Math. Prog. Comp. (2012) 4:333–361
DOI 10.1007/s12532-012-0044-1

FULL LENGTH PAPER

Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm

Zaiwen Wen · Wotao Yin · Yin Zhang

Received: 14 October 2011 / Accepted: 25 June 2012 / Published online: 13 July 2012

Abstract The matrix completion problem is to recover a low-rank matrix from a subset of its entries. The main solution strategy for this problem has been based on nuclear-norm minimization which requires computing singular value decompositions—a task that is increasingly costly as matrix sizes and ranks increase. To improve the capacity of solving large-scale problems, we propose a low-rank factorization model and construct a nonlinear successive over-relaxation (SOR) algorithm that only requires solving a linear least squares problem per iteration. Extensive numerical experiments show that the algorithm can reliably solve a wide range of problems at a speed at least several times faster than many nuclear-norm minimization algorithms. In addition, convergence of this nonlinear SOR algorithm to a stationary point is analyzed.

Z. Wen's research was supported in part by NSF DMS-0439872 through UCLA IPAM and the NSFC grant 11101274.
W. Yin's research was supported in part by NSF CAREER Award DMS-07-48839, ONR Grant N00014-08-1-1101, and an Alfred P. Sloan Research Fellowship.
Y. Zhang's research was supported in part by NSF grants DMS-0405831 and DMS-0811188 and ONR grant N00014-08-1-1101.

Z. Wen (✉)
Department of Mathematics and Institute of Natural Sciences,
Shanghai Jiaotong University, Shanghai, China
e-mail: zw2109@sjtu.edu.cn

W. Yin · Y. Zhang
Department of Computational and Applied Mathematics,
Rice University, Houston, TX 77005, USA
e-mail: wotao.yin@rice.edu

Y. Zhang
e-mail: yzhang@rice.edu

Keywords Matrix completion · Alternating minimization · Nonlinear GS method · Nonlinear SOR method

Mathematics Subject Classification 65K05 · 90C06 · 93C41 · 68Q32

1 Introduction

The problem of minimizing the rank of a matrix arises in many applications, for example, control and systems theory, model reduction and minimum order control synthesis [22], recovering shape and motion from image streams [27,34], data mining and pattern recognitions [8] and machine learning such as latent semantic indexing, collaborative prediction and low-dimensional embedding. In this paper, we consider the Matrix Completion (MC) problem of finding a lowest-rank matrix given a subset of its entries, that is,

$$\min_{W \in \mathbb{R}^{m\times n}} \ \operatorname{rank}(W), \ \text{s.t.}\ W_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega, \tag{1.1}$$

where $\operatorname{rank}(W)$ denotes the rank of W, and $M_{i,j} \in \mathbb{R}$ are given for $(i,j) \in \Omega \subset \{(i,j) : 1 \le i \le m, 1 \le j \le n\}$. Although problem (1.1) is generally NP-hard due to the combinational nature of the function $\operatorname{rank}(\cdot)$, it has been shown in [3,4,30] that, under some reasonable conditions, the solution of problem (1.1) can be found by solving a convex optimization problem:

$$\min_{W \in \mathbb{R}^{m\times n}} \ \|W\|_*, \ \text{s.t.}\ W_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega, \tag{1.2}$$

where the *nuclear* or *trace* norm $\|W\|_*$ is the summation of the singular values of W. In particular, Candès and Recht in [3] proved that a given rank-r matrix M satisfying certain incoherence conditions can be recovered exactly by (1.2) with high probability from a subset Ω of uniformly sampled entries whose cardinality $|\Omega|$ is of the order $O(r(m+n)\operatorname{polylog}(m+n))$. For more refined theoretical results on matrix completion we refer the reader to [2,4,13,16,17,29,42].

Various types of algorithms have been proposed to recover the solution of (1.1) based on solving (1.2). One method is the singular value thresholding algorithm [15] using soft-thresholding operations on the singular values of a certain matrix at each iteration. Another approach is the fixed-point shrinkage algorithm [24] which solves the regularized linear least problem:

$$\min_{W \in \mathbb{R}^{m\times n}} \ \mu\|W\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(W - M)\|_F^2, \tag{1.3}$$

where $\mathcal{P}_\Omega$ is the projection onto the subspace of sparse matrices with nonzeros restricted to the index subset Ω. An accelerated proximal gradient algorithm is developed in [33] based on a fast iterative shrinkage-thresholding algorithm [1] for compressive sensing. The classical alternating direction augmented Lagrangian methods have been applied to solve (1.2) in [10,38] and the closely related sparse and low-rank matrix decomposition in [39]. Other approaches include [7,18,20,21,25,26]. All of these algorithms bear the computational cost required by singular value decompositions (SVD) which becomes increasingly costly as the sizes of the underlying matrices increase.

It is evident that computing a full SVD at every iteration is too costly to be practical for solving truly large-scale problems. In some recent implementations such as [33], a partial SVD strategy has been deployed where one only computes a proper subset of dominant singular pairs (values and vectors) instead of the full set. This strategy, capable of solving large scale problems, requires an estimated upper bound on the rank of the solution at each iteration. Even so, the cost of computing a partial SVD can still be quite high on a wide range of large matrices. It is therefore desirable to exploit an alternative approach that avoids SVD computation all together, by replacing it with some less expensive computation. The purpose of this paper is to investigate such a non-SVD approach in order to more efficiently solve large-scale matrix completion problems.

It is well known that any matrix $W \in \mathbb{R}^{m\times n}$ of a rank up to K can be written into a matrix product form $W = XY$ where $X \in \mathbb{R}^{m\times K}$ and $Y \in \mathbb{R}^{K\times n}$. Now we propose to solve the following low-rank factorization model

$$\min_{X,Y,Z} \frac{1}{2}\|XY - Z\|_F^2 \ \text{ s.t. } \ Z_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega, \tag{1.4}$$

where $X \in \mathbb{R}^{m\times K}$, $Y \in \mathbb{R}^{K\times n}$, $Z \in \mathbb{R}^{m\times n}$, and the integer K should ideally be equal to the rank of the the data matrix M. Since the correct rank is generally unknown in advance, we will dynamically estimate and adjust the value of K during iterations, just as in a partial SVD strategy for solving model (1.2). The variable Z is introduced for a computational purpose that should become clear later. The premise of introducing the low-rank factorization model (1.4) is that it can generally be solved much faster than model (1.2). More specifically, the main computation of solving model (1.4) at each iteration is a rank-revealing or regular QR factorization [5,6] instead of a full or partial SVD. As a result, our rank estimation is based on rank-revealing QR factorizations instead of on singular value decompositions as in a partial SVD strategy. Both heuristics are effective tools in estimating the correct rank, but on many matrices QR factorizations are several times less expensive than corresponding partial SVDs.

There does exist a potential drawback of the low-rank factorization model (1.4), that is, the non-convexity in the model may prevent one from getting a global solution. There is also a secondary concern that the approach requires an initial rank estimate K. In this paper, we present convincing evidence to show that (a) on a wide range of problems tested, the low-rank factorization model (1.4) is empirically as reliable as the nuclear norm minimization model (1.2); and (b) the initial rank estimate need not be close to the exact rank r of M; for example, one could start with $K = \min(m, n)$ (though a better initial estimate would be computationally benefitial). We also allow a strategy of starting from $K = 1$ and gradually increasing K. We observe that the global optimal value of (1.4) is monotonically non-increasing with respect to K. In principle, if K is smaller than the unknown rank r, the quality of the solution in terms of the objective function value can be improved by minimizing

(1.4) again, starting from the current point, with an appropriately increased rank estimate.

A recent work in [16,18] is also based on a low-rank factorization model closely related to (1.4) where the factorization is in the form of USV^T where U and V have orthonormal columns. The authors derived a theoretical guarantee of recovery with high probability for their approach that consists of three steps. The first step is called trimming that removes from the sample $\mathcal{P}_\Omega(M)$ "over-represented" rows or columns. The second step finds the best rank-r approximation matrix to the remaining sample matrix via singular value decomposition (SVD) where r is the true rank and assumed to be known. In the final step, starting from the computed SVD factor as an initial guess, they solve the factorization model via a special gradient descent method that keeps the variables U and V orthonormal. The key intuition for their theoretical result is that the initial guess is so good that it falls into a certain neighborhood of the global minimum where there exists no other stationary point with high probability. This enables the authors to prove that their gradient descent method generates a sequence residing within this small neighborhood and converging to the global solution in the limit, despite the non-convexity of the factorization model. Given that our factorization model (1.4) is essentially the same as theirs, our approach should be able to benefit from the same initial point and possibly attain a similar theoretical guarantee. However, the proofs in [16] are specially tailored to the particularities of their algorithm and do not apply to our algorithm presented in this paper. Extending a similar theoretical result to our case is a topic of interest for future research. Meanwhile, the present paper concentrates on algorithm construction, convergence (to stationary point) analysis and performance evaluations. A low-rank factorization method based on the augmented Lagrangian framework is proposed in [30] for an equivalent quadratic formulation of the model (1.2). However, this method is only conceptual and the authors used SeDuMi to solve the SDP formulation of (1.2) in their numerical experiments.

Our main contribution is the development of an efficient algorithm for (1.4) that can reliably solve a wide range of matrix completion and approximation problems at a speed much faster than the best of existing nuclear norm minimization algorithms. Like in many other similar cases, the structure of (1.4) suggests an alternating minimization scheme as a natural choice. In this case, one can update each of the variables X, Y or Z efficiently while fixing the other two. The subproblems with respect to either the variable X or Y are linear least squares problems only involving $K \times K$ coefficient matrices in their normal equations, and the solution of the subproblem for Z can also be carried out efficiently. This alternating minimization procedure can also be viewed as a nonlinear (block) Gauss–Seidel (GS) scheme or a block coordinate descent method. In this paper, we propose a more sophisticated nonlinear successive over-relaxation (SOR) scheme with a strategy to adjust the relaxation weight dynamically. Numerical experiments show that this SOR scheme is significantly faster than the straightforward nonlinear GS scheme. The convergence of nonlinear GS (coordinate descent) methods for several optimization problems has been studied, for example, in [12,23,35,36]. However, we are unaware of any general convergence result for nonlinear SOR methods on non-convex optimization that is directly applicable to our nonlinear SOR algorithm. In this paper, we proved that our approach converges to a stationary point under a very mild assumption.

The rest of this paper is organized as follows. We first present an alternating minimization scheme for (1.4) in Sect. 2.1 with two efficient implementation variants. Our nonlinear SOR algorithm is introduced in Sect. 2.2. An convergence analysis for the nonlinear SOR algorithm is given in Sect. 3. Finally, two strategies for adjusting the rank estimate K and numerical results are presented in Sect. 4 to demonstrate the robustness and efficiency of our algorithm.

2 Alternating minimization schemes

2.1 Nonlinear Gauss–Seidel method

We start with a straightforward alternating minimization scheme for solving problem (1.4). Although alternating minimization is a common strategy widely used in many other similar situations, there is a subtlety in this case regarding efficiency. Given the current iterates X, Y and Z, the algorithm updates these three variables by minimizing (1.4) with respect to each one separately while fixing the other two. For example, by fixing the values of Y and Z, we obtain the new point X_+:

$$X_+ = ZY^\dagger = \operatorname*{argmin}_{X\in\mathbb{R}^{m\times K}} \frac{1}{2}\|XY - Z\|_F^2,$$

where $A^\dagger$ is the Moore–Penrose pseudo-inverse of A. Similarly, we can update Y and then Z, while fixing others at their latest available values. This procedure yields the following iterative scheme:

$$X_+ \leftarrow ZY^\dagger \equiv ZY^\top(YY^\top)^\dagger, \tag{2.1a}$$
$$Y_+ \leftarrow (X_+)^\dagger Z \equiv (X_+^\top X_+)^\dagger(X_+^\top Z), \tag{2.1b}$$
$$Z_+ \leftarrow X_+Y_+ + \mathcal{P}_\Omega(M - X_+Y_+). \tag{2.1c}$$

It follows from (2.1a) and (2.1b) that

$$X_+Y_+ = \left(X_+(X_+^\top X_+)^\dagger X_+^\top\right) Z = \mathcal{P}_{X_+}Z,$$

where $\mathcal{P}_A := A(A^\top A)^\dagger A^\top = QQ^\top$ is the orthogonal projection onto the range space $\mathcal{R}(A)$ of A and $Q := \texttt{orth}(A)$ is an orthonormal basis for $\mathcal{R}(A)$. The pseudo-inverse of A, the orthonormal basis of $\mathcal{R}(A)$ and the orthogonal projection onto $\mathcal{R}(A)$ can be computed from either the SVD or the QR factorization of A. One can verify that $\mathcal{R}(X_+) = \mathcal{R}(ZY^\top)$. Indeed, let $Y = U\Sigma V^\top$ be the economy-form SVD of Y, then $X_+ = ZV\Sigma^\dagger U^\top$ and $ZY^\top = ZV\Sigma U^\top$, implying that $\mathcal{R}(X_+) = \mathcal{R}(ZY^\top) = \mathcal{R}(ZV)$ and leading to the following lemma.

