu(\sigma).$$

Theorem 3 Given a matrix $Y \in \mathbb{R}^{m \times n}$ with $\text{rank}(Y) = t$ and SVD $Y = U_Y \text{Diag}(\gamma) V_Y^\top$, where $U_Y \in \mathbb{R}^{m \times t}$, $\gamma \in \mathbb{R}_+^t$, $V_Y \in \mathbb{R}^{n \times t}$, and a scalar $\nu > 0$,

$$X := S_\nu(Y) = U_Y \text{Diag}(s_\nu(\gamma)) V_Y^\top \quad (2.12)$$

is an optimal solution of the problem

$$\min_{X \in \mathbb{R}^{m \times n}} f(X) := \nu \|X\|_* + \frac{1}{2} \|X - Y\|_F^2. \quad (2.13)$$

Proof Without loss of generality, we assume $m \leq n$. Suppose that the solution $X \in \mathbb{R}^{m \times n}$ to problem (2.13) has the SVD $X = U \text{Diag}(\sigma) V^\top$, where $U \in \mathbb{R}^{m \times r}$, $\sigma \in \mathbb{R}_+^r$, $V \in \mathbb{R}^{n \times r}$. Hence, X must satisfy the optimality conditions for (2.13) which are

$$\mathbf{0} \in \nu \partial \|X\|_* + X - Y;$$

i.e., there exists a matrix

$$W = \bar{U} \begin{bmatrix} \text{Diag}(\bar{\sigma}) & 0 \end{bmatrix} \bar{V}^\top,$$

where $\bar{U} \in \mathbb{R}^{m \times (m-r)}$, $\bar{V} \in \mathbb{R}^{n \times (n-r)}$, $\bar{\sigma} \in \mathbb{R}_+^{m-r}$, $\|\bar{\sigma}\|_\infty \leq 1$ and both $\hat{U} = [U, \bar{U}]$ and $\hat{V} = [V, \bar{V}]$ are orthogonal matrices, such that

$$\mathbf{0} = \nu(UV^\top + W) + X - Y. \quad (2.14)$$

Hence,

$$\hat{U} \begin{bmatrix} \nu I + \text{Diag}(\sigma) & 0 & 0 \\ 0 & \nu \text{Diag}(\bar{\sigma}) & 0 \end{bmatrix} \hat{V}^\top - U_Y \text{Diag}(\gamma) V_Y^\top = \mathbf{0}. \quad (2.15)$$

To verify that (2.12) satisfies (2.15), consider the following two cases:

Case 1: $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_t > \nu$. In this case, choosing X as above, with $r = t$, $U = U_Y$, $V = V_Y$ and $\sigma = s_\nu(\gamma) = \gamma - \nu e$, where e is a vector of r ones, and choosing $\bar{\sigma} = 0$ (i.e., $W = \mathbf{0}$) satisfies (2.15).

Case 2: $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_k > \nu \geq \gamma_{k+1} \geq \dots \geq \gamma_t$. In this case, by choosing $r = k$, $\hat{U}(1:t) = U_Y$, $\hat{V}(1:t) = V_Y$, $\sigma = s_\nu((\gamma_1, \dots, \gamma_k))$ and $\bar{\sigma}_1 = \gamma_{k+1}/\nu, \dots, \bar{\sigma}_{t-k} = \gamma_t/\nu, \bar{\sigma}_{t-k+1} = \dots = \bar{\sigma}_{m-r} = 0$, X and W satisfy (2.15).

Note that in both cases, X can be written as the form in (2.12) based on the way we construct X . \square

Based on the above we obtain the fixed point iterative scheme for solving problem (1.8):

$$\begin{cases} Y^k = X^k - \tau g(X^k) \\ X^{k+1} = S_{\tau\mu}(Y^k). \end{cases} \quad (2.16)$$

3 Convergence results

In this section, we analyze the convergence properties of the fixed point iterative scheme (2.16). Before we prove the main convergence result, we need some lemmas.

Lemma 1 *The shrinkage operator S_v is non-expansive, i.e., for any Y_1 and $Y_2 \in \mathbb{R}^{m \times n}$,*

$$\|S_v(Y_1) - S_v(Y_2)\|_F \leq \|Y_1 - Y_2\|_F. \quad (3.1)$$

Moreover, if Y_1 and Y_2 have the SVDs $Y_1 = U_1 \Sigma V_1^\top$, $Y_2 = U_2 \Gamma V_2^\top$, and we define

$$D(Y_1, Y_2) := \|Y_1 - Y_2\|_F^2 - \|S_v(Y_1) - S_v(Y_2)\|_F^2, \quad (3.2)$$

then for fixed Y_1 and Γ , $D(Y_1, Y_2)$ achieves its minimum when $U_2(1 : q) = U_1(1 : q)$ and $V_2(1 : q) = V_1(1 : q)$, where $q = \min\{\text{rank}(Y_1), \text{rank}(Y_2)\}$. Furthermore, the inequality (3.1) holds as an equality (i.e., the minimum of $D(Y_1, Y_2)$ is 0) only when $\sigma_i(Y_1) = \gamma_i(Y_2)$ if one of them is less than or equal to v , which also implies that

$$Y_1 - Y_2 = S_v(Y_1) - S_v(Y_2). \quad (3.3)$$

Proof Without loss of generality, we assume $m \leq n$. Let $Y_1 = U_1 \Sigma V_1^\top$, $Y_2 = U_2 \Gamma V_2^\top$, where

$$\Sigma = \begin{pmatrix} \text{Diag}(\sigma) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m \times n}, \Gamma = \begin{pmatrix} \text{Diag}(\gamma) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m \times n},$$

$\sigma = (\sigma_1, \dots, \sigma_s)$, $\sigma_1 \geq \dots \geq \sigma_s > 0$ and $\gamma = (\gamma_1, \dots, \gamma_t)$, $\gamma_1 \geq \dots \geq \gamma_t > 0$. Note that here U_1, V_1, U_2 and V_2 are (full) orthogonal matrices; $\Sigma, \Gamma \in \mathbb{R}^{m \times n}$. Suppose that $\sigma_1 \geq \dots \geq \sigma_k > v \geq \sigma_{k+1} \geq \dots \geq \sigma_s$ and $\gamma_1 \geq \dots \geq \gamma_l > v \geq \gamma_{l+1} \geq \dots \geq \gamma_t$, then

$$\bar{Y}_1 := S_v(Y_1) = U_1 \bar{\Sigma} V_1^\top, \bar{Y}_2 := S_v(Y_2) = U_2 \bar{\Gamma} V_2^\top,$$

where

$$\bar{\Sigma} = \begin{pmatrix} \text{Diag}(\bar{\sigma}) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m \times n}, \bar{\Gamma} = \begin{pmatrix} \text{Diag}(\bar{\gamma}) & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m \times n},$$

$\bar{\sigma} = (\sigma_1 - v, \dots, \sigma_k - v)$ and $\bar{\gamma} = (\gamma_1 - v, \dots, \gamma_l - v)$.

Thus,

$$\begin{aligned} \|Y_1 - Y_2\|_F^2 - \|\bar{Y}_1 - \bar{Y}_2\|_F^2 &= \text{Tr}((Y_1 - Y_2)^\top (Y_1 - Y_2)) - \text{Tr}((\bar{Y}_1 - \bar{Y}_2)^\top (\bar{Y}_1 - \bar{Y}_2)) \\ &= \text{Tr}(Y_1^\top Y_1 - \bar{Y}_1^\top \bar{Y}_1 + Y_2^\top Y_2 - \bar{Y}_2^\top \bar{Y}_2) - 2\text{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2) \\ &= \sum_{i=1}^s \sigma_i^2 - \sum_{i=1}^k (\sigma_i - v)^2 + \sum_{i=1}^t \gamma_i^2 - \sum_{i=1}^l (\gamma_i - v)^2 - 2\text{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2) \end{aligned}$$

Now let us derive an upper bound for $\text{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2)$.

$$\begin{aligned} \text{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2) &= \text{Tr}(V_1 \Sigma^\top U_1^\top U_2 \Gamma V_2^\top - V_1 \bar{\Sigma}^\top U_1^\top U_2 \bar{\Gamma} V_2^\top) \\ &= \text{Tr}(\Sigma^\top U_1^\top U_2 \Gamma V_2^\top V_1 - \bar{\Sigma}^\top U_1^\top U_2 \bar{\Gamma} V_2^\top V_1). \end{aligned}$$

Let $U = U_1^\top U_2, V = V_1^\top V_2$. Then

$$\begin{aligned} \text{Tr}(Y_1^\top Y_2 - \bar{Y}_1^\top \bar{Y}_2) &= \text{Tr}(\Sigma^\top U \Gamma V^\top - \bar{\Sigma}^\top U \bar{\Gamma} V^\top) \\ &= \text{Tr}(\Sigma^\top U \Gamma V^\top - \Sigma^\top U \bar{\Gamma} V^\top + \Sigma^\top U \bar{\Gamma} V^\top - \bar{\Sigma}^\top U \bar{\Gamma} V^\top) \\ &= \text{Tr}(\Sigma^\top U (\Gamma - \bar{\Gamma}) V^\top) + \text{Tr}((\Sigma - \bar{\Sigma})^\top U \bar{\Gamma} V^\top) \end{aligned}$$

Both $\text{Tr}(\Sigma^\top U (\Gamma - \bar{\Gamma}) V^\top)$ and $\text{Tr}((\Sigma - \bar{\Sigma})^\top U \bar{\Gamma} V^\top)$ achieve their maximum value when $u_{ii} = v_{ii} = 1, i = 1, \dots, s$. To see this, just note that the KKT optimality conditions for the problem

$$\max\{\text{Tr}(DQ\bar{D}\bar{Q}) \mid Q^\top Q = I, \bar{Q}^\top \bar{Q} = I\},$$

where D and \bar{D} are nonnegative diagonal matrices, are $\bar{D}\bar{Q}D - \Lambda Q = 0$ and $DQ\bar{D} - \Omega \bar{Q} = 0$, which are satisfied by $Q = \bar{Q} = I, \Lambda = \Omega = D\bar{D}$.

Thus, without loss of generality, assuming $k \leq l \leq s \leq t$, we have,

$$\begin{aligned} &\|Y_1 - Y_2\|_F^2 - \|S_v(Y_1) - S_v(Y_2)\|_F^2 \\ &\geq \sum_{i=1}^s \sigma_i^2 - \sum_{i=1}^k (\sigma_i - v)^2 + \sum_{i=1}^l \gamma_i^2 - \sum_{i=1}^l (\gamma_i - v)^2 - 2\left(\sum_{i=1}^l \sigma_i v + \sum_{i=l+1}^s \sigma_i \gamma_i + \sum_{i=1}^k (\gamma_i - v)v + \sum_{i=k+1}^l \sigma_i (\gamma_i - v)\right) \\ &= \sum_{i=k+1}^l (2\gamma_i v - v^2 + \sigma_i^2 - 2\sigma_i \gamma_i) + \left(\sum_{i=l+1}^s \sigma_i^2 + \sum_{i=l+1}^l \gamma_i^2 - \sum_{i=l+1}^s 2\sigma_i \gamma_i\right). \end{aligned}$$

Now

$$\sum_{i=l+1}^s \sigma_i^2 + \sum_{i=l+1}^l \gamma_i^2 - \sum_{i=l+1}^s 2\sigma_i \gamma_i \geq 0$$

since $t \geq s$ and $\sigma_i^2 + \gamma_i^2 - 2\sigma_i \gamma_i \geq 0$. Also, since the function $f(x) := 2\gamma_i x - x^2$ is monotonely increasing in $(-\infty, \gamma_i]$ and $\sigma_i \leq v < \gamma_i, i = k+1, \dots, l$,

$$2\gamma_i v - v^2 + \sigma_i^2 - 2\sigma_i \gamma_i \geq 0, i = k+1, \dots, l.$$

Thus we get

$$\|Y_1 - Y_2\|_F^2 - \|S_v(Y_1) - S_v(Y_2)\|_F^2 \geq 0;$$

i.e., (3.1) holds.

Also, $D(Y_1, Y_2)$ achieves its minimum when $u_{ii} = v_{ii} = 1, i = 1, \dots, s$; i.e., $U_1(1:s) = U_2(1:s)$ and $V_1(1:s) = V_2(1:s)$. Furthermore, if equality in (3.1) holds, then $k = l, s = t$, and $\sigma_i = \gamma_i, i = k+1, \dots, s$, which further implies

$$Y_1 - Y_2 = S_v(Y_1) - S_v(Y_2).$$

□

Lemma 2 Let $\mathcal{A}X = \text{Avec}(X)$ and assume that $\tau \in (0, 2/\lambda_{\max}(A^\top A))$. Then the operator $h(\cdot) = I(\cdot) - \tau g(\cdot)$ is non-expansive. Moreover, $g(X) = g(X')$ if $\|h(X) - h(X')\|_F = \|X - X'\|_F$.

Proof First, we note that

$$h(X) - h(X') = X - X' - \tau(g(X) - g(X')).$$

Thus,

$$\begin{aligned}
\|h(X) - h(X')\|_F &= \|X - X' - \tau(g(X) - g(X'))\|_F \\
&= \|(I - \tau A^\top A)(\text{vec}(X) - \text{vec}(X'))\|_2 \\
&\leq \max\{|1 - \tau\lambda_{\max}(A^\top A)|, |1 - \tau\lambda_{\min}(A^\top A)|\} \|\text{vec}(X) - \text{vec}(X')\|_2 \\
&\leq \|\text{vec}(X) - \text{vec}(X')\|_2 \\
&= \|X - X'\|_F.
\end{aligned}$$

Let $Z = X - X'$ and $w := (A^\top A)^{1/2} \text{vec}(Z)$. Then

$$\begin{aligned}
&\|h(X) - h(X')\|_F = \|X - X'\|_F \\
&\iff \|\text{vec}(Z) - \tau A^\top A \text{vec}(Z)\|_2 = \|\text{vec}(Z)\|_2 \\
&\iff -2\tau \text{vec}(Z)^\top A^\top A \text{vec}(Z) + \tau^2 \text{vec}(Z)^\top (A^\top A)^2 \text{vec}(Z) = 0 \\
&\iff \tau w^\top A^\top A w = 2w^\top w \\
&\implies \tau \frac{w^\top A^\top A w}{w^\top w} = 2 \text{ if } w \neq 0,
\end{aligned}$$

which contradicts $\tau < 2/\lambda_{\max}(A^\top A)$ since $\frac{w^\top A^\top A w}{w^\top w} \leq \lambda_{\max}(A^\top A)$. Hence, $w = 0$, so that

$$g(X) - g(X') = (A^\top A)^{1/2} w = 0,$$

which completes the proof. \square

In the following, we always choose $\tau \in (0, 2/\lambda_{\max}(A^\top A))$ to guarantee that $h(\cdot)$ is non-expansive.

Lemma 3 *If*

$$\|S_V(h(X)) - S_V(h(X^*))\|_F \equiv \|S_V(h(X)) - X^*\|_F = \|X - X^*\|_F, \quad (3.4)$$

then X is a fixed point, that is,

$$X = S_V(h(X)).$$

Proof From (3.4),

$$\|X - X^*\|_F = \|S_V(h(X)) - S_V(h(X^*))\|_F \leq \|h(X) - h(X^*)\|_F \leq \|X - X^*\|_F.$$

Hence, both inequalities hold with equality. From Lemma 1, we have

$$S_V(h(X)) - S_V(h(X^*)) = h(X) - h(X^*) = X - X^* - \tau(g(X) - g(X^*)).$$

Since $S_V(h(X^*)) = X^*$, we get from the equality above

$$S_V(h(X)) = X - \tau(g(X) - g(X^*)).$$

From Lemma 2, we have $g(X) - g(X^*) = 0$ since $\|h(X) - h(X^*)\|_F = \|X - X^*\|_F$. Thus, X is a fixed point. \square

Lemma 4 *The sequence $\{\|X^k - X^*\|_F\}$ is non-increasing.*

Proof $S_V(\cdot)$ and $h(\cdot)$ are both non-expansive operators. Thus

$$\|X^{k+1} - X^*\|_F = \|S_V(h(X^k)) - S_V(h(X^*))\|_F \leq \|h(X^k) - h(X^*)\|_F \leq \|X^k - X^*\|_F.$$

\square

We now claim that the fixed-point iterations (2.16) converge to an optimal solution of (1.8).