Lemma 2.1 *Let* (X_+, Y_+) *be generated by* (2.1)*.There holds*

$$X_+Y_+ = \mathcal{P}_{ZY^\top}Z = ZY^\top(YZ^\top ZY^\top)^\dagger(YZ^\top)Z. \tag{2.2}$$

We next present two iterative schemes equivalent to (2.1). Since the objective function (1.4) is determined by the product X_+Y_+, different values of X_+ and Y_+ are essentially equivalent as long as they give the same product X_+Y_+. Lemma 2.1 shows that the inversion $(YY^\top)^\dagger$ can be saved when the projection $\mathcal{P}_{ZY^\top}$ is computed. The unique feature of our new schemes is that only one least square problem is involved at each iteration. The first variant is to replace the step (2.1a) by

$$X_+ \leftarrow ZY^\top, \tag{2.3a}$$

while Y_+ and Z_+ are still generated by step (2.1b) and (2.1c). The second variant computes the orthogonal projection $\mathcal{P}_{ZY^\top} = VV^\top$, where $V := \mathtt{orth}(ZY^\top)$ is an orthogonal basis of $\mathcal{R}(ZY^\top)$. Hence, (2.2) can be rewritten as $X_+Y_+ = VV^\top Z$ and one can derive:

$$X_+ \leftarrow V, \tag{2.4a}$$
$$Y_+ \leftarrow V^\top Z, \tag{2.4b}$$

while Z_+ is still generated by step (2.1c). The scheme (2.4) is often preferred since computing the step (2.4b) by QR factorization is generally more stable than solving the normal equations. Note that the schemes (2.1), (2.3) and (2.4) can be used interchangeably in deriving properties of the product X_+Y_+.

By introducing a Lagrange multiplier $\Lambda \in \mathbb{R}^{m\times n}$ so that $\Lambda = \mathcal{P}_\Omega(\Lambda)$, the Lagrangian function of (1.4) is defined as

$$\mathcal{L}(X, Y, Z, \Lambda) = \frac{1}{2}\|XY - Z\|_F^2 - \Lambda \bullet \mathcal{P}_\Omega(Z - M), \tag{2.5}$$

where the inner product between two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times n}$ is defined as $A \bullet B := \sum_{ij} A_{ij}B_{ij}$. Differentiating the Lagrangian function $\mathcal{L}(X, Y, Z, \Lambda)$, we have the first-order optimality conditions for (1.4):

$$(XY - Z)Y^\top = 0, \tag{2.6a}$$
$$X^\top(XY - Z) = 0, \tag{2.6b}$$
$$\mathcal{P}_{\Omega^c}(Z - XY) = 0, \tag{2.6c}$$
$$\mathcal{P}_\Omega(Z - M) = 0, \tag{2.6d}$$

plus the equations

$$\mathcal{P}_\Omega(Z - XY) = \Lambda. \tag{2.7}$$

Clearly, the multiplier matrix Λ measures the residual $Z - XY$ in Ω and has no effect in the process of determining X, Y, Z. It is also easy to see that the above alternating minimization schemes are exactly a Gauss–Seidel (GS) method applied to the nonlinear and square system (2.6).

2.2 A nonlinear SOR-like scheme

Numerical simulations shows that the simple approach in Sect. 2.1, though being very reliable, is not efficient on large yet very low-rank matrices. A possible acceleration technique may involve applying an extension of the classic augmented-Lagrangian-based alternating direction method (ADM) for convex optimization to the factorization model (see [31,37,41] for such ADM extensions). However, in this paper, we investigate a nonlinear Successive Over-Relaxation (SOR) approach that we found to be particularly effective for solving the matrix completion problem.

In numerical linear algebra, the SOR method [11] for solving a linear system of equations is devised by applying extrapolation to the GS method, that is, the new trial point is a weighted average between the previous iterate and the computed GS iterate successively for each component. A proper value of the weight often results in faster convergence. Applying the same idea to the basic schemes (2.1), (2.3) and (2.4) gives a nonlinear SOR scheme:

$$\begin{aligned} X_+ &\leftarrow ZY^\top(YY^\top)^\dagger, && (2.8a)\\ X_+(\omega) &\leftarrow \omega X_+ + (1-\omega)X, && (2.8b)\\ Y_+ &\leftarrow (X_+(\omega)^\top X_+(\omega))^\dagger(X_+(\omega)^\top Z), && (2.8c)\\ Y_+(\omega) &\leftarrow \omega Y_+ + (1-\omega)Y, && (2.8d)\\ Z_+(\omega) &\leftarrow X_+(\omega)Y_+(\omega) + \mathcal{P}_\Omega(M - X_+(\omega)Y_+(\omega)), && (2.8e) \end{aligned}$$

where the weight $\omega \geq 1$. Obviously, $\omega = 1$ gives the GS method.

Assuming that the matrix Y has full row rank, the two least squares problems in (2.8) can be reduced into one like the second basic scheme (2.3). Let us denote the residual by

$$S = \mathcal{P}_\Omega(M - XY), \tag{2.9}$$

which will be used to measure optimality. After each iteration, the variable Z, which is feasible, can be expressed as $Z = XY + S$. Let Z_ω be a weighted sum of the matrices XY and S, that is,

$$Z_\omega \triangleq XY + \omega S = \omega Z + (1-\omega)XY. \tag{2.10}$$

Using the fact that the matrix $YY^\top(YY^\top)^\dagger$ is the identity from our assumption, we obtain

$$\begin{aligned} Z_\omega Y^\top(YY^\top)^\dagger &= \omega ZY^\top(YY^\top)^\dagger + (1-\omega)XYY^\top(YY^\top)^\dagger\\ &= \omega X_+ + (1-\omega)X, \end{aligned}$$

which is exactly the step (2.8b). Replacing Z by Z_ω in (2.3) and (2.4), we have the following SOR-like scheme:

$$\begin{aligned}
Z_\omega &\leftarrow \omega Z + (1-\omega)XY, &\text{(2.11a)}\\
X_+(\omega) &\leftarrow Z_\omega Y^\top \text{ or } Z_\omega Y^\top (YY^\top)^\dagger, &\text{(2.11b)}\\
Y_+(\omega) &\leftarrow (X_+(\omega)^\top X_+(\omega))^\dagger (X_+(\omega)^\top Z_\omega), &\text{(2.11c)}\\
\mathcal{P}_{\Omega^c}(Z_+(\omega)) &\leftarrow \mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega)), &\text{(2.11d)}\\
\mathcal{P}_\Omega(Z_+(\omega)) &\leftarrow \mathcal{P}_\Omega(M). &\text{(2.11e)}
\end{aligned}$$

Again, an implementation with a single QR decomposition can be utilized just as in scheme (2.4).

Since a fixed weight ω is generally inefficient for nonlinear problems, we next present an updating strategy for ω that is similar to the one adjusting the trust-region radius in the trust region method [28] for nonlinear programming. After the point $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ is computed, we calculate the residual ratio

$$\gamma(\omega) = \frac{\|S_+(\omega)\|_F}{\|S\|_F}, \tag{2.12}$$

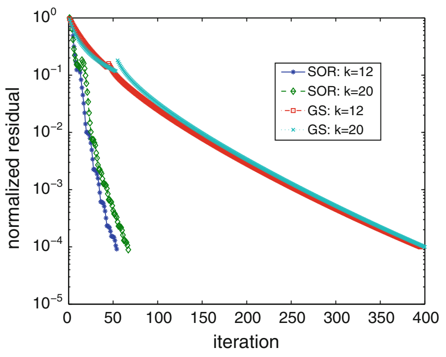
where

$$S_+(\omega) \triangleq \mathcal{P}_\Omega(M - X_+(\omega)Y_+(\omega)). \tag{2.13}$$

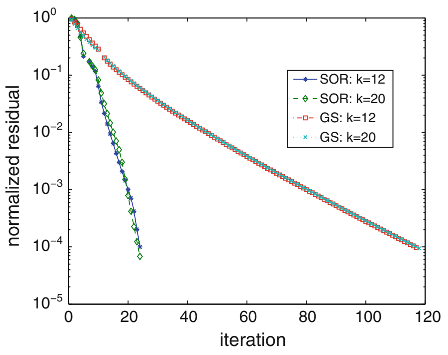
If $\gamma(\omega) < 1$, this new pair of point is accepted as the next iterate since our object to reduce the residual $\|S\|_F := \|\mathcal{P}_\Omega(M - XY)\|_F$ is achieved. In this case, the step is called "successful"; otherwise, the step is "unsuccessful" and we have to generate a new trial point using a new weight ω so that $\gamma(\omega) < 1$ is guaranteed. Since the basic GS method corresponds to $\omega = 1$ and it can reduce the residual $\|S\|_F$, we simply reset ω to 1 in a "unsuccessful" case. Once a trial point is acceptable, we consider whether the weight ω should be updated. As our goal is to minimize the residual $\|S\|$, a small $\gamma(\omega)$ indicates that the current weight value ω works well so far and keeping the current value will very likely continue to provide good progress. Hence, ω is increased only if the calculated point is acceptable but the residual ratio $\gamma(\omega)$ is considered "too large"; that is, $\gamma(\omega) \in [\gamma_1, 1)$ for some $\gamma_1 \in (0, 1)$. If this happens, we increase ω to $\min(\omega + \delta, \tilde{\omega})$, where $\delta > 0$ is an increment and $\tilde{\omega} > 1$ is an upper bound. From the above considerations, we arrive at Algorithm 1 below.

Algorithm 1: A low-rank matrix fitting algorithm (`LMaFit`)

```
1 Input index set $\Omega$, data $\mathcal{P}_\Omega(M)$ and a rank overestimate $K \ge r$.;
2 Set $Y^0 \in \mathbb{R}^{K\times n}$, $Z^0 = \mathcal{P}_\Omega(M)$, $\omega = 1$, $\tilde{\omega} > 1$, $\delta > 0$, $\gamma_1 \in (0,1)$ and $k = 0$.;
3 while not convergent do
4     Compute $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ according to (2.11) with $(X, Y, Z) = (X^k, Y^k, Z^k)$.;
5     Compute the residual ratio $\gamma(\omega)$ according to (2.12).;
6     if $\gamma(\omega) \ge 1$ then set $\omega = 1$ and go to step 4. ;
7     Update $(X^{k+1}, Y^{k+1}, Z^{k+1}) = (X_+(\omega), Y_+(\omega), Z_+(\omega))$ and increment $k$.;
8     if $\gamma(\omega) \ge \gamma_1$ then set $\delta = \max(\delta, 0.25(\omega - 1))$ and $\omega = \min(\omega + \delta, \tilde{\omega})$. ;
```



(a) n=1000, r=10, sampling ratio = 0.08



(b) n=1000, r=10, sampling ratio=0.15

Fig. 1 Comparison between the nonlinear GS and SOR schemes

For illustration, we compare the efficiency of the GS scheme (2.1) and the nonlinear SOR-like scheme (2.11) on two random matrices M with $m = n = 1{,}000$, $r = 10$ with two different sampling ratios at, respectively, 0.08 and 0.15 (see Sect. 4.2 for detailed construction procedure and algorithmic parameter setting). The algorithms were run by using two different rank estimations $K = 12$ and 20. The normalized residuals $\|\mathcal{P}_\Omega(M - XY)\|_F / \|\mathcal{P}_\Omega(M)\|_F$ are depicted in Fig. 1a and b, respectively. The apparent jumps in the residuals were due to adjustments of rank estimations, which will be explained later. From the figures, it is evident that the nonlinear SOR scheme is significantly faster than the nonlinear GS scheme.

3 Convergence analysis

We now analyze Algorithm 1 by revealing the relationships between the residuals $\|S\|_F$ and $\|S_+(\omega)\|_F$. Let $V(\omega) := \texttt{orth}(X_+(\omega))$ and $U := \texttt{orth}(Y^\top)$ be orthogonal bases of the range spaces of $\mathcal{R}(X_+(\omega))$ and $\mathcal{R}(Y^\top)$, respectively. Consequently, the orthogonal projections onto $\mathcal{R}(X_+(\omega))$ and $\mathcal{R}(Y^\top)$ can be expressed as:

$$\begin{aligned} Q(\omega) &:= V(\omega)V(\omega)^\top = X_+(\omega)(X_+(\omega)^\top X_+(\omega))^\dagger X_+(\omega)^\top, \\ P &:= UU^\top = Y^\top (YY^\top)^\dagger Y. \end{aligned}$$

We list several useful identities that can be verified from the definition of pseudo-inverse. For any $A \in \mathbb{R}^{m\times n}$,

$$\begin{aligned} A^\dagger &= A^\dagger (A^\dagger)^\top A^\top = A^\top (A^\dagger)^\top A^\dagger = (A^\top A)^\dagger A^\top = A^\top (AA^\top)^\dagger, \\ A &= (A^\dagger)^\top A^\top A = AA^\top (A^\dagger)^\top. \end{aligned} \tag{3.1}$$

The lemma below and its proof will provide us a key equality.

Lemma 3.1 *Let $(X_+(\omega), Y_+(\omega))$ be generated by (2.11) and S be defined in (2.9). There holds*

$$\omega S \bullet (X_+(\omega)Y_+(\omega) - XY) = \|X_+(\omega)Y_+(\omega) - XY\|_F^2. \tag{3.2}$$

Proof It follows from $Y^\top = Y^\dagger Y Y^\top$ (see (3.1)), $X_+(\omega) = Z_\omega Y^\dagger$ and $Z_\omega = XY + \omega S$ that

$$X_+(\omega)YY^\top = Z_\omega Y^\top = XYY^\top + \omega S Y^\top.$$

Post-multiplying both sides by $(YY^\top)^\dagger Y$ and rearranging, we have $(X_+(\omega) - X)Y = \omega S Y^\top (YY^\top)^\dagger Y$; i.e.,

$$(X_+(\omega) - X)Y = \omega SP. \tag{3.3}$$

On the other hand, the equalities $X_+(\omega)^\top = X_+(\omega)^\top X_+(\omega)(X_+(\omega))^\dagger$ and (3.3) yield

$$\begin{aligned} X_+(\omega)^\top X_+(\omega) Y_+(\omega) &= X_+(\omega)^\top Z_\omega = X_+(\omega)^\top (XY + \omega S) \\ &= X_+(\omega)^\top (X_+(\omega)Y - (X_+(\omega) - X)Y + \omega S) \\ &= X_+(\omega)^\top X_+(\omega) Y + \omega X_+(\omega)^\top S(I - P). \end{aligned}$$

Pre-multiplying both sided by $X_+(\omega)(X_+(\omega)^\top X_+(\omega))^\dagger$ and rearranging, we arrive at

$$X_+(\omega)(Y_+(\omega) - Y) = \omega Q(\omega) S (I - P). \tag{3.4}$$

Therefore, in view of (3.3) and (3.4), we obtain

$$\begin{aligned} X_+(\omega)Y_+(\omega) - XY &= (X_+(\omega) - X)Y + X_+(\omega)(Y_+(\omega) - Y) \\ &= \omega SP + \omega Q(\omega)S(I - P) \qquad (3.5) \\ &= \omega(I - Q(\omega))SP + \omega Q(\omega)S. \qquad (3.6) \end{aligned}$$

Therefore,

$$\|X_+(\omega)Y_+(\omega) - XY\|_F^2 = \omega^2 \|(I - Q(\omega))SP\|_F^2 + \omega^2 \|Q(\omega)S\|_F^2. \tag{3.7}$$

Finally, in view of (3.6) and the properties of orthogonal projections, we have:

$$\begin{aligned} \omega S \bullet (X_+(\omega)Y_+(\omega) - XY) &= \omega^2 S \bullet (I - Q(\omega))SP + \omega^2 S \bullet Q(\omega) S \\ &= \omega^2 SP \bullet (I - Q(\omega))SP + \omega^2 S \bullet Q(\omega) S \\ &= \omega^2 \|(I - Q(\omega))SP\|_F^2 + \omega^2 \|Q(\omega)S\|_F^2 \\ &= \|X_+(\omega)Y_+(\omega) - XY\|_F^2, \end{aligned}$$

which proves the lemma. □

It is easy to see that

$$\frac{1}{\omega}\|XY - Z_\omega\|_F = \|S\|_F. \tag{3.8}$$

Therefore, after the first two steps in (2.11),

$$\frac{1}{\omega}\|X_+(\omega)Y_+(\omega) - Z_\omega\|_F \leq \|S\|_F$$

and the strict inequality holds unless the first two equations of the optimality conditions of (2.6) already hold. Or equivalently,

$$\frac{1}{\omega^2}\left(\|\mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega) - Z_\omega)\|_F^2 + \|\mathcal{P}_\Omega(X_+(\omega)Y_+(\omega) - Z_\omega)\|_F^2\right) \leq \|S\|_F^2. \tag{3.9}$$

Next we examine the residual reduction $\|S\|_F^2 - \|S_+(\omega)\|_F^2$ after each step of the algorithm in detail.

Lemma 3.2 *Let $(X_+(\omega), Y_+(\omega))$ be generated by (2.11) for any $\omega \geq 1$, then*

$$\frac{1}{\omega^2}\|X_+(\omega)Y_+(\omega) - Z_\omega\|_F^2 = \|(I - Q(\omega))S(I - P)\|_F^2 = \|S\|_F^2 - \rho_{12}(\omega), \tag{3.10}$$

where

$$\rho_{12}(\omega) \triangleq \frac{1}{\omega^2}\|X_+(\omega)Y_+(\omega) - XY\|_F^2 = \|SP\|_F^2 + \|Q(\omega)S(I - P)\|_F^2 \tag{3.11}$$

is the amount of residual reduction from $\|S\|_F^2$ after steps 1 and 2 in (2.11).

Proof From the definition of Z_ω and (3.5), we obtain

$$\begin{aligned} X_+(\omega)Y_+(\omega) - Z_\omega &= X_+(\omega)Y_+(\omega) - XY - \omega S = \omega SP + \omega Q(\omega)S(I - P) - \omega S \\ &= -\omega(I - Q(\omega))S(I - P), \end{aligned}$$

which proves the first equality in (3.10). Using (3.2) and (3.7), we have:

$$\begin{aligned} \|X_+(\omega)Y_+(\omega) - Z_\omega\|_F^2 &= \|X_+(\omega)Y_+(\omega) - XY\|_F^2 \\ &\quad + \omega^2\|S\|_F^2 - 2\omega S \bullet (X_+(\omega)Y_+(\omega) - XY) \\ &= \omega^2\|S\|_F^2 - \|X_+(\omega)Y_+(\omega) - XY\|_F^2 \\ &= \omega^2\|S\|_F^2 - \omega^2\rho_{12}(\omega), \end{aligned}$$

which proves the second equality in (3.10). □

After the third step in (2.11), we have

$$\|\mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega) - Z_+(\omega))\|_F = 0.$$

Since $\mathcal{P}_{\Omega^c}(Z_\omega) \equiv \mathcal{P}_{\Omega^c}(XY)$ independent of ω, the residual reduction in the third step is

$$\rho_3(\omega) \triangleq \frac{1}{\omega^2}\|\mathcal{P}_{\Omega^c}(X_+(\omega)Y_+(\omega) - XY)\|_F^2. \tag{3.12}$$

Finally, the change of the residual value after the fourth step is

$$\rho_4(\omega) \triangleq \frac{1}{\omega^2}\|\mathcal{P}_{\Omega}(X_+(\omega)Y_+(\omega) - Z_\omega)\|_F^2 - \|S_+(\omega)\|_F^2;$$

or equivalently,

$$\rho_4(\omega) \triangleq \frac{1}{\omega^2}\|S_+(\omega) + (\omega - 1)S\|_F^2 - \|S_+(\omega)\|_F^2. \tag{3.13}$$

Clearly, $\rho_4(1) = 0$. For $\omega > 1$, it follows from (3.13) that

$$\frac{\omega^2\rho_4(\omega)}{\omega - 1} = (\omega - 1)(\|S\|_F^2 - \|S_+(\omega)\|_F^2) - 2S_+(\omega) \bullet (S_+(\omega) - S). \tag{3.14}$$

We will show next that the rate of change of $\rho_4(\omega)$ at $\omega = 1^+$ is nonnegative.

Lemma 3.3

$$\lim_{\omega\to 1^+}\frac{\rho_4(\omega)}{\omega - 1} = 2\|\mathcal{P}_{\Omega^c}(X_+(1)Y_+(1) - XY)\|_F^2 \geq 0. \tag{3.15}$$

Proof Let $\omega \to 1$ and $S_+ \triangleq S_+(1)$. We obtain from (3.14), the definitions of S in (2.9), and (3.2) that

$$\begin{aligned}
\lim_{\omega\to 1^+}\frac{\rho_4(\omega)}{\omega - 1} = \lim_{\omega\to 1^+}\frac{\omega^2\rho_4(\omega)}{\omega - 1} &= -2S_+ \bullet (S_+ - S)\\
&= -2\|S_+ - S\|_F^2 - 2S \bullet (S_+ - S)\\
&= -2\|\mathcal{P}_{\Omega}(X_+Y_+ - XY)\|_F^2 + 2S \bullet (X_+Y_+ - XY)\\
&= 2\|\mathcal{P}_{\Omega^c}(X_+Y_+ - XY)\|_F^2,
\end{aligned}$$

which completes the proof. □

If $\rho_4(\omega)$ is continuous, then Lemma 3.3 guarantees that $\rho_4(\omega) > 0$ in some range of $\omega > 1$. In fact, suppose that $\text{rank}(Z_\omega) = \text{rank}(Z)$ as $\omega \to 1^+$. The equality $\text{rank}(YZ_\omega^\top Z_\omega Y^\top) = \text{rank}(YZ^\top ZY^\top)$ holds as $\omega \to 1^+$, hence, $\lim_{\omega\to 1^+}(YZ_\omega^\top Z_\omega Y^\top)^\dagger = (YZ^\top ZY^\top)^\dagger$ holds by [32]. The continuity of the product $X_+(\omega)Y_+(\omega)$

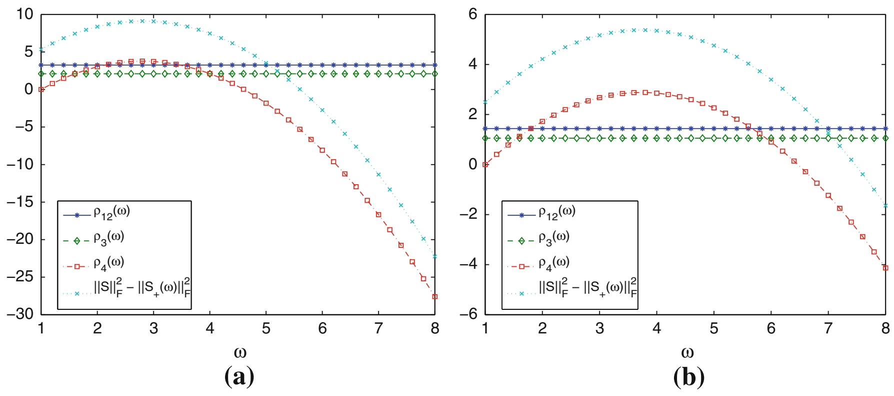


Fig. 2 Continuity of the functions $\rho_{12}(\omega)$, $\rho_3(\omega)$ and $\rho_4(\omega)$

implies that $\rho_4(\omega)$ is continuous as $\omega \to 1^+$. In Fig. 2a, b, we depict the continuity of the functions $\rho_{12}(\omega)$, $\rho_3(\omega)$ on a randomly generated problem from two different pair of points (X, Y, Z). As can be seen, the benefit of increasing ω can be quite significant. For example, in Fig. 2b, when ω is increased from 1 to 4, the amount of total residual reduction is more than doubled.