Theorem 4 *The sequence $\{X^k\}$ generated by the fixed point iterations converges to some $X^* \in \mathcal{X}^*$, where \mathcal{X}^* is the optimal set of problem (1.8).*

Proof Since $S_V(h(\cdot))$ is non-expansive, $\{X^k\}$ lies in a compact set and must have a limit point, say

$$\bar{X} = \lim_{j \rightarrow \infty} X^{k_j}.$$

Since for any given fixed point X^* the sequence $\{\|X^k - X^*\|_F\}$ is monotonically non-increasing, it has a limit which can be written as

$$\lim_{k \rightarrow \infty} \|X^k - X^*\|_F = \|\bar{X} - X^*\|_F, \quad (3.5)$$

where \bar{X} can be any limit point of $\{X^k\}$. That is, all limit points, if more than one exists, must have an equal distance to any given fixed point $X^* \in \mathcal{X}^*$.

By the continuity of $S_V(h(\cdot))$, the image of \bar{X} ,

$$S_V(h(\bar{X})) = \lim_{j \rightarrow \infty} S_V(h(X^{k_j})) = \lim_{j \rightarrow \infty} X^{k_j+1},$$

is also a limit point of $\{X^k\}$. Therefore, we have

$$\|S_V(h(\bar{X})) - S_V(h(X^*))\|_F = \|\bar{X} - X^*\|_F,$$

which allows us to apply Lemma 3 to \bar{X} and establish the optimality of \bar{X} .

Finally, by setting $X^* = \bar{X} \in \mathcal{X}^*$ in (3.5), we get that $\{X^k\}$ converges to its unique limit point \bar{X} :

$$\lim_{k \rightarrow \infty} \|X^k - \bar{X}\|_F = 0.$$

□

4 Fixed point continuation

In this section, we discuss a continuation technique (i.e., homotopy approach) for accelerating the convergence of the fixed point iterative algorithm (2.16).

4.1 Continuation

Inspired by the work of Hale et al. [22], we first describe a continuation technique to accelerate the convergence of the fixed point iteration (2.16). Our fixed point continuation (FPC) iterative scheme for solving (1.8) is outlined below. The parameter η_μ determines the rate of reduction of the consecutive μ_k , i.e.,

$$\mu_{k+1} = \max\{\mu_k \eta_\mu, \bar{\mu}\}, \quad k = 1, \dots, L-1$$

4.2 Stopping criteria for inner iterations

Note that in the fixed point continuation algorithm, in the k -th inner iteration we solve (1.8) for a fixed $\mu = \mu_k$. There are several ways to determine when to stop this inner iteration, decrease μ and go to the next inner iteration. The

Fixed Point Continuation (FPC)

- Initialize: Given X_0 , $\bar{\mu} > 0$. Select $\mu_1 > \mu_2 > \dots > \mu_L = \bar{\mu} > 0$. Set $X = X_0$.
 - **for** $\mu = \mu_1, \mu_2, \dots, \mu_L$, **do**
 - **while** NOT converged, **do**
 - select $\tau > 0$
 - compute $Y = X - \tau \mathcal{A}^*(\mathcal{A}(X) - b)$, and SVD of Y , $Y = U \text{Diag}(\sigma) V^\top$
 - compute $X = U \text{Diag}(s_{\tau\mu}(\sigma)) V^\top$
 - **end while**
 - **end for**
-

optimality condition for (1.8) is given by (2.8a) and (2.8b). Thus we can use the following condition as a stopping criterion:

$$\|U_k V_k^\top + g^k / \mu\|_2 - 1 < gtol, \quad (4.1)$$

where $gtol$ is a small positive parameter. However, computing the largest singular value of a large matrix imposes a large burden that greatly decreases the speed of the algorithm. Hence, we do not use this criterion as a stopping rule for large matrices. Instead, we use the following criterion to stop the inner iteration.

When X^k is getting close to an optimal solution X^* , X^k and X^{k+1} should be very close to each other. Thus, another stopping criterion is

$$\frac{\|X^{k+1} - X^k\|_F}{\max\{1, \|X^k\|_F\}} < xtol, \quad (4.2)$$

where $xtol$ is a small positive number.

4.3 Debiasing

Debiasing is another technique that can improve the performance of FPC. Debiasing has been used in compressed sensing algorithms for solving (1.4) and its variants, where debiasing is performed after a support set \mathcal{S} has been tentatively identified. Debiasing is the process of solving a least squares problem restricted to the support set \mathcal{S} , i.e., we solve

$$\min \|A_{\mathcal{S}} x_{\mathcal{S}} - b\|_2, \quad (4.3)$$

where $A_{\mathcal{S}}$ is a submatrix of A whose columns correspond to the support index set \mathcal{S} , and $x_{\mathcal{S}}$ is a subvector of x corresponding to \mathcal{S} .

Our debiasing procedure for the matrix completion problem differs from the procedure used in compressed sensing since the concept of a support set is not applicable. When we do debiasing, we fix the matrices U and V in the singular value decomposition and then solve a least squares problem to determine the correct singular value $\sigma \in \mathbb{R}_+^r$; i.e., we solve

$$\min_{\sigma \geq 0} \|\mathcal{A}(U \text{Diag}(\sigma) V) - b\|_2, \quad (4.4)$$

where r is the rank of current point X . Because debiasing can be costly, we use a rule proposed in [40] to decide when to do it. In the continuation framework, we know that in each subproblem with a fixed μ , $\|X_{k+1} - X_k\|_F$ converges to zero, and $\|g\|_2$ converges to μ when X_k converges to the optimal solution of the subproblem. We therefore choose to do debiasing when $\|g\|_2 / \|X_{k+1} - X_k\|_F$ becomes large because this indicates that the change

between two consecutive iterates is relatively small. Specifically, we call for debiasing in the solver FPC3 (see Section 7) when $\|g\|_2/\|X_{k+1} - X_k\|_F > 10$.

5 Bregman iterative algorithm

Algorithm FPC is designed to solve (1.8), an optimal solution of which approaches an optimal solution of the nuclear norm minimization problem (1.7) as μ goes to zero. However, by incorporating FPC into a Bregman iterative technique, we can solve (1.7) by solving a limited number of instances of (1.8), each corresponding to a different b .

Given a convex function $J(\cdot)$, the Bregman distance [4] of point u from point v is defined as

$$D_J^p(u, v) := J(u) - J(v) - \langle p, u - v \rangle, \quad (5.1)$$

where $p \in \partial J(v)$ is some subgradient in the subdifferential of J at the point v .

Bregman iterative regularization was introduced by Osher et al. in the context of image processing [28]. They extended the Rudin-Osher-Fatemi [31] model

$$u = \operatorname{argmin}_u \mu \int |\nabla u| + \frac{1}{2} \|u - b\|_2^2 \quad (5.2)$$

into an iterative regularization model based on the Bregman distance related to the total variation functional

$$J(u) = \mu TV(u) = \mu \int |\nabla u|.$$

The Bregman iterative regularization procedure of Osher et al. [28] recursively solves

$$u^{k+1} \leftarrow \min_u D_J^{p^k}(u, u^k) + \frac{1}{2} \|u - b\|_2^2 \quad (5.3)$$

for $k = 0, 1, \dots$ starting with $u^0 = \mathbf{0}$ and $p^0 = \mathbf{0}$. Since (5.3) is a convex programming problem, the optimality conditions are given by $\mathbf{0} \in \partial J(u^{k+1}) - p^k + u^{k+1} - b$, from which we get the update formula for p^{k+1} :

$$p^{k+1} := p^k + b - u^{k+1}. \quad (5.4)$$

Therefore, the Bregman iterative scheme is given by

$$\begin{cases} u^{k+1} \leftarrow \min_u D_J^{p^k}(u, u^k) + \frac{1}{2} \|u - b\|_2^2 \\ p^{k+1} = p^k + b - u^{k+1}. \end{cases} \quad (5.5)$$

Interestingly, this turns out to be equivalent to the iterative process

$$\begin{cases} b^{k+1} = b + (b^k - u^k) \\ u^{k+1} \leftarrow \min_u J(u) + \frac{1}{2} \|u - b^{k+1}\|_2^2. \end{cases} \quad (5.6)$$

Iteration (5.6) can be easily conducted using existing algorithms for (5.2) with different inputs b .

Subsequently, Yin et al. [41] proposed solving the basis pursuit problem (1.4) by applying the Bregman iterative regularization algorithm to

$$\min_x J(x) + \frac{1}{2} \|Ax - b\|_2^2 \quad (5.7)$$

for $J(x) = \mu \|x\|_1$, and obtained the following two equivalent iterative schemes.

- Version 1:
 - $x^0 \leftarrow \mathbf{0}, p^0 \leftarrow \mathbf{0}$,
 - for $k = 0, 1, \dots$ do
 - $x^{k+1} \leftarrow \operatorname{argmin}_x D_J^{p^k}(x, x^k) + \frac{1}{2} \|Ax - b\|_2^2$
 - $p^{k+1} \leftarrow p^k - A^\top (Ax^{k+1} - b)$
- Version 2:
 - $b^0 \leftarrow \mathbf{0}, x^0 \leftarrow \mathbf{0}$,
 - for $k = 0, 1, \dots$ do
 - $b^{k+1} \leftarrow b + (b^k - Ax^k)$
 - $x^{k+1} \leftarrow \operatorname{argmin}_x J(x) + \frac{1}{2} \|Ax - b^{k+1}\|_2^2$.

For the nuclear norm minimization problem (1.7), one can use the same Bregman iterative regularization algorithm. Instead of solving (1.7) directly, one applies the Bregman iterative algorithm to the unconstrained problem (1.8), i.e., one iteratively solves (1.8) by

$$X^{k+1} \leftarrow \min_X D_J^{p^k}(X, X^k) + \frac{1}{2} \|\mathcal{A}(X) - b\|_2^2, \quad (5.8)$$

and updates the subgradient p^{k+1} by

$$p^{k+1} := p^k - \mathcal{A}^*(A(X^{k+1}) - b), \quad (5.9)$$

where $J(X) = \mu \|X\|_*$.

Equivalently, one can also use the following iterative scheme:

$$\begin{cases} b^{k+1} \leftarrow b + (b^k - \mathcal{A}(X^k)) \\ X^{k+1} \leftarrow \operatorname{argmin}_X \mu \|X\|_* + \frac{1}{2} \|\mathcal{A}(X) - b^{k+1}\|_2^2 \end{cases} \quad (5.10)$$

Thus, our Bregman iterative algorithm for nuclear norm minimization (1.7) can be outlined as follows. The last

Bregman Iterative Algorithm

- $b^0 \leftarrow \mathbf{0}, X^0 \leftarrow \mathbf{0}$,
 - for $k = 0, 1, \dots$ do
 - $b^{k+1} \leftarrow b + (b^k - \mathcal{A}(X^k))$,
 - $X^{k+1} \leftarrow \operatorname{argmin}_X \mu \|X\|_* + \frac{1}{2} \|\mathcal{A}(X) - b^{k+1}\|_2^2$.
-

step in it can be solved by Algorithm FPC.

6 An approximate SVD based FPC algorithm

Computing singular value decompositions is the main computational cost in Algorithm FPC. One way to reduce the time for computing SVDs is to compute an approximate SVD by a fast Monte Carlo algorithm such as the so-called Linear Time SVD algorithm developed by Drineas et al. [15]. For a given matrix $A \in \mathbb{R}^{m \times n}$, and parameters $c_s, k_s \in \mathbb{Z}^+$ with $1 \leq k_s \leq c_s \leq n$ and $\{p_i\}_{i=1}^n, p_i \geq 0, \sum_{i=1}^n p_i = 1$, this algorithm returns an approximation to the largest k_s singular values and corresponding left singular vectors of the matrix A in linear $O(m+n)$ time. The Linear Time SVD Algorithm is outlined below.

Linear Time Approximate SVD Algorithm[15]

- **Input:** $A \in \mathbb{R}^{m \times n}$, $c_s, k_s \in \mathbb{Z}^+$ s.t. $1 \leq k_s \leq c_s \leq n$, $\{p_i\}_{i=1}^n$ s.t. $p_i \geq 0, \sum_{i=1}^n p_i = 1$.
 - **Output:** $H_k \in \mathbb{R}^{m \times k_s}$ and $\sigma_t(C), t = 1, \dots, k_s$.
 - For $t = 1$ to c_s ,
 - Pick $i_t \in 1, \dots, n$ with $Pr[i_t = \alpha] = p_\alpha, \alpha = 1, \dots, n$.
 - Set $C^{(t)} = A^{(i_t)} / \sqrt{c_s p_{i_t}}$.
 - Compute $C^\top C$ and its SVD; say $C^\top C = \sum_{t=1}^{c_s} \sigma_t^2(C) y^t y^{t\top}$.
 - Compute $h^t = C y^t / \sigma_t(C)$ for $t = 1, \dots, k_s$.
 - Return H_{k_s} , where $H_{k_s}^{(t)} = h^t$, and $\sigma_t(C), t = 1, \dots, k_s$.
-

The outputs $\sigma_t(C), t = 1, \dots, k_s$ are approximations to the largest k_s singular values and $H_{k_s}^{(t)}, t = 1, \dots, k_s$ are approximations to the corresponding left singular vectors of the matrix A . Thus, the SVD of A is approximated by

$$A \approx A_{k_s} := H_{k_s} \text{Diag}(\sigma(C)) (A^\top H_{k_s} \text{Diag}(1/\sigma(C)))^\top.$$

Drineas et al. [15] proved that with high probability, the following estimate holds for both $\xi = 2$ and $\xi = F$:

$$\|A - A_{k_s}\|_\xi^2 \leq \min_{D: \text{rank}(D) \leq k_s} \|A - D\|_\xi^2 + \text{poly}(k_s, 1/c_s) \|A\|_F^2, \quad (6.1)$$

where $\text{poly}(k_s, 1/c_s)$ is a polynomial in k_s and $1/c_s$. Thus, A_{k_s} is a approximation to the best rank- k_s approximation to A . (For any matrix $M \in \mathbb{R}^{m \times n}$ with SVD $M = \sum_{i=1}^r \sigma_i u_i v_i^\top$, where $\sigma_1 \geq \dots \geq \sigma_r > 0, u_i \in \mathbb{R}^m, v_i \in \mathbb{R}^n$, the best rank- k approximation to M is given by $\bar{M} = \sum_{i=1}^k \sigma_i u_i v_i^\top$).

Note that in this algorithm, we compute an exact SVD of a smaller matrix $C^\top C \in \mathbb{R}^{c_s \times c_s}$. Thus, c_s determines the speed of this algorithm. If we choose a large c_s , we need more time to compute the SVD of $C^\top C$. However, the larger c_s is, the more likely are the $\sigma_t(C), t = 1, \dots, k_s$ to be close to the largest k_s singular values of the matrix A since the second term in the right hand side of (6.1) is smaller. In our numerical experiments, we found that we could choose a relatively small c_s so that the computational time was reduced without significantly degrading the accuracy. For example, we chose $c_s = 20$ when $m = n = 40$, $c_s = 25$ when $m = n = 100$ and $c_s = 100$ when $m = n = 1000$ and obtained very good results.

In our numerical experiments, we set k_s using the following procedure. In the k -th iteration, when computing the approximate SVD of $Y^k = X^k - \tau g^k$, we set k_s equal to the number of components in \bar{s}_{k-1} that are no less than $\epsilon_{k_s} \max\{\bar{s}_{k-1}\}$, where ϵ_{k_s} is a small positive number and $\max\{\bar{s}_{k-1}\}$ is the largest component in the vector \bar{s}_{k-1} used to form $X^k = U^{k-1} \text{Diag}(\bar{s}_{k-1}) V^{k-1\top}$. Note that k_s is non-increasing in this procedure. However, if k_s is too small at some iteration, the non-expansive property (3.1) of the shrinkage operator S_V may be violated since the approximate SVD is not a valid approximation when k_s is too small. Thus, in Algorithm FPC, if (3.1) is violated 10 times, we increase k_s by 1. Our numerical experience indicates that this technique makes our algorithm very robust.

Our numerical results in Section 7 show that this approximate SVD based FPC algorithm is very fast, robust, and significantly outperforms other solvers (such as SDPT3) in recovering low-rank matrices. One reason for this is that in the approximate SVD algorithm, we compute a low-rank approximation to the original matrix. Hence, the iterative matrices produced by our algorithm are more likely to be low-rank, which is exactly what we want.

7 Numerical results

In this section, we report on the application of our FPC and Bregman iterative algorithms to a series of matrix completion problems of the form (1.2) to demonstrate the ability of these algorithms to efficiently recover low-rank matrices.

To illustrate the performance of our algorithmic approach combined with exact and approximate SVD algorithms, different stopping rules, and with or without debiasing, we tested the following solvers.

- FPC1. Exact SVD, no debiasing, stopping rule: (4.2).
- FPC2. Exact SVD, no debiasing, stopping rule: (4.1) and (4.2).
- FPC3. Exact SVD with debiasing, stopping rule: (4.2).
- FPCA. Approximate SVD, no debiasing, stopping rule: (4.2).
- Bregman. Bregman iterative method using FPC2 to solve the subproblems.