We have proved the following result about the residual-reduction property of the nonlinear SOR algorithm.

Theorem 3.4 *Assume that rank*$(Z_\omega) =$ *rank*$(Z), \forall\, \omega \in [1, \omega_1]$ *for some* $\omega_1 \geq 1$. *Let* $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ *be generated by the SOR scheme* (2.11) *starting from a non-stationary point* (X, Y, Z)*, and* ρ_{12}, ρ_3 *and* ρ_4 *be defined as in* (3.11)*,* (3.12) *and* (3.13)*, respectively. Then there exists some* $\omega_2 \geq 1$ *such that*

$$\|S\|_F^2 - \|S_+(\omega)\|_F^2 = \rho_{12}(\omega) + \rho_3(\omega) + \rho_4(\omega) > 0, \quad \forall\, \omega \in [1, \omega_2], \tag{3.16}$$

where $\rho_{12}(\omega), \rho_3(\omega) \geq 0$ *by definition. Moreover, whenever* $\rho_3(1) > 0$ (*equivalently* $\mathcal{P}_{\Omega^c}(X_+(1)Y_+(1) - XY) \neq 0$)*, there exists* $\bar{\omega} > 1$*, so that* $\rho_4(\omega) > 0, \forall\, \omega \in (1, \bar{\omega}]$.

Next we present a convergence result for our algorithm. Since model (1.4) is non-convex, we are only able to establish convergence to a stationary point under a mild assumption. Note that the objective function is bounded below by zero and is decreased by at least an amount of ρ_3 at every iteration. There must hold (see (3.12))

$$\mathcal{P}_{\Omega^c}(X^{k+1}Y^{k+1} - X^kY^k) \to 0.$$

In light of the above, it is reasonable to assume that $\{\mathcal{P}_{\Omega^c}(X^kY^k)\}$ remains bounded, barring the unlikely alternative that $\|\mathcal{P}_{\Omega^c}(X^kY^k)\| \to \infty$.

Theorem 3.5 *Let* $\{(X^k, Y^k, Z^k)\}$ *be generated by Algorithm* 1 *and* $\{\mathcal{P}_{\Omega^c}(X^kY^k)\}$ *be bounded. Then there exists at least a subsequence of* $\{(X^k, Y^k, Z^k)\}$ *that satisfies the first-order optimality conditions* (2.6) *in the limit.*

Proof It follows from the boundedness of $\{\mathcal{P}_{\Omega^c}(X^kY^k)\}$ and the algorithm construction that both $\{Z^k\}$ and $\{X^kY^k\}$ are bounded sequences. It suffices to prove (2.6a)–(2.6b) since the other conditions are satisfied by the construction of Algorithm 1. Without loss of generality, we assume that $\{X^k\}$ is generated by a scheme analogous to (2.4): given $(X, Y) = (X^k, Y^k)$ and $\omega \in [1, \tilde{\omega}]$

$$Z_\omega = \omega Z + (1-\omega)XY, \quad X_+ = \text{orth}(Z_\omega Y^T), \quad Y_+ = X_+^T Z_w.$$

Obviously, $\{X^k\}$ is bounded. In addition, $\{Y^k\}$ is also bounded due to the boundedness of both $\{Z^k\}$ and $\{X^kY^k\}$.

Let $\mathcal{I} = \{k : \rho_4^k(\omega^k) \geq 0\}$, and $\mathcal{I}^c$ be the complement of $\mathcal{I}$. It follows from (3.16) that

$$\|S^0\|_F^2 \geq \sum_{i\in\mathcal{I}} \rho_{12}^i(\omega) = \sum_{i\in\mathcal{I}} \|S^iP^i\|_F^2 + \|Q^iS^i(I-P^i)\|_F^2. \tag{3.17}$$

We consider the following three cases.

(i) Suppose $|\mathcal{I}^c| < \infty$. It follows from (3.17) that

$$\lim_{i\to\infty} \|S^iP^i\|_F^2 = 0 \text{ and } \lim_{i\to\infty} \|Q^iS^i\|_F^2 = 0. \tag{3.18}$$

The construction of the scheme (2.11) gives the equalities:

$$\mathcal{P}_\Omega(M) = \mathcal{P}_\Omega(Z^i), \quad \mathcal{P}_{\Omega^c}(Z^i) = \mathcal{P}_{\Omega^c}(X^iY^i), \quad P^i = U^i(U^i)^\top,$$

where $U^i = \texttt{orth}((Y^i)^\top)$. Therefore, we obtain

$$\begin{aligned} S^iP^i &= \mathcal{P}_\Omega(M - X^iY^i)P^i = \mathcal{P}_\Omega(Z^i - X^iY^i)P^i \\ &= (Z^i - X^iY^i)P^i = (Z^i - X^iY^i)U^i(U^i)^\top, \end{aligned}$$

which yields $\lim_{i\to\infty}(Z^i - X^iY^i)U^i = 0$ in view of the first part of (3.18). Since U^i is an orthonormal basis for $\mathcal{R}((Y^i)^\top)$ and the sequence $\{Y^i\}$ is bounded, we have

$$\lim_{i\to\infty} (Z^i - X^iY^i)(Y^i)^\top = 0. \tag{3.19}$$

Using $Q^i = V^i(V^i)^\top$, where V^i is an orthonormal basis for $\mathcal{R}(X^{i+1})$, we obtain

$$Q^iS^i = Q^iS^{i+1} + Q^i(S^i - S^{i+1}) = V^i(V^i)^\top(Z^{i+1} - X^{i+1}Y^{i+1}) + Q^i(S^i - S^{i+1}). \tag{3.20}$$

Using (3.7) and (3.18), we obtain

$$\|S^i - S^{i+1}\|_F^2 \leq \|X^{i+1}Y^{i+1} - X^iY^i\|_F^2 \leq (\tilde{\omega})^2(\|S^iP^i\|_F^2 + \|Q^iS^i(I-P^i)\|_F^2) \to 0,$$

hence, $\lim_{i\to\infty}\|S^i - S^{i+1}\|_F = 0$. This fact, together with (3.18) and (3.20), proves

$$(V^i)^\top(Z^{i+1} - X^{i+1}Y^{i+1}) \to 0.$$

In view of the boundedness of $\{X^i\}$, we arrive at

$$\lim_{i\to\infty}(X^i)^\top(X^iY^i - Z^i) = 0. \tag{3.21}$$

(ii) Suppose $|\mathcal{I}^c| = \infty$ and $|\{k \in \mathcal{I}^c : \gamma(\omega^k) \geq \gamma_1\}| < \infty$. That is, for $k \in \mathcal{I}^c$ sufficiently large we have

$$\|S^{k+1}\|_F < \gamma_1\|S^k\|_F.$$

Consquently, $\lim_{k\to\infty, k\in\mathcal{I}^c}\|S^k\|_F = 0$. Since $\|S^k\|$ is nonincreasing, the full sequence converges to the global minimizer of (1.4).

(iii) Suppose $|\mathcal{I}^c| = \infty$ and $|\{i \in \mathcal{I}^c : \gamma(\omega^i) \geq \gamma_1\}| = \infty$. Then Algorithm 1 resets $\omega^i = 1$ for an infinite number of iterations. We obtain from (3.17) that

$$\|S^0\|_F^2 \geq \sum_{i\in\mathcal{I}_1}\rho_{12}^i(\omega) = \sum_{i\in\mathcal{I}_1}\|S^iP^i\|_F^2 + \|Q^iS^i(I-P^i)\|_F^2. \tag{3.22}$$

Hence, the subsequence in $\mathcal{I}_1$ satisfies (3.19) and (3.21) by repeating, in an analogous fashion, the proof of part i). □

4 Computational results

In this section, we report numerical results on our nonlinear SOR algorithm and other algorithms. The code `LMaFit` [40] for our algorithm is implemented in Matlab with a couple of small tasks written in C to avoid ineffective memory usage in Matlab. Other tested solvers include `APGL` (version 2012) [33], `FPCA` [24] and `OptSpace` [18], where the first two are nuclear minimization codes implemented under the Matlab environment. `APGL` also utilizes a Matlab version (with the task of reorthogonalization implemented in C) of the SVD package PROPACK [19], and `FPCA` uses a fast Monte Carlo algorithm for SVD calculations implemented in Matlab. The code `OptSpace`, which has a C version that was used in our tests, solves a model closely related to (1.4) using a gradient descent approach and starting from a specially constructed initial guess. All experiments were performed on a Lenovo D20 Workstation with two Intel Xeon E5506 Processors and 10GB of RAM.

We tested and compared these solvers on two classes of matrix problems: completion and low-rank approximation. The key difference between the two classes lies in whether a given sample is from a true low-rank matrix (with or without noise) or not. Although theoretical guarantees exist for matrix completion, to the best of our knowledge no such guarantees exist for low-rank approximation if samples are taken from a matrix of mathematically full rank. On the other hand, low-rank approximation problems are more likely to appear in practical applications.

4.1 Implementation details and rank estimation

Algorithm 1 starts from an initial guess $Y^0 \in \mathbb{R}^{K\times n}$. For the sake of simplicity, in all our experiments we set Y^0 to a diagonal matrix with 1's on the diagonal even though more elaborate choices certainly exist that may lead to better performance. The default values of the parameters $\tilde{\omega}$, δ and γ_1 were set to $+\infty$, 1 and 0.7, respectively. Since the increment δ is non-decreasing in Algorithm 1, the parameter ω can be increased too fast. Hence, we also reset δ to $0.1 * \max(\omega - 1, \delta)$ whenever $\gamma(\omega) \geq 1$. The stopping criteria in our numerical experiments follow

$$\texttt{relres} = \frac{\|\mathcal{P}_\Omega(M - X^k Y^k)\|_F}{\|\mathcal{P}_\Omega(M)\|_F} \leq \texttt{tol}$$

and

$$\texttt{reschg} = \left| 1 - \frac{\|\mathcal{P}_\Omega(M - X^k Y^k)\|_F}{\|\mathcal{P}_\Omega(M - X^{k-1} Y^{k-1})\|_F} \right| \leq \texttt{tol}/2,$$

where `tol` is a moderately small number.

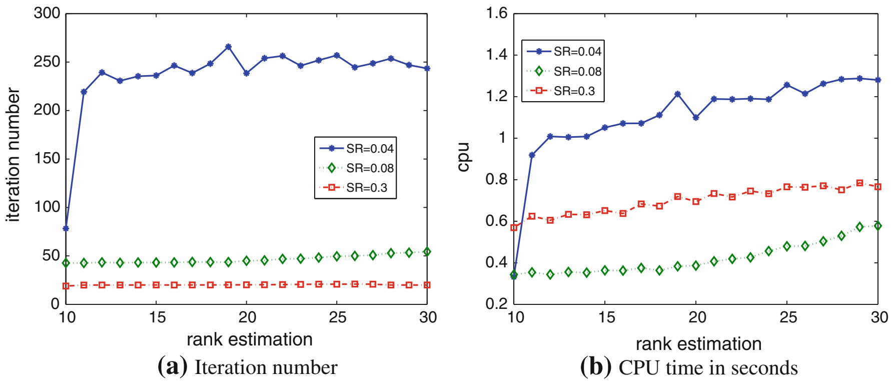
Since a proper estimation to the rank K for the model (1.4) is essential for the success of `LMaFit`, two heuristic strategies for choosing K were implemented. In the first strategy, we start from a large K ($K \geq r$) and decrease it aggressively once a dramatic change in the estimated rank of the variable X is detected based on its QR factorization [11] which usually occurs after a few iterations.