7.1 FPC and Bregman iterative algorithms for random matrices

In our first series of tests, we created random matrices $M \in \mathbb{R}^{m \times n}$ with rank r by the following procedure: we first generated random matrices $M_L \in \mathbb{R}^{m \times r}$ and $M_R \in \mathbb{R}^{n \times r}$ with i.i.d. Gaussian entries and then set $M = M_L M_R^\top$. We then sampled a subset Ω of p entries uniformly at random. For each problem with $m \times n$ matrix M , measurement number p and rank r , we solved 50 randomly created matrix completion problems. We use $SR = p/(mn)$, i.e., the number of measurements divided by the number of entries of the matrix, to denote the sampling ratio. We also list $FR = r(m+n-r)/p$, i.e. the dimension of the set of rank r matrices divided by the number of measurements, in the tables. Note that if $FR > 1$, then there is always an infinite number of matrices with rank r with the given entries, so we cannot hope to recover the matrix in this situation. We use NS to denote the number of matrices that are recovered successfully. We use AT to denote the average time (seconds) for the examples that are successfully solved.

We used the relative error

$$rel.err. := \frac{\|X_{opt} - M\|_F}{\|M\|_F}$$

to estimate the closeness of X_{opt} to M , where X_{opt} is the ‘‘optimal’’ solution to (1.10) produced by our algorithms. We declared M to be recovered if the relative error was less than 10^{-3} , which is the criterion used in [29] and [9]. We use RA, RU, RL to denote the average, largest and smallest relative error of the successfully recovered matrices, respectively.

We summarize the parameter settings used by the algorithms in Table 1. We also set the maximum number of iterations allowed for solving each subproblem in FPC equal to 500, i.e., if the stopping rules (4.2) (and (4.1)) are not satisfied after 500 iterations, we terminate the subproblem and decrease μ to start the next subproblem.

Table 1 Parameters in Algorithm FPC

FPC	$\bar{\mu} = 10^{-8}, \mu_1 = \ \mathcal{A}^* b\ _2, \eta_\mu = 1/4, \tau = 1, xtol = 10^{-10}, gtol = 10^{-4}$	
Approx SVD	$m = n = 40$	$c_s = 20, \varepsilon_{k_s} = 10^{-2}, p_i = 1/n, \forall i$
	$m = n = 100$	$c_s = 25, \varepsilon_{k_s} = 10^{-2}, p_i = 1/n, \forall i$
	$m = n = 1000$	$c_s = 100, \varepsilon_{k_s} = 10^{-2}, p_i = 1/n, \forall i$

All numerical experiments were run in MATLAB 7.3.0 on a Dell Precision 670 workstation with an Intel Xeon(TM) 3.4GHZ CPU and 6GB of RAM.

The comparisons between FPC1, FPC2, FPC3 and SDPT3 for small matrix completion problems are presented in Table 2. From Table 2 we can see that FPC1 and FPC2 achieve almost the same recoverability and relative error, which means that as long as we set xtol to be very small (like 10^{-10}), we only need to use (4.2) as the stopping rule for the inner iterations. That is, use of stopping rule (4.1) does not affect the performance of the algorithm. Of course FPC2 costs more time than FPC1 since more iterations are sometimes needed to satisfy the stopping rules in FPC2. While FPC3 can improve the recoverability, it costs more time for performing debiasing. SDPT3 seems to obtain more accurate solutions than FPC1, FPC2 or FPC3.

Table 2 Comparisons of FPC1, FPC2, FPC3 and SDPT3 for randomly created small matrix completion problems ($m=n=40$, $p=800$, $SR=0.5$)

r	FR	Solver	NS	AT	RA	RU	RL
1	0.0988	FPC1	50	1.81	1.67e-9	1.22e-8	6.06e-10
		FPC2	50	3.61	1.32e-9	1.20e-8	2.55e-10
		FPC3	50	16.81	1.06e-9	2.22e-9	5.68e-10
		SDPT3	50	1.81	6.30e-10	3.46e-9	8.72e-11
2	0.1950	FPC1	42	3.05	1.01e-6	4.23e-5	8.36e-10
		FPC2	42	17.97	1.01e-6	4.23e-5	2.78e-10
		FPC3	49	16.86	1.26e-5	3.53e-4	7.62e-10
		SDPT3	44	1.90	1.50e-9	7.18e-9	1.82e-10
3	0.2888	FPC1	35	5.50	9.72e-9	2.85e-8	1.93e-9
		FPC2	35	20.33	2.17e-9	1.41e-8	3.88e-10
		FPC3	42	16.87	3.58e-5	7.40e-4	1.34e-9
		SDPT3	37	1.95	2.66e-9	1.58e-8	3.08e-10
4	0.3800	FPC1	22	9.08	7.91e-5	5.46e-4	3.57e-9
		FPC2	22	18.43	7.91e-5	5.46e-4	4.87e-10
		FPC3	29	16.95	3.83e-5	6.18e-4	2.57e-9
		SDPT3	29	2.09	1.18e-8	7.03e-8	7.97e-10
5	0.4688	FPC1	1	10.41	2.10e-8	2.10e-8	2.10e-8
		FPC2	1	17.88	2.70e-9	2.70e-9	2.70e-9
		FPC3	5	16.70	1.78e-4	6.73e-4	6.33e-9
		SDPT3	8	2.26	1.83e-7	8.12e-7	2.56e-9
6	0.5550	FPC1	0	—	—	—	—
		FPC2	0	—	—	—	—
		FPC3	0	—	—	—	—
		SDPT3	1	2.87	6.58e-7	6.58e-7	6.58e-7

To illustrate the performance of Bregman iterative algorithm, we compare the results of our Bregman iterative algorithm and FPC2 in Table 3. From our numerical experience, for those problems for which the Bregman iterative algorithm greatly improves the recoverability, the Bregman iterative algorithm usually takes 2 to 3 iterations. Thus, in our numerical tests, we fixed the number of subproblems solved by our Bregman algorithm to 3. Since our Bregman algorithm achieves as good a relative error as the FPC algorithm, we only report how many of the examples that are successfully recovered by FPC, are improved greatly by using our Bregman iterative algorithm. In Table 3, NIM is the number of examples that the Bregman iterative algorithm outperformed FPC2 greatly (the relative errors obtained from FPC2 were at least 10^4 times larger than those obtained by the Bregman algorithm). From Table 3 we can see that for more than half of the examples successfully recovered by FPC2, the Bregman iterative algorithm improved the relative errors greatly (from $[1e-10, 1e-9]$ to $[1e-16, 1e-15]$). Of course the run times for the Bregman iterative algorithm were about three times that for algorithm FPC2, since the former calls the latter three times to solve the subproblems.

Table 3 Numerical results of Bregman iterative method for small matrix completion problems ($m=n=40$, $p=800$, $SR=0.5$)

Problem		NIM (NS)	FPC2		Bregman	
r	FR		RU	RL	RU	RL
1	0.0988	32 (50)	2.22e-9	2.55e-10	1.87e-15	3.35e-16
2	0.1950	29 (42)	5.01e-9	2.80e-10	2.96e-15	6.83e-16
3	0.2888	24 (35)	2.77e-9	3.88e-10	2.93e-15	1.00e-15
4	0.3800	10 (22)	5.51e-9	4.87e-10	3.11e-15	1.30e-15

In the following, we discuss the numerical results obtained by our approximate SVD based FPC algorithm (FPCA). We will see from these numerical results that FPCA achieves much better recoverability and is much faster than any of the solvers FPC1, FPC2, FPC3 or SDPT3.