Specifically, let $QR = XE$ be the economy-size QR factorization of X, where E is a permutation matrix so that $d := |\text{diag}(R)|$ is non-increasing, where $\text{diag}(R)$ is a vector whose ith component is R_{ii}. We compute the quotient sequence $\tilde{d}_i = d_i/d_{i+1}$, $i = 1, \ldots, K-1$, and examine the ratio

$$\tau = \frac{(K-1)\tilde{d}(p)}{\sum_{i \neq p} \tilde{d}_i},$$

where $\tilde{d}(p)$ is the maximal element of $\{\tilde{d}_i\}$ and p is the corresponding index. A large τ value indicates a large drop in the magnitude of d right after the pth element. In the current implementation, we reset K to p once $\tau > 10$, and this adjustment is done only one time. On the other hand, by starting from a small initial guess, our second strategy is to increase K to $\min(K + \kappa, \texttt{rank\_max})$ when the algorithm stagnates, i.e., `reschg<10*tol`. Here, `rank_max` is the maximal rank estimation, and the increment $\kappa = \texttt{rk\_inc}$ if the current $K < 50$; otherwise, $\kappa = 2 * \texttt{rk\_inc}$. The default value of `rk_inc` is 5. In our code `LMaFit`, the first and second (or decreasing and increasing rank) strategies can be specified by setting the option `est_rank` to 1 or 2, respectively, and will be called the decreasing rank and increasing rank strategies, respectively.

Each strategy has its own advantages and disadvantages, and should be selected according to the properties of the targeted problems. As will be shown by our numerical results, the decreasing rank strategy is preferable for reasonably well-conditioned



(a) Iteration number **(b)** CPU time in seconds

Fig. 3 The sensitivity `LMaFit`with respect to the initial rank estimation K

matrix completion problems, while the increasing rank strategy is more suitable for low-rank approximation problems where there does not exist a clear-cut desirable rank. Based on these observations, we use the decreasing rank strategy in the experiments of Sect. 4.2, while the increasing rank strategy is used in Sects. 4.3–4.5.

4.2 Experiments on random matrix completion problems

The test matrices $M \in \mathbb{R}^{m\times n}$ with rank r in this subsection were created randomly by the following procedure (see also [24]): two random matrices $M_L \in \mathbb{R}^{m\times r}$ and $M_R \in \mathbb{R}^{n\times r}$ with i.i.d. standard Gaussian entries were first generated and then $M = M_L M_R^\top$ was assembled; then a subset Ω of p entries was sampled uniformly at random. The ratio $p/(mn)$ between the number of measurements and the number of entries in the matrix is denoted by "SR" (sampling ratio). The ratio $r(m+n-r)/p$ between the degree of freedom in a rank r matrix to the number of samples is denoted by "FR".

We first evaluate the sensitivity of `LMaFit` to the initial rank estimate K using the decreasing rank strategy of rank estimation. In this test, we used matrices with $m = n = 1{,}000$ and $r = 10$. Three test cases were generated at the sampling ratios SR equal to 0.04, 0.08 and 0.3, respectively. In each case, we ran `LMaFit` for each of $K = 10, 11, 12, \ldots, 30$ on 50 random instances. The average number of iterations and average CPU time corresponding to this set of K values are depicted in Fig. 3a and b, respectively. In these two figures, we observe a notable difference at the rank estimate $K = 10$ when the sampling ratio SR $= 0.04$. The reason is that at this low sampling ratio the rank estimate routine of `LMaFit` mistakenly reduced the working rank to be less than 10 and resulted in premature exists. For all other cases, we see that the number of iterations stayed at almost the same level and the CPU time only grew slightly as K increased from 10 to 30. Overall, we conclude that `LMaFit` is not particularly sensitive to the change of K value on this class of problems over a considerable range. Based on this observation, in all tests using the decreasing rank strategy, we set the initial rank estimate K either to $\lfloor 1.25r\rfloor$ or to $\lfloor 1.5r\rfloor$, where $\lfloor x\rfloor$ is the largest integer not exceeding x. Numerical results generated from these two K values should still be sufficiently representative.

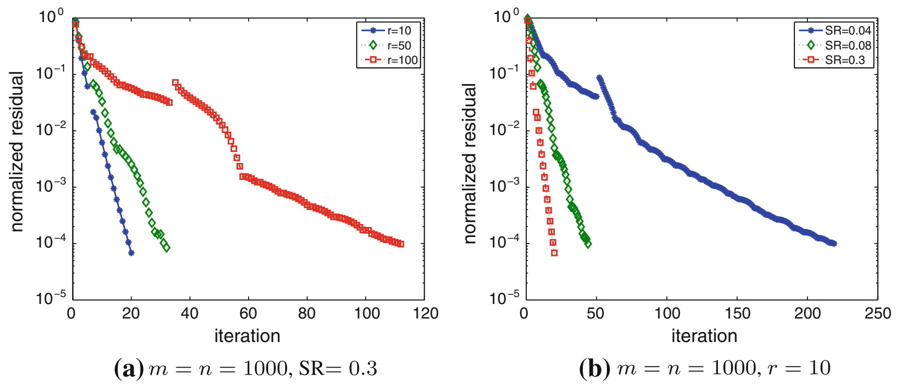


Fig. 4 Convergence behavior of the residual in `LMaFit`runs

Our next test is to study the convergence behavior of `LMaFit` with respect to the sampling ratio and true rank of M. In this test the dimension of M were set to $m = n =$ 1,000 and the initial rank estimate was set to $\lfloor 1.25r \rfloor$ in `LMaFit`. In Fig. 4a, we plot the normalized residual $\|\mathcal{P}_\Omega(M - XY)\|_F / \|\mathcal{P}_\Omega(M)\|_F$ at all iterations for three test cases where the sampling ratio was fixed at SR = 0.3 and rank $r = 10, 50$ and 100, respectively. On the other hand, Fig. 4b is for three test cases where SR = 0.04, 0.08 and 0.3, respectively, while the rank was fixed at $r = 10$. Not surprisingly, these figures show that when the sampling ratio is fixed, the higher the rank is, the harder the problem is; and when the rank is fixed, the smaller the sampling ratio is, the harder the problem is. In all cases, the convergence of the residual sequences appeared to be linear, but at quite different rates.

An important question about the factorization model (1.4) and our nonlinear SOR algorithm is whether or not our approach (model plus algorithm) has an ability in recovering low-rank matrices similar to that of solving the nuclear norm minimization model by a good solver. Or simply put, does our algorithm for (1.4) provide a comparable recoverability to that of a good nuclear norm minimization algorithm for (1.2) or (1.3)? We address this recoverability issue in the next test by generating phase diagrams in Fig. 5a, b for the two models (1.3) and (1.4), respectively. The solver `FPCA` [24] was chosen to solve (1.3) since it has been reported to have a better recoverability than a number of other nuclear norm minimization solvers. In this test, we used random matrices of size $m = n = 500$.

We ran each solver on 50 randomly generated problems with the sampling ratio SR chosen in the order as it appear in $\{0.01, 0.06, 0.11, \dots, 0.86\}$ and with each rank value $r \in \{5, 8, 11, \dots, 59\}$. The two phase diagrams depict the success rates out of every 50 runs by each solver for each test case where a run was successful when the relative error $\|M - W\|_F / \|M\|_F$ between the true and the recovered matrices M and W was smaller than 10^{-3}. If a solver recovered all 50 random instances successfully for SR $= \alpha$ and $r = \beta$, then it ought to have equal or higher recoverability for SR $> \alpha$ and $r = \beta$. To expedite the part of the experiment involving `FPCA`, we chose to stop testing all other SR $> \alpha$ with $r = \beta$, and increment the r value. In Fig. 5a, b, a white box indicates a 100% recovery rate, while a black box means a 0% rate. The parameter setting for `LMaFit` was `tol` $= 10^{-4}$, $K = \lfloor 1.25r \rfloor$ and `est_rank` $= 1$,

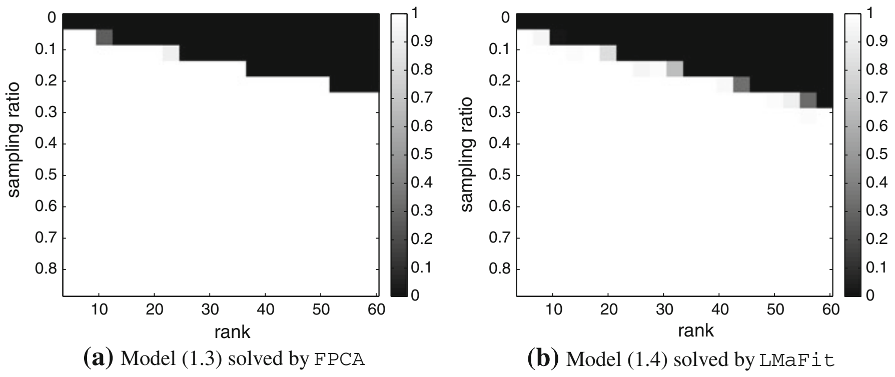


Fig. 5 Phase diagrams for matrix completion recoverability

while for FPCA it was tol $= 10^{-4}$ and $\mu = 10^{-4}$. All other parameters were set to their respective default values. The two phase diagrams indicate that the recoverability of LMaFit is marginally inferior to that of FPCA in this experiment. Given the reported better recoverability of FPCA, it is reasonable to infer that the recoverability of LMaFit is comparable to those of the other nuclear norm minimization solvers studied in [24].

To have a quick assessment on the speed of LMaFit relative to those of other state-of-the-art solvers, we compared LMaFit with two nuclear norm minimization solvers, APGL and FPCA, and with a c version of OptSpace that solves a factorization model similar to ours but uses an SVD-based initial guess, on a set of small problems with $m = n = 1000$. The parameter setting for LMaFit was the same as in the previous experiment. In particular, the decreasing rank strategy was used. The parameter μ for the model (1.3) was set to $10^{-4}\sigma$ as suggested by the testing scripts in the package APGL, where σ is the largest singular value of $\mathcal{P}_\Omega(M)$. The stopping tolerance for all solvers was set to 10^{-4} and all other parameters were set to their default values. A summary of the computational results is presented in Table 1, where "time" denotes the CPU time measured in seconds and rel.err $:= \|W - M\|_F/\|M\|_F$ denotes the relative error between the true and the recovered matrices M and W ("tsvd" will be explained below).

From Table 1, we see that LMaFit is at least several times (often a few orders of magnitude) faster than all other solvers to achieve a comparable accuracy. We note that the accuracy of the solver OptSpace on problems with rank 100 could not be improved by using a smaller tolerance. Of course, the reported performances of all the solvers involved were pertinent to their tested versions under the specific testing environment. Improved performances are possible for different parameter settings, on different test problems, or by different versions. However, given the magnitude of the timing gaps between LMaFit and others, the speed advantage of LMaFit should be more than evident on these test problems. (We also tested LMaFit with the increasing rank strategy and found that it was not as effective as the decreasing rank strategy on these random matrix completion problems.)

In Table 1, the item "tsvd" is the percentage of CPU time spent on SVD-related calculations, as estimated by the MATLAB profiler and obtained from separate runs.

Table 1 Comparison of four solvers on small problems with varying rank and sampling ratio

Problem			APGL				FPCA			OptSpace		LMaFit			
												$K = \lfloor 1.25r \rfloor$		$K = \lfloor 1.5r \rfloor$	
r	SR	FR	μ	Time	Rel.arr	tsvd (%)	Time	Rel.arr	tsvd (%)	Time	Rel.arr	Time	Rel.arr	Time	Rel.arr
10	0.04	0.50	5.76e−03	3.01	2.91e−03	84	28.68	8.21e−01	17	21.98	4.44e−04	0.89	4.72e−04	0.95	4.35e−04
10	0.08	0.25	1.02e−02	1.62	4.63e−04	75	11.04	7.30e−04	18	10.16	2.42e−04	0.34	2.27e−04	0.36	2.19e−04
10	0.15	0.13	1.78e−02	1.97	2.47e−04	75	7.01	4.21e−04	42	8.26	1.32e−04	0.37	1.16e−04	0.39	1.48e−04
10	0.30	0.07	3.42e−02	2.89	1.56e−04	70	16.63	1.97e−04	71	9.36	1.02e−04	0.60	8.99e−05	0.64	9.91e−05
50	0.20	0.49	2.94e−02	75.50	2.75e−01	96	66.78	4.64e−04	55	312.05	2.71e−04	3.79	3.03e−04	4.75	2.63e−04
50	0.25	0.39	3.59e−02	21.23	5.52e−04	90	97.44	3.24e−04	65	228.40	1.84e−04	2.84	1.89e−04	3.03	2.11e−04
50	0.30	0.33	4.21e−02	15.29	6.20e−04	88	140.91	2.64e−04	73	236.26	8.90e−05	2.51	1.78e−04	2.70	1.91e−04
50	0.40	0.24	5.53e−02	15.86	3.68e−04	86	41.27	2.16e−04	77	120.72	7.79e−05	2.25	1.11e−04	2.61	1.65e−04
100	0.35	0.54	5.70e−02	49.96	1.42e−03	93	249.19	5.41e−04	77	1,410.64	2.83e−04	12.46	3.01e−04	16.41	3.09e−04
100	0.40	0.47	6.37e−02	43.11	5.50e−04	92	290.57	4.11e−04	79	1,202.14	2.33e−04	9.15	2.56e−04	10.74	2.41e−04
100	0.50	0.38	7.71e−02	50.12	4.71e−04	92	374.63	3.10e−04	80	908.55	1.65e−04	6.77	1.55e−04	7.11	1.92e−04
100	0.55	0.35	8.40e−02	42.56	4.32e−04	90	353.15	2.89e−04	81	851.55	1.52e−04	6.21	1.14e−04	6.97	9.99e−05

As can be seen, for `APGL` and `FPCA` SVD-related calculations essentially dominate their total costs (with the exception of extremely low-rank cases for `FPCA`). On the other hand, for `LMaFit`, the total cost is dominated by low-rank or sparse matrix to matrix multiplications (which are also required by other solvers), while the cost of solving the least squares problem in (2.11c) is either negligible or at most 11 % of the total CPU time. Therefore, avoiding SVD-related calculations is a main reason why `LMaFit` is much faster than the nuclear norm minimization solvers `APGL` and `FPCA`, validating our original motivation of solving the factorization model.

The next test was on large-scale random matrix completion problems in which we compared `LMaFit` with `APGL` following the experiment setup given in Sect. 4.2 of [33]. The other solvers `FPCA` and `OptSpace` were excluded from this comparison since they would have demanded excessive CPU times. Summaries of the computational results are presented in Table 2 for noiseless data and Table 3 for noisy data, where both the noiseless and noisy data were generated as in [33]. In these two table, "iter" denotes the number of iterations used, and "#sv" denotes the rank of the recovered solution. The statistics contained in these two tables verify two key observations: (a) solving the factorization model is reliable for matrix completion on a wide range of problems, and (b) our nonlinear SOR algorithm, as implemented in `LMaFit`, has a clear speed advantage in solving many large-scale problems.

4.3 Experiments on random low-rank approximation problems

We now consider applying matrix completion algorithms to randomly generated low-rank matrix approximation problems. The goal is to find a low-rank approximation to a mathematically full-rank matrix M whose singular values gradually tend to zero,

Table 2 Numerical results on large random matrix completion problems without noise

Problem				APGL					LMaFit ($K = \lfloor 1.25r \rfloor$)				LMaFit ($K = \lfloor 1.5r \rfloor$)			
n	r	SR	FR	μ	Iter	#sv	Time	Rel.arr	Iter	#sv	Time	Rel.arr	Iter	#sv	Time	Rel.arr
1000	10	0.119	0.167	1.44e−02	38	10	2.16	3.42e−04	28	10	0.36	1.63e−04	28	10	0.35	1.68e−04
1000	50	0.390	0.250	5.36e−02	40	50	13.70	3.08e−04	21	50	0.83	1.46e−04	24	50	0.97	1.45e−04
1000	100	0.570	0.334	8.58e−02	48	100	40.93	3.84e−04	21	100	1.58	1.57e−04	21	100	1.57	1.79e−04
5000	10	0.024	0.166	1.37e−02	48	10	9.98	1.85e−04	35	10	2.46	1.36e−04	35	10	2.64	1.39e−04
5000	50	0.099	0.200	6.14e−02	48	50	86.10	2.89e−04	29	50	22.44	1.58e−04	29	50	23.60	1.68e−04
5000	100	0.158	0.250	1.02e−01	52	100	242.09	2.60e−04	27	100	23.47	1.56e−04	28	100	24.85	1.32e−04
10000	10	0.012	0.166	1.37e−02	49	10	19.66	1.68e−04	38	10	5.68	1.48e−04	38	10	6.15	1.61e−04
10000	50	0.050	0.200	5.97e−02	50	50	188.35	2.54e−04	31	50	59.77	1.51e−04	31	50	64.04	1.53e−04
10000	100	0.080	0.250	9.94e−02	50	100	501.29	3.71e−04	34	100	187.41	1.74e−04	34	100	195.50	1.81e−04
20000	10	0.006	0.167	1.35e−02	55	10	48.44	2.03e−04	44	10	14.09	1.73e−04	45	10	15.63	1.74e−04
30000	10	0.004	0.167	1.35e−02	52	10	71.23	2.97e−04	46	10	24.00	1.40e−04	46	10	26.61	1.41e−04
50000	10	0.002	0.167	1.35e−02	61	10	153.55	1.73e−04	50	10	54.94	1.65e−04	49	10	59.33	1.64e−04
100000	10	0.001	0.167	1.34e−02	62	10	368.40	4.16e−04	52	10	144.61	1.57e−04	49	10	159.65	1.37e−04

Table 3 Numerical results on large random matrix completion problems with noise

Problem				APGL					LMaFit ($K = \lfloor 1.25r \rfloor$)				LMaFit ($K = \lfloor 1.5r \rfloor$)			
n	r	SR	FR	μ	Iter	#sv	Time	Rel.arr	Iter	#sv	Time	Rel.arr	Iter	#sv	Time	Rel.arr
1000	10	0.119	0.167	1.44e−02	33	10	2.45	4.54e−02	19	10	0.26	4.53e−02	19	10	0.24	4.53e−02
1000	50	0.390	0.250	5.36e−02	38	50	12.65	5.51e−02	17	50	0.66	5.51e−02	21	50	0.82	5.51e−02
1000	100	0.570	0.334	8.59e−02	44	100	33.50	6.40e−02	17	100	1.24	6.40e−02	17	100	1.33	6.40e−02
5000	10	0.024	0.166	1.38e−02	41	10	11.19	4.52e−02	26	10	1.81	4.51e−02	26	10	1.98	4.51e−02
5000	50	0.099	0.200	6.14e−02	38	50	59.31	4.98e−02	20	50	15.51	4.97e−02	20	50	16.76	4.97e−02
5000	100	0.158	0.250	1.02e−01	45	100	189.76	5.69e−02	19	100	16.75	5.68e−02	18	100	16.16	5.68e−02
10000	10	0.012	0.166	1.37e−02	45	10	26.92	4.52e−02	29	10	4.38	4.52e−02	29	10	4.84	4.52e−02
10000	50	0.050	0.200	5.97e−02	41	50	137.55	4.99e−02	23	50	46.14	4.99e−02	23	50	50.11	4.99e−02
10000	100	0.080	0.250	9.95e−02	49	100	493.15	5.73e−02	22	100	125.06	5.73e−02	22	100	133.10	5.73e−02
20000	10	0.006	0.167	1.35e−02	50	10	52.47	4.53e−02	33	10	10.70	4.53e−02	33	10	12.13	4.53e−02
30000	10	0.004	0.167	1.35e−02	52	10	87.87	4.52e−02	34	10	18.45	4.52e−02	34	10	21.18	4.52e−02
50000	10	0.002	0.167	1.35e−02	57	10	183.44	4.53e−02	37	10	39.92	4.53e−02	37	10	45.23	4.53e−02
100000	10	0.001	0.167	1.34e−02	58	10	398.58	4.53e−02	40	10	129.09	4.53e−02	40	10	142.82	4.53e−02

though none is exactly zero. Since there does not exist a "best low rank matrix" in such approximations, any evaluation of solution quality must take into consideration of two competing criteria: rank and accuracy. The only clear-cut case of superiority is when one solution dominates another by both criteria, i.e., a lower rank approximation with a higher accuracy.

In this experiment, all random instances of $M \in \mathbb{R}^{m\times n}$ were created as follows: two matrices $M_L \in \mathbb{R}^{n\times n}$ and $M_R \in \mathbb{R}^{n\times n}$ with i.i.d. standard Gaussian entries are first generated randomly; then M_L and M_R are orthogonalized to obtain U and V, respectively; finally the matrix $M = U\Sigma V^\top$ is assembled. Here Σ is a diagonal matrix whose diagonal elements σ_i, for $i = 1, \ldots, n$, are either the power-law decaying, that

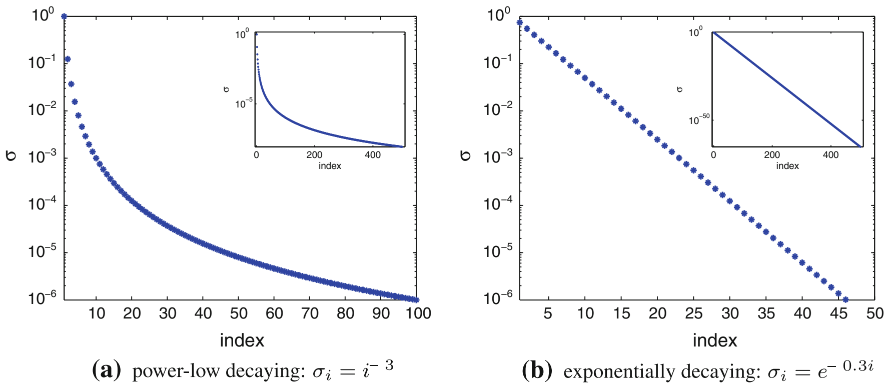


Fig. 6 Illustration of decaying patterns of the singular values σ

Table 4 Numerical results on approximate low-rank problems

Problem		APGL				FPCA			LMaFit (est_rank=1)			LMaFit (est_rank=2)		
SR	FR	μ	#sv	Time	Rel.arr	#sv	Time	Rel.arr	#sv	Time	Rel.arr	#sv	Time	Rel.arr
Power-low decaying														
0.04	0.99	1.00e−04	96	5.14	6.59e−01	1	31.45	1.39e−01	5	0.69	3.68e−01	11	0.31	8.96e−03
0.08	0.49	1.00e−04	92	4.91	2.25e−01	2	36.90	4.38e−02	5	1.55	2.20e−01	20	0.43	1.13e−03
0.15	0.26	1.00e−04	63	3.12	2.64e−02	4	12.17	1.78e−02	5	1.38	1.52e−01	20	0.70	4.57e−04
0.30	0.13	1.00e−04	11	2.11	2.39e−03	4	28.65	1.04e−02	5	2.97	8.12e−02	22	1.26	2.36e−04
Exponentially decaying														
0.04	0.99	1.00e−04	100	5.65	7.54e−01	14	31.69	5.05e−01	5	0.47	3.92e−01	16	0.85	4.08e−01
0.08	0.49	1.00e−04	102	5.96	3.47e−01	8	35.75	1.24e−01	5	0.43	2.66e−01	26	1.80	1.98e−02
0.15	0.26	1.00e−04	99	5.96	5.55e−02	13	11.40	2.76e−02	5	0.62	2.39e−01	28	1.61	7.26e−04
0.30	0.13	1.00e−04	74	4.08	9.61e−03	14	27.81	1.71e−02	6	1.02	1.71e−01	30	2.01	2.38e−04

is, $\sigma_i = i^{-3}$, or the exponentially decaying, that is, $\sigma_i = e^{-0.3i}$. Hence, all singular values are positive, and there are 99 and 46 entries whose magnitude are greater than 10^{-6} in these two types of Σ, respectively. These diagonals are illustrated in Fig. 6a, b. The sampling procedures are the same as those in Sect. 4.2. In this test, the dimension and rank of M were set to $n = 500$ and $r = 10$, respectively.