We present the numerical results of FPCA for small ($m=n=40$), medium ($m=n=100$) and large ($m=n=1000$) problems in Tables 4, 5 and 6, respectively. Since we found that $xtol = 1e - 6$ is small enough to guarantee very good recoverability, we set $xtol = 1e - 6$ in algorithm FPCA and used only (4.2) as stopping rule for the inner iterations. From these tables, we can see that our FPCA algorithm is much more powerful than SDPT3 for randomly created matrix completion problems. When $m = n = 40$ and $p = 800$, and the rank r was less than or equal to 8, FPCA recovered the matrices in all 50 examples. When rank $r = 9$, it failed on only one example. Even for rank $r = 10$, which is almost the largest rank that satisfies $FR \leq 1$, FPCA still recovered the solution in more than 60% of the examples. However, SDPT3 started to fail to recover the matrices when the rank $r = 2$. When $r = 6$, there was only one example out of 50 where the correct solution matrix was recovered. When $r \geq 7$, none of the 50 examples could be recovered. For the medium sized matrices ($m = n = 100$) we used $p = 2000$, which is only a 20% measurement rate, FPCA recovered the matrices in all 50 examples when $r \leq 4$. For $r = 5, 6, 7$, FPCA recovered the matrices in most of the examples (more than nearly 90%). When $r = 8$, more than 60% of the matrices were recovered successfully by FPCA. Even when $r = 9$, FPCA still recovered 2 matrices. However, SDPT3 could not recover all of the matrices even when the rank $r = 1$ and none of the matrices were recovered when $r \geq 4$. When we increased the number of measurements to 3000, we recovered the matrices in all 50 examples up to rank $r = 12$. When $r = 13, 14$, we still recovered most of them. However, SDPT3 started to fail for some matrices when $r = 3$. When $r \geq 8$, SDPT3 failed to recover any of the matrices. We can also see that for the medium sized problems, FPCA was much faster than SDPT3. For the large problems ($m = n = 1000$), we used a sample ratio of 0.2 (i.e., we sampled 20% of the elements of the matrices). Also, for each rank r , we created only 10 examples to save time. We can see from Table 6 that FPCA took less than 27 minutes to recover the matrix in each of the examples in which the rank r was between 50 and 60. The relative errors were consistently of order $1e-6$. With 20% measurements, FPCA can recover even larger rank matrices. We do not report these numerical results here to conserve space. In contrast, SDPT3 can barely solve a matrix completion problem of size 1000×1000 .

7.2 Results for real problems

In this section, we consider matrix completion problems based on two real data sets: the Jester joke data set [20] and the DNA data set [32]. The Jester joke data set contains 4.1 million ratings for 100 jokes from 73,421 users and is available on the website <http://www.ieor.berkeley.edu/~Egoldberg/jester-data/>. Since the number of jokes is only 100, but the number of users is quite large, we randomly selected n_u users to get a modestly sized matrix for testing purpose. As in [34], we randomly held out two ratings for each user. Since some entries in the matrix are missing, we cannot compute the relative error as we did for the randomly created matrices. Instead, we computed

Table 4 Numerical results for FPCA and SDPT3 for randomly created small matrix completion problems ($m=n=40$, $p=800$, $SR=0.5$)

Problems		FPCA					SDPT3				
r	FR	NS	AT	RA	RU	RL	NS	AT	RA	RU	RL
1	0.0988	50	4.24	6.60e-7	2.26e-6	1.37e-7	50	1.84	6.30e-10	3.46e-9	8.70e-11
2	0.1950	50	4.35	1.08e-6	2.75e-6	3.68e-7	44	1.93	1.50e-9	7.18e-9	1.82e-10
3	0.2888	50	4.83	1.83e-6	6.47e-6	8.63e-7	37	1.99	2.66e-9	1.58e-8	3.10e-10
4	0.3800	50	4.92	2.56e-6	5.93e-6	1.19e-6	29	2.12	1.18e-8	7.03e-8	8.00e-10
5	0.4688	50	5.06	3.38e-6	1.28e-5	1.60e-6	8	2.30	1.83e-7	8.12e-7	2.60e-9
6	0.5550	50	5.48	3.72e-6	8.14e-6	1.77e-6	1	2.89	6.58e-7	6.58e-7	6.58e-7
7	0.6388	50	5.79	4.78e-6	9.92e-6	2.90e-6	0	—	—	—	—
8	0.7200	50	6.03	8.57e-6	4.48e-5	4.22e-6	0	—	—	—	—
9	0.7987	49	6.75	1.27e-5	7.64e-5	4.24e-6	0	—	—	—	—
10	0.8750	32	8.71	7.49e-5	6.34e-4	1.00e-5	0	—	—	—	—
11	0.9487	0	—	—	—	—	0	—	—	—	—

Table 5 Numerical results for FPCA and SDPT3 for randomly created medium matrix completion problems ($m=n=100$)

Problems				FPCA					SDPT3				
p	r	SR	FR	NS	AT	RA	RU	RL	NS	AT	RA	RU	RL
2000	1	0.2	0.0995	50	7.94	6.11e-6	2.13e-5	2.99e-6	47	15.10	1.55e-9	1.83e-8	1.40e-10
2000	2	0.2	0.1980	50	8.17	6.51e-6	1.78e-5	3.20e-6	31	16.02	7.95e-9	8.69e-8	5.20e-10
2000	3	0.2	0.2955	50	9.09	7.36e-6	1.48e-5	4.20e-6	13	19.23	1.05e-4	9.70e-4	9.08e-10
2000	4	0.2	0.3920	50	9.33	1.09e-5	5.14e-5	6.79e-6	0	—	—	—	—
2000	5	0.2	0.4875	49	9.91	2.99e-5	3.79e-4	8.4e-6	0	—	—	—	—
2000	6	0.2	0.5820	47	10.81	3.99e-5	2.92e-4	1.12e-5	0	—	—	—	—
2000	7	0.2	0.6755	44	12.63	8.87e-5	8.99e-4	1.26e-5	0	—	—	—	—
2000	8	0.2	0.7680	31	16.30	1.24e-4	5.49e-4	3.19e-5	0	—	—	—	—
2000	9	0.2	0.8595	2	17.88	6.19e-4	8.64e-4	3.74e-4	0	—	—	—	—
2000	10	0.2	0.9500	0	—	—	—	—	0	—	—	—	—
3000	1	0.3	0.0663	50	8.39	1.83e-6	4.34e-6	8.65e-7	50	36.68	2.01e-10	9.64e-10	7.52e-11
3000	2	0.3	0.1320	50	8.53	1.86e-6	4.40e-6	1.10e-6	50	36.50	1.13e-9	2.97e-9	1.77e-10
3000	3	0.3	0.1970	50	9.30	2.11e-6	3.19e-6	1.15e-6	46	38.50	1.28e-5	5.89e-4	2.10e-10
3000	4	0.3	0.2613	50	9.72	2.88e-6	5.41e-6	1.77e-6	42	41.28	4.60e-6	1.21e-4	4.53e-10
3000	5	0.3	0.3250	50	9.87	3.60e-6	6.90e-6	2.56e-6	32	43.92	7.82e-8	1.50e-6	1.23e-9
3000	6	0.3	0.3880	50	9.96	3.93e-6	5.93e-6	2.47e-6	17	49.60	3.44e-7	4.29e-6	3.68e-9
3000	7	0.3	0.4503	50	10.19	4.27e-6	7.30e-6	3.20e-6	3	59.18	1.43e-4	4.28e-4	1.57e-7
3000	8	0.3	0.5120	50	10.65	4.38e-6	7.55e-6	3.31e-6	0	—	—	—	—
3000	9	0.3	0.5730	50	11.74	5.01e-6	7.29e-6	3.99e-6	0	—	—	—	—
3000	10	0.3	0.6333	50	11.76	6.30e-6	1.08e-5	4.59e-6	0	—	—	—	—
3000	11	0.3	0.6930	50	12.08	8.29e-6	2.65e-5	5.47e-6	0	—	—	—	—
3000	12	0.3	0.7520	50	13.67	2.64e-5	4.27e-4	6.10e-6	0	—	—	—	—
3000	13	0.3	0.8103	48	16.00	2.95e-5	2.36e-4	9.50e-6	0	—	—	—	—
3000	14	0.3	0.8680	40	20.51	1.35e-4	9.36e-4	1.24e-5	0	—	—	—	—
3000	15	0.3	0.9250	0	—	—	—	—	0	—	—	—	—
3000	16	0.3	0.9813	0	—	—	—	—	0	—	—	—	—

the Normalized Mean Absolute Error (NMAE) as in [20] and [34]. The Mean Absolute Error (MAE) is defined as

$$MAE = \frac{1}{2N} \sum_{i=1}^N |\hat{r}_{i_1}^i - r_{i_1}^i| + |\hat{r}_{i_2}^i - r_{i_2}^i|, \quad (7.1)$$

Table 6 Numerical results of FPCA for randomly created large matrix completion problems ($m=n=1000$, $p=2e+5$, $SR=0.2$)

r	FR	NS	AT	RA	RU	RL
50	0.4875	10	1500.7	2.73e-6	2.84e-6	2.61e-6
51	0.4970	10	1510.2	2.75e-6	2.80e-6	2.69e-6
52	0.5065	10	1515.0	2.80e-6	2.88e-6	2.68e-6
53	0.5160	10	1520.6	2.79e-6	2.92e-6	2.69e-6
54	0.5254	10	1535.9	2.77e-6	2.88e-6	2.66e-6
55	0.5349	10	1543.6	2.80e-6	2.98e-6	2.72e-6
56	0.5443	10	1556.3	2.78e-6	2.96e-6	2.60e-6
57	0.5538	10	1567.3	2.74e-6	2.86e-6	2.57e-6
58	0.5632	10	1586.4	2.69e-6	2.79e-6	2.60e-6
59	0.5726	10	1576.1	2.66e-6	2.80e-6	2.52e-6
60	0.5820	10	1602.0	2.55e-6	2.64e-6	2.49e-6

where r_j^i and \hat{r}_j^i are the withheld and predicted ratings of movie j by user i , respectively, for $j = i_1, i_2$. NMAE is defined as

$$NMAE = \frac{MAE}{r_{\max} - r_{\min}}, \quad (7.2)$$

where r_{\min} and r_{\max} are lower and upper bounds for the ratings. Since all ratings are scaled to the range $[-10, +10]$, we have $r_{\min} = -10$ and $r_{\max} = 10$.