We compared `LMaFit` with the solvers `APGL` and `FPCA` [24]. The parameter μ for the model (1.3) was set to 10^{-4}. The stopping tolerance for all solvers was set to 10^{-4}. We set the parameters `truncation` = 1, and `truncation_gap` = 100 in `APGL`. For `LMaFit` with `est_rank` = 1, we set $K = 50$, and for `LMaFit` with `est_rank` = 2, we set $K = 1$, `rank_max` = 50 and `rk_inc` = 1. All other parameters were set to default values for the two solvers. A summary of the computational results is presented in Table 4. We can see that `LMaFit` with `est_rank` = 2 was significantly better than other solvers. The decreasing rank strategy of `LMaFit`, as it is currently implemented with `est_rank` = 1, is clearly not suitable for these low-rank approximation problems since there is no "true low rank" as in matrix completion problems. Specifically, this strategy (`est_rank` = 1) reduced the rank estimate too aggressively.

4.4 Experiments on "real data"

In this subsection, we consider low-rank matrix approximation problems based on two "real data" sets: the Jester joke data set [9] and the MovieLens data set [14]. In these data set, only partial data are available from the underlying unknown matrices which are unlikely to be of exactly low rank. Nevertheless, matrix completion solvers have been applied to such problems to test their ability in producing low-rank approximations. As is mentioned above, an assessment of solution quality should take into consideration of both rank and accuracy. The Jester joke data set consists of four problems "jester-1", "jester-2", "jester-3" and "jester-all", where the last one is obtained by combining all of the first three data sets, and the MovieLens data set has three problems "movie-100K", "movie-1M" and "movie-10M".[1] For `LMaFit`, we set the parameters to $\texttt{tol} = 10^{-3}$, $\texttt{est\_rank} = 2$, $K = 1$, and $\texttt{rk\_inc} = 2$. For `APGL`, the parameter setting was $\texttt{tol} = 10^{-3}$, $\texttt{truncation} = 1$, and $\texttt{truncation\_gap} = 20$. In addition, the model parameter μ for `APGL` was set to $\mu = 10^{-4}$ which produced better solutions than choosing $10^{-3}\sigma$ as suggested by the testing scripts in the package `APGL`, where σ is the largest singular value of the sampling matrix. Moreover, we set the maximum rank estimate to 80 for the jester problems and to 100 for the MovieLens problems for both `LMaFit` and `APGL` by specifying their parameters `rank_max` or `maxrank`, respectively. We note that since the jester problems have only 100 columns, it is not meaningful to fit a matrix of rank 100 to a jester data set. Since the entries of a underlying matrix M are available only on an index set Ω, to measure accuracy we computed the Normalized Mean Absolute Error (NMAE) as was used in [9,24,33], i.e.,

$$\text{NMAE} = \frac{1}{(r_{\max} - r_{\min})|\Omega|} \sum_{(i,j)\in\Omega} |W_{i,j} - M_{i,j}|,$$

where $r_{\min}$ and $r_{\max}$ are the lower and upper bounds for the ratings. Specifically, we have $r_{\min} = -10$ and $r_{\max} = 10$ for the jester joke data sets and $r_{\min} = 1$ and $r_{\max} = 5$ for the MovieLens data sets. We tried using a part of the available data as was done in [33] and found that `APGL` generally returned solutions with slightly higher NMAE-accuracy but also higher ranks than those returned by `LMaFit`, creating difficulties in interpreting solution quality (though the speed advantage of `LMaFit` was still clear). Therefore, we only report numerical results using all the available data in Table 5, where "#asv" denotes the approximate rank of a computed solution defined as the total number of singular values exceeding 10^{-8}.

As can be seen from Table 5, `LMaFit` and `APGL` obtained low-rank approximation matrices of comparable quality on the all the problems, while `LMaFit` ran more than twice as fast, and returned matrices of slightly lower approximate ranks (except for "jester-all" and "movie-10M"). It is particularly interesting to compare the two solvers on problem "jester-3" for which `LMaFit` reported a solution of rank 43 while `APGL` of rank 80. Even with a much lower rank, the `LMaFit` solution is almost as accurate

[1] They are available at http://www.ieor.berkeley.edu/~Egoldberg/jester-data and http://www.grouplens.org, respectively.

Table 5 Numerical results on "real data"

Problem		APGL						LMaFit				
Name	m/n	μ	Iter	Time	NMAE	Rel.arr	#asv	Iter	Time	NMAE	Rel.arr	#asv
jester-1	24983/100	1.00e−04	33	84.02	2.34e−02	1.77e−01	80	117	46.24	2.50e−02	1.86e−01	78
jester-2	23500/100	1.00e−04	32	84.52	2.41e−02	1.78e−01	80	118	42.83	2.56e−02	1.87e−01	78
jester-3	24938/100	1.00e−04	34	46.16	9.30e−07	3.16e−05	80	235	29.50	4.06e−05	9.31e−04	43
jester-all	73421/100	1.00e−04	32	166.66	2.01e−02	1.66e−01	80	114	96.32	2.03e−02	1.65e−01	80
moive-100K	943/1682	1.00e−04	100	101.78	1.03e−03	2.05e−03	100	507	25.12	9.95e−04	2.07e−03	94
moive-1M	6040/3706	1.00e−04	61	172.76	6.67e−02	9.61e−02	100	190	67.44	6.78e−02	9.85e−02	92
moive-10M	71567/10677	1.00e−04	57	1,633.74	7.83e−02	1.32e−01	100	178	654.24	7.59e−02	1.29e−01	100

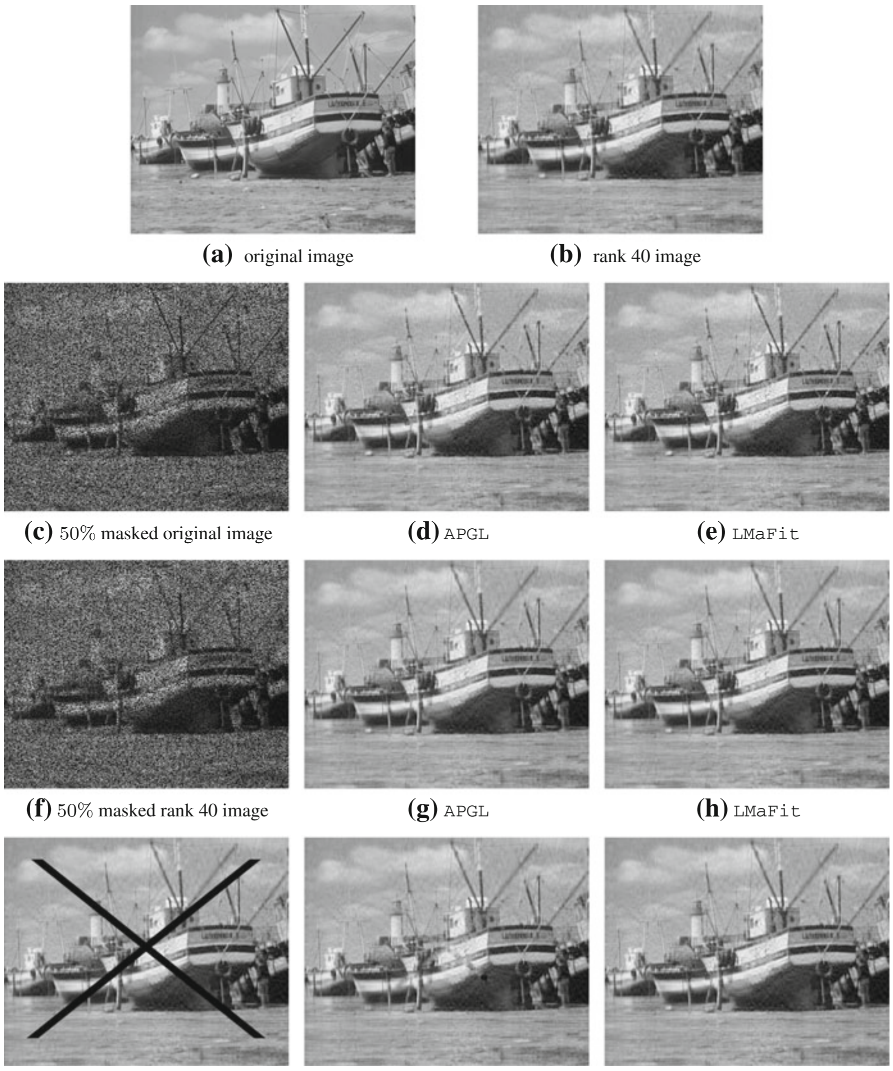
as the APGL solution. Finally, we comment that without proper rank restrictions, the jester problems do not appear to be good test problems for low-rank matrix approximation since the matrices to be approximated have only 100 columns to begin with. In fact, LMaFit with est_rank $= 1$ and K=100 was able to find "solutions" of rank 100 after one iteration whose NMAE is of order 10^{-16}.

4.5 Image and video denoising or inpainting

In this subsection we apply LMaFit and APGL to grayscale image denoising (similar to what was done in [24]) and to color video denoising of impulsive noise for visualizing solution quality. The task here is to fill in the missing pixel values of an image or video at given pixel positions that have been determined to contain impulsive noise. This process is also called inpainting, especially when the missing pixel positions are not randomly distributed. In their original forms, these problems are not true matrix completion problems, but matrix completion solvers can be applied to obtain low-rank approximations.

In the first test, the 512×512 original grayscale image is shown in Fig. 7a, and we truncated the SVD of the image to get an image of rank 40 in Fig. 7b. Figure 7c and f were constructed from Fig. 7a and b by sampling half of their pixels uniformly at random, respectively. Figure 7i was obtained by masking 6.34 % of the pixels of Fig. 74b in a non-random fashion. We set the parameters tol $= 10^{-3}$, est_rank $= 2$, $K = 20$ and max_rank $= 50$ for LMaFit, and tol $= 10^{-3}$, truncation $= 1$, truncation_gap $= 20$ and maxrank $= 50$ for APGL. The recovered images of Fig. 7c, f and i are depicted in Fig. 7d and e, g and h, and j and k, respectively. A summary of the computational results is shown in Table 6. In the table, rel.err denotes the relative error between the original and recovered images. From these figures and the table, we can see that LMaFit can recover the images as well as APGL can, but significantly faster.