The numerical results for the Jester data set using FPC1 and FPCA are presented in Tables 7 and 8, respectively. In these two tables, σ_{\max} and σ_{\min} are the largest and smallest positive singular values of the recovered matrices, and $rank$ is the rank of the recovered matrices. The distributions of the singular values of the recovered matrices are shown in Figures 1 and 2. From Tables 7 and 8 we can see that by using FPC1 and FPCA to recover these matrices, we can get relatively low NMAEs, which are comparable to the results shown in [34] and [20].

Table 7 Numerical results of FPC1 for Jester joke data set

num.user	num.samp	samp.ratio	rank	σ_{\max}	σ_{\min}	NMAE	Time
100	7172	0.7172	79	285.6520	3.4916e-004	0.1727	34.3
1000	71152	0.7115	100	786.3651	38.4326	0.1667	304.8125
2000	140691	0.7035	100	1.1242e+003	65.0607	0.1582	661.6563

Table 8 Numerical results of FPCA for Jester joke data set (c_s is the number of rows we picked for approximated SVD)

num.user	num.samp	samp.ratio	ϵ_{k_s}	c_s	rank	σ_{\max}	σ_{\min}	NMAE	Time
100	7172	0.7172	1e-2	25	20	295.1449	32.6798	0.1627	26.7344
1000	71152	0.7115	1e-2	100	85	859.2710	48.0393	0.2008	808.5156
1000	71152	0.7115	1e-4	100	90	859.4588	44.6220	0.2101	778.5625
2000	140691	0.7035	1e-4	200	100	1.1518e+003	63.5244	0.1564	1.1345e+003

We also used two data sets of DNA microarrays from [32]. These data sets are available on the website <http://cellcycle-www.stanford.edu/>. The first microarray data set is a matrix that represents the expression of 6178 genes in 14 experiments based on the Elutriation data set in [32]. The second microarray data set is based on the Cdc15 data set in [32], and represents the expression of 6178 genes in 24 experiments. However, some entries in these two matrices

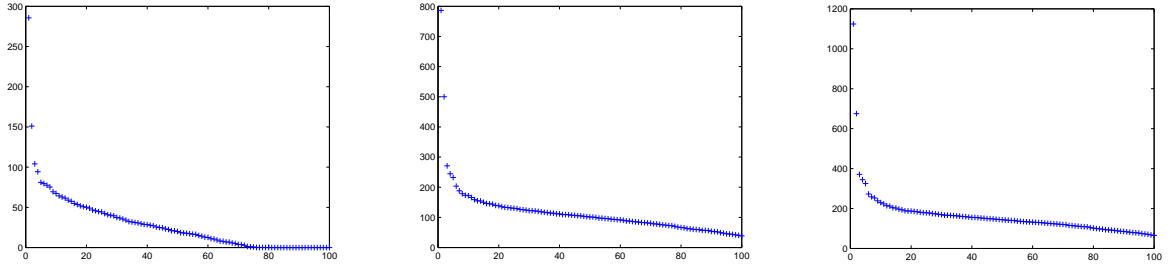


Fig. 1 Distribution of singular values of recovered matrices for Jester data set using FPC1. Left: 100 users, Middle: 1000 users, Right: 2000 users

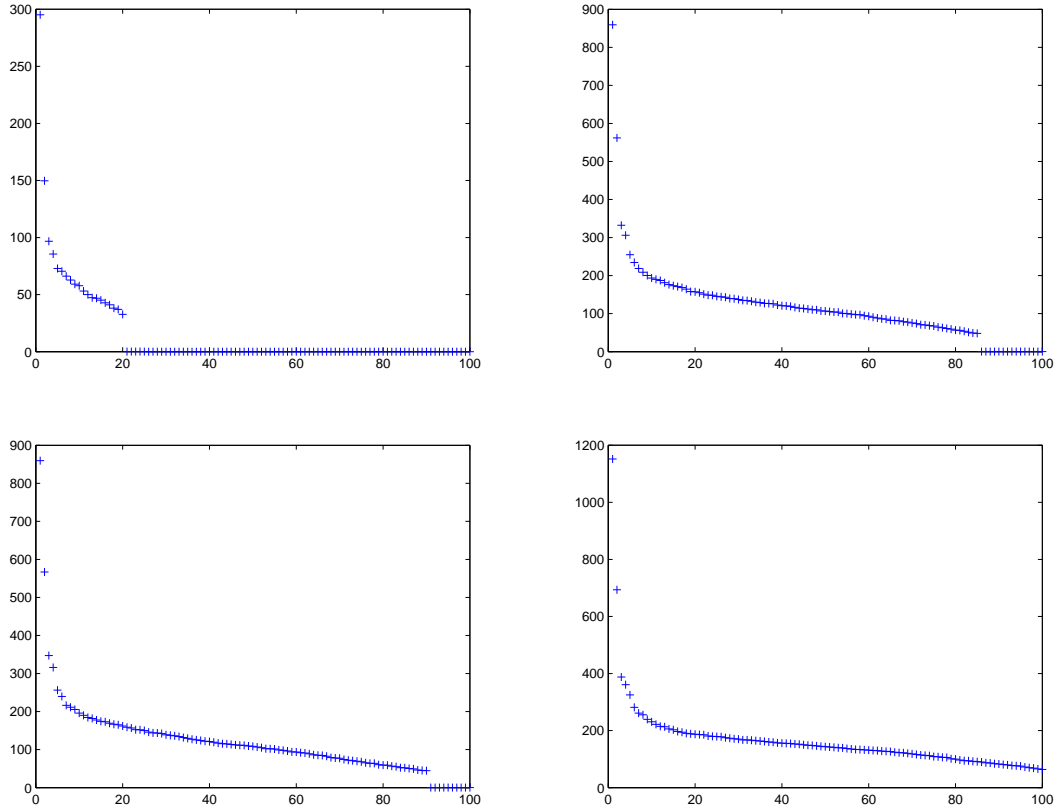


Fig. 2 Distribution of singular values of recovered matrices for Jester data set using FPCA. Upper Left: 100 users, $\epsilon_{k_s} = 1e-2, c_s = 25$; Upper Right: 1000 users, $\epsilon_{k_s} = 1e-2, c_s = 100$; Bottom Left: 1000 users, $\epsilon_{k_s} = 1e-4, c_s = 100$; Bottom Right: 2000 users, $\epsilon_{k_s} = 1e-4, c_s = 200$

are missing. For evaluating our algorithms, we created complete matrices by deleted all rows containing missing values. This is similar to how the DNA microarray data set was preprocessed in [38]. The resulting complete matrix for the Elutriation data set was 5766×14 . The complete matrix for the Cdc15 data set was 4381×24 . We must point out that these DNA microarray matrices are neither low-rank nor even approximately low-rank although such claims have been made in some papers. The distributions of the singular values of these two matrices are shown in Figure 3. From this figure we can see that in each microarray matrix, only one singular value is close to zero, while the others are far away from zero. Thus there is no way to claim that the rank of the Elutriation matrix is less than 13, or the rank of the Cdc15 matrix is less than 23. Since these matrices are not low-rank, we cannot expect our algorithms