Next, we apply LMaFit and APGL to fill in the missing pixels of a video sequence "xylophone.mpg" (available with the MATLAB Image Processing Toolbox). The video consists of p frames and each frame is an image stored in the RGB format, which is an m_r-by-n_r-by-3 cube. Here, $m_r = 240$, $n_r = 320$, and $p = 141$. The video was then reshaped into a $(m_r \times n_r)$-by-$(3 \times p)$, or 76,800-by-423, matrix



(a) original image **(b)** rank 40 image

(c) 50% masked original image **(d)** APGL **(e)** LMaFit

(f) 50% masked rank 40 image **(g)** APGL **(h)** LMaFit

(i) 6.34% masked rank 40 image **(j)** APGL **(k)** LMaFit

Fig. 7 Image denoising and inpainting

Table 6 Numerical results on image inpainting

Problem		APGL					LMaFit			
Image	r	μ	Iter	#sv	Time	Rel.arr	Iter	#sv	Time	Rel.arr
(c)	512	1.34e−02	33	50	4.81	8.73e−02	53	50	0.57	9.28e−02
(f)	40	1.34e−02	34	50	5.33	8.01e−02	45	40	0.47	7.94e−02
(i)	40	2.51e−02	32	50	4.97	9.07e−02	89	40	1.31	7.98e−02

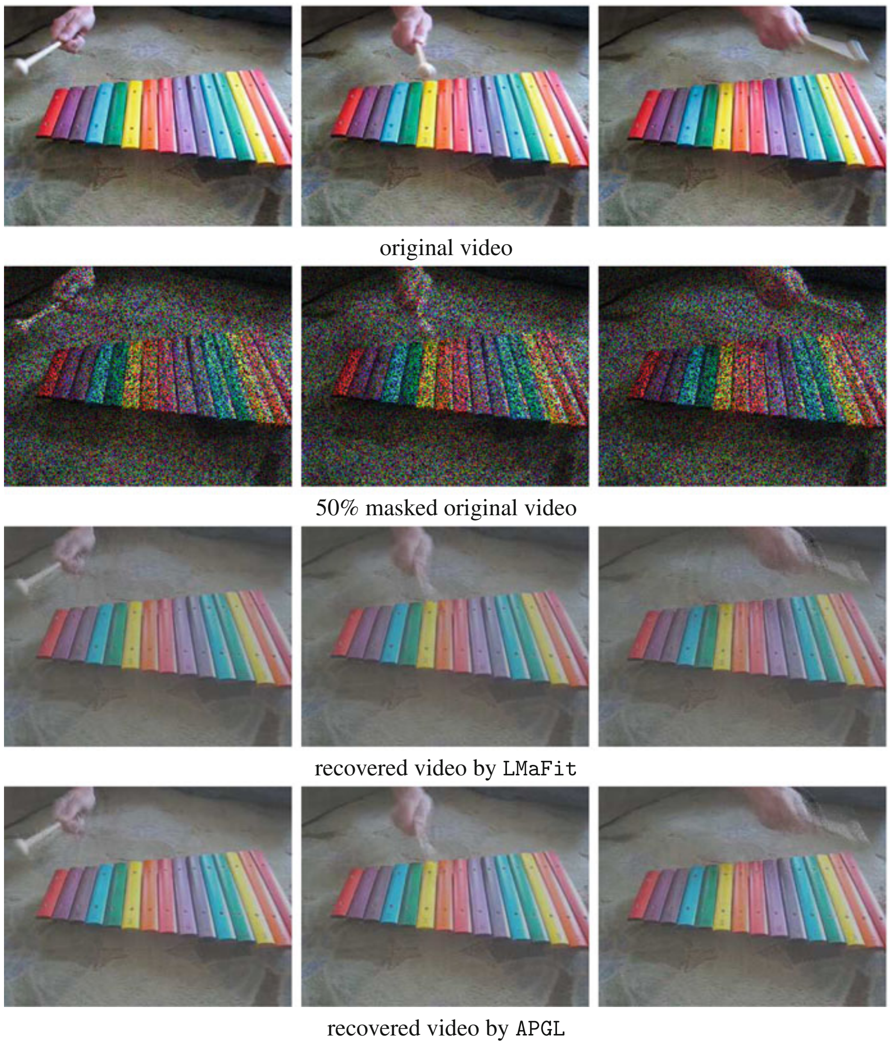


Fig. 8 Video denoising

M. We sampled 50% pixels of the video uniformly at random. Three frames of the original video and the corresponding 50 % masked images are shown in the first and second rows of Fig. 8, respectively. We set the parameters `tol` $= 10^{-3}$, $K = 20$, `rank_max` $= 80$ and `est_rank` $= 2$ for `LMaFit`, and `tol` $= 10^{-3}$, `truncation` $= 1$, `truncation_gap` $= 20$ and `maxrank` $= 80$ for `APGL`. A summary of computational results is presented in Table 7 and the recovered images are shown in the third and fourth rows of Figure 8. From these figures, we can see that `LMaFit` was able to restore the static part of the video quite successfully, and the moving part of the video was still recognizable. Table 7 shows that `APGL` obtained a slightly higher accuracy than `LMaFit` did, but the latter was approximately 4 times faster in reaching the same order of accuracy.

Table 7 Numerical results on video inpainting

Problem		APGL					LMaFit			
Video	m/n	μ	Iter	#sv	Time	Rel.arr	Iter	#sv	Time	Rel.arr
Xylophone	76800/423	3.44e+01	34	80	432.73	4.58e−02	64	75	91.63	4.93e−02

We emphasize again that the purpose of the above image/video denoising or inpainting experiments was to visualize the solution quality for the tested algorithms, rather than demonstrating the suitability of these algorithms for the tasks of denoising or inpainting.

4.6 Summary of computational results

We performed extensive computational experiments on two classes of problems: matrix completion and low-rank approximation. On the completion problems, our nonlinear SOR algorithm, coupled with the decreasing rank strategy, has shown good recoverability, being able to solve almost all tested problems as reliably as other solvers. We do point out that randomly generated matrix completion problems are numerically well-conditioned with high probability. On the other hand, any solver, including ours, can break down in the face of severe ill-conditioning. On low-rank approximation problems where the concept of rank can be numerically blurry and the quality of solutions less clear-cut, our nonlinear SOR algorithm, coupled with the increasing rank strategy, has demonstrated a capacity of producing solutions of competitive quality on a diverse range of test problems.

Our numerical results, especially those on matrix completion, have confirmed the motivating premise for our approach that avoiding SVD-related calculations can lead to a much accelerated solution speed for solving matrix completion and approximation problems. Indeed, in our tests `LMaFit` has consistently shown a running speed that is several times, ofter a couple of magnitudes, faster than that of other state-of-the-art solvers.

5 Conclusion

The matrix completion problems is to recover a low-rank matrix from a subset of its entries. It has recently been proven that, by solving a nuclear-norm minimization model, an incoherent low-rank matrix can be exactly recovered with high probability from a uniformly sampled subset of its entries as long as the sample size is sufficiently large relative to the matrix sizes and rank. In this paper, we study the approach of solving a low-rank factorization model for matrix completion. Despite the lack of a theoretical guarantee for global optimality due to model non-convexity, we have shown empirically that the approach is capable of solving a wide range of randomly generated matrix completion problems as reliably as solving the convex nuclear-norm minimization model. It remains a theoretical challenge to prove, or disprove, that under suitable conditions the low-rank factorization model can indeed solve matrix completion problems with high probability.

The main contribution of the paper is the development and analysis of an efficient nonlinear Successive Over-Relaxation (SOR) scheme that only requires solving a linear least-squares problem per iteration instead of a singular-value decomposition. The algorithm can be started from a rough over-estimate of the true matrix rank for completion problems, or started from a small initial rank (say, rank-1) for low-rank approximation problems. Extensive numerical results show that the algorithm can provide multi-fold accelerations over nuclear-norm minimization algorithms on a wide range of matrix completion or low-rank approximation problems, thus significantly extending our ability in solving large-scale problems in this area.

In order to solve large-scale and difficult problems, further research on rank estimation techniques is still needed to improve the robustness and efficiency of not only our algorithm, but also nuclear norm minimization algorithms that use partial singular value decompositions rather than full ones. Given the richness of matrix completion and approximation problems, different algorithms should be able to find usefulness in various areas of applications.

Acknowledgments We would like to thank Kim-Chuan Toh for the discussion on the code NNLS, and Sewoong Oh for the discussion on the code OptSpace. The authors are grateful to the Associate Editor and two anonymous referees for their detailed and valuable comments and suggestions.

References

1. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**, 183–202 (2009)
2. Candès, E., Plan, Y.: Matrix completion with noise. Proc. IEEE **98**, 925–936 (2010)
3. Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Found. Comput. Math. (2009)
4. Candès, E.J., Tao, T.: The power of convex relaxation near-optimal matrix completion. IEEE Trans. Inf. Theory **56**, 2053–2080 (2010)
5. Chan, T.F.: Rank revealing QR factorizations. Linear Algebra Appl. **88**(89), 67–82 (1987)
6. Chan, T.F., Hansen, P.C.: Low-rank revealing QR factorizations. Numer. Linear Algebra Appl. **1**, 33–44 (1994)
7. Dai, W., Milenkovic, O.: Set an algorithm for consistent matrix completion CoRR. abs/0909.2705 (2009)
8. Eldén, L.: Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms). Society for Industrial and Applied Mathematics, Philadelphia (2007)
9. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste A constant time collaborative filtering algorithm. Inf. Retr. **4**, 133–151 (2001)
10. Goldfarb, D., Ma, S., Wen, Z.: Solving low-rank matrix completion problems efficiently. In: Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing, Allerton'09, pp. 1013–1020 (2009)
11. Golub, G.H., Van Loan, C.F.: Matrix computations. In: Johns Hopkins Studies in the Mathematical Sciences, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
12. Grippo, L., Sciandrone, M.: On the convergence of the block nonlinear Gauss–Seidel method under convex constraints. Oper. Res. Lett. **26**, 127–136 (2000)
13. Gross, D.: Recovering low-rank matrices from few coefficients in any basis, tech. rep. Leibniz University (2009)
14. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 230–237. ACM, New York (1999)

15. Jian-Feng, C., Candes, E.J., Zuowei, S.: A singular value thresholding algorithm for matrix completion export. SIAM J. Optim. **20**, 1956–1982 (2010)
16. Keshavan, R.H., Montanari, A., Oh, S.: Matrix completion from a few entries. IEEE Trans. Inform. Theory **56**, 2980–2998 (2010)
17. Keshavan, R.H., Montanari, A., Oh, S.: Matrix completion from noisy entries. J. Mach. Learn. Res. **99**, 2057–2078 (2010)
18. Keshavan, R.H., Oh, S.: A gradient descent algorithm on the grassman manifold for matrix completion, tech. rep., Dept. of Electrical Engineering, Stanford University (2009)
19. Larsen, R.M.: PROPACK Software for large and sparse svd calculations. http://soi.stanford.edu/rmunk/PROPACK
20. Lee, K., Bresler, Y.: Admira atomic decomposition for minimum rank approximation. IEEE Trans. Inf. Theor. **56**, 4402–4416 (2010)
21. Liu, Y.-J., Sun, D., Toh, K.-C.: An implementable proximal point algorithmic framework for nuclear norm minimization. Math. Program., **133**(1–2), 399–436 (2011)
22. Liu, Z., Vandenberghe, L.: Interior-point method for nuclear norm approximation with application to system identification. SIAM J. Matrix Anal. Appl. **31**, 1235–1256 (2009)
23. Luo, Q.Z., Tseng, P.: On the convergence of the coordinate descent method for convex differentiable minimization. J. Optim. Theory Appl. **72**, 7–35 (1992)
24. Ma, S., Goldfarb, D., Chen, L.: Fixed point and bregman iterative methods for matrix rank minimization. Math. Program., **128**(1–2), 321–353 (2009)
25. Mazumder, R., Hastie, T., Tibshirani, R.: Regularization methods for learning incomplete matrices. Stanford University, tech. rep. (2009)
26. Meka, R., Jain, P., Dhillon, I.S.: Guaranteed rank minimization via singular value projection. CoRR, abs/0909.5457 (2009)
27. Morita, T., Kanade, T.: A sequential factorization method for recovering shape and motion from image streams. IEEE Trans. Pattern Anal. Mach. Intell. **19**, 858–867 (1997)
28. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Berlin (2006)
29. Recht, B.: A simpler approach to matrix completion. J. Mach. Learn. Res. (2010)
30. Recht, B., Fazel, M., Parrilo, P.A.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev. **52**, 471–501 (2010)
31. Shen, Y., Wen, Z., Zhang, Y.: Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization, tech. rep. Rice University (2010)
32. Stewart, G.W.: On the continuity of the generalized inverse. SIAM J. Appl. Math. **17**, 33–45 (1969)
33. Toh, K.-C., Yun, S.: An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Pacific J. Optim. **6**, 615–640 (2010)
34. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: a factorization method. Int. J. Comput. Vis. **9**, 137–154 (1992)
35. Tseng, P.: Dual ascent methods for problems with strictly convex costs and linear constraints: a unified approach. SIAM J. Control Optim. **28**, 214–242 (1990)
36. Tseng, P.: Convergence of a block coordinate descent method for nondifferentiable minimization. J. Optim. Theory Appl. **109**, 475–494 (2001)
37. Xu, Y., Yin, W., Wen, Z., Zhang, Y.: An alternating direction algorithm for matrix completion with nonnegative factors. tech. rep., Shanghai Jiaotong University (2011)
38. Yang, J., Yuan, X.: An inexact alternating direction method for trace norm regularized least squares problem. tech. rep., Dept. of Mathematics