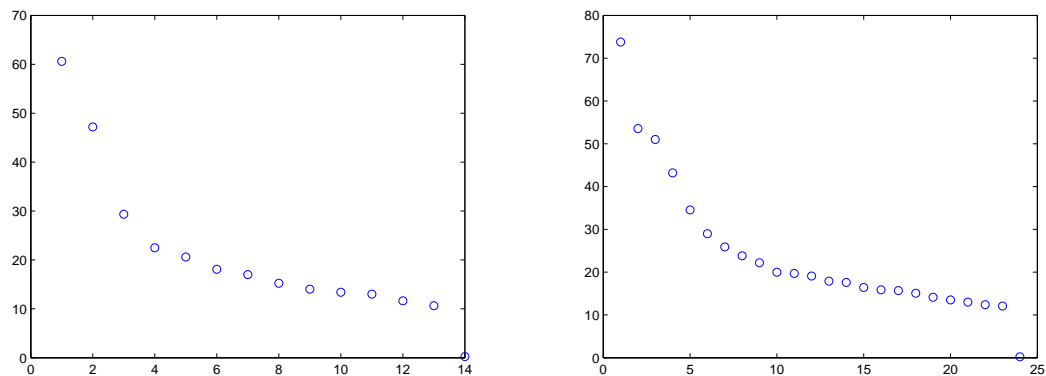


Fig. 3 Distribution of singular values of the original DNA microarray data sets. Left: Elutriation matrix; Right: Cdc15 matrix.

to recover these matrices by sampling only a small portion of their entries. Thus we needed to further modify the data sets to yield low-rank matrices. Specifically, we used the best rank-2 approximation to the Elutriation matrix as the new complete Elutriation matrix and the best rank-5 approximation to the Cdc15 matrix as the new complete Cdc15 matrix. The numerical results for FPCA for recovering these two matrices are presented in Table 9. In the FPCA algorithm, we set $\epsilon_{k_s} = 10^{-2}$ and $xtol = 10^{-6}$. For the Elutriation matrix, we set $c_s = 115$ and for the Cdc15 matrix, we set $c_s = 88$. The observed entries were randomly sampled. From Table 9 we can see that by taking 60% of the entries of the matrices, our FPCA algorithm can recover these matrices very well, yielding relative errors as low as $1e-5$ and $1e-6$, which is promising for practical use.

Table 9 Numerical results of FPCA for DNA microarray data sets

Matrix	m	n	p	rank	SR	FR	rel.err	Time
Elutriation	5766	14	48434	2	0.6	0.2386	$1.83e-5$	218.01
Cdc15	4381	24	63086	5	0.6	0.3487	$7.95e-6$	189.32

8 Conclusions and discussions

In this paper, we derived a fixed point continuation algorithm and a Bregman iterative algorithm for solving the linearly constrained nuclear norm minimization problem, which is a convex relaxation of the NP-hard linearly constrained matrix rank minimization problem. The convergence of the fixed point iterative scheme was established. By adopting an approximate SVD technique, we obtained a very powerful algorithm for the matrix rank minimization problem. This algorithm greatly outperforms SDP solvers such as SDPT3 in both speed and recoverability of low-rank matrices. FPC can be further accelerated if we incorporate line search and BB step techniques. Variants of Bregman iterative regularization such as Linearized Bregman iterations [7] and the Split Bregman algorithm [21] also deserve further investigation. We will discuss these issues in a subsequent paper.

References

1. Bach, F.R.: Consistency of trace norm minimization. *Journal of Machine Learning Research* **9**(Jun), 1019–1048 (2008)

2. van den Berg, E., Friedlander, M.P.: Probing the Pareto frontier for basis pursuit solutions. Preprint available at Optimization Online: 2008.01.1889 (2008)
3. Borwein, J.M., Lewis, A.S.: Convex Analysis and Nonlinear Optimization. Springer-Verlag (2003)
4. Bregman, L.: The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics* **7**, 200–217 (1967)
5. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (Series B)* **95**, 329–357 (2003)
6. Burer, S., Monteiro, R.D.C.: Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming* **103**(3), 427–444 (2005)
7. Cai, J., Osher, S., Shen, Z.: Linearized Bregman iterations for compressed sensing. Tech. rep., UCLA (2008)
8. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory* **52**, 489–509 (2006)
9. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Submitted (2008)
10. Candès, E.J., Romberg, J.: ℓ_1 -MAGIC: Recovery of sparse signals via convex programming. Tech. rep., Caltech (2005)
11. Dai, W., Milenkovic, O.: Subspace pursuit for compressive sensing: closing the gap between performance and complexity. Preprint available at arXiv: 0803.0811 (2008)
12. Donoho, D.: Compressed sensing. *IEEE Transactions on Information Theory* **52**, 1289–1306 (2006)
13. Donoho, D., Tsai, Y., Drori, I., Starck, J.C.: Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Submitted to *IEEE Transactions on Information Theory* (2006)
14. Donoho, D.L., Tsai, Y.: Fast solution of ℓ_1 -norm minimization problems when the solution may be sparse. Tech. rep., Department of Statistics, Stanford University (2006)
15. Drineas, P., Kannan, R., Mahoney, M.W.: Fast Monte Carlo algorithms for matrices ii: Computing low-rank approximations to a matrix. *SIAM J. Computing* **36**, 132–157 (2006)
16. Fazel, M.: Matrix rank minimization with applications. Ph.D. thesis, Stanford University (2002)
17. Fazel, M., Hindi, H., Boyd, S.: A rank minimization heuristic with application to minimum order system approximation. In: *Proceedings of the American Control Conference* (2001)
18. Figueiredo, M.A.T., Nowak, R.D., Wright, S.J.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal on Selected Topics in Signal Processing* **1**(4) (2007)
19. Ghaoui, L.E., Gahinet, P.: Rank minimization under LMI constraints: A framework for output feedback problems. In: *Proceedings of the European Control Conference* (1993)
20. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval* **4**(2), 133–151 (2001)
21. Goldstein, T., Osher, S.: The split Bregman algorithm for ℓ_1 regularized problems. Tech. rep., UCLA (2008)
22. Hale, E.T., Yin, W., Zhang, Y.: A fixed-point continuation method for ℓ_1 -regularized minimization with applications to compressed sensing. Tech. rep., CAAM TR07-07 (2007)
23. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms II: Advanced Theory and Bundle Methods. Springer-Verlag, New York (1993)
24. Kim, S.J., Koh, K., Lustig, M., Boyd, S., Gorinevsky, D.: A method for large-scale ℓ_1 -regularized least-squares. *IEEE Journal on Selected Topics in Signal Processing* **4**(1), 606–617 (2007)
25. Linial, N., London, E., Rabinovich, Y.: The geometry of graphs and some of its algorithmic applications. *Combinatorica* **15**, 215–245 (1995)
26. Liu, Z., Vandenberghe, L.: Interior-point method for nuclear norm approximation with application to system identification. Submitted to *Mathematical Programming Series B* (2008)

27. Natarajan, B.K.: Sparse approximation solutions to linear systems. *SIAM J. Computing* **24**(2), 227–234 (1995)
28. Osher, S., Burger, M., Goldfarb, D., Xu, J., Yin, W.: An iterative regularization method for total variation-based image restoration. *SIAM MMS* **4**(2), 460–489 (2005)
29. Recht, B., Fazel, M., Parrilo, P.: Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. Submitted to *SIAM Review* (2007)
30. Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: *Proceedings of the International Conference of Machine Learning* (2005)
31. Rudin, L., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. *Physica D* **60**, 259–268 (1992)
32. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell* **9**, 3273–3297 (1998)
33. Srebro, N.: Learning with matrix factorizations. Ph.D. thesis, Massachusetts Institute of Technology (2004)
34. Srebro, N., Jaakkola, T.: Weighted low-rank approximations. In: *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)* (2003)
35. Sturm, J.F.: Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software* **11-12**, 625–653 (1999)
36. Tibshirani, R.: Regression shrinkage and selection via the lasso. *Journal Royal Statistical Society B* **58**, 267–288 (1996)
37. Tropp, J.: Just relax: Convex programming methods for identifying sparse signals. *IEEE Transactions on Information Theory* **51**, 1030–1051 (2006)
38. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for DNA microarrays. *Bioinformatics* **17**(6), 520–525 (2001)
39. Tütüncü, R.H., Toh, K.C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming Series B* **95**, 189–217 (2003)
40. Wen, Z., Yin, W., Goldfarb, D.: An algorithm for ℓ_1 minimization using shrinkage and subspace optimization. Tech. rep., Department of IEOR, Columbia University (2008)
41. Yin, W., Osher, S., Goldfarb, D., Darbon, J.: Bregman iterative algorithms for ℓ_1 -minimization with applications to compressed sensing. *SIAM J. Imaging Sci* **1**(1), 143–168 (2008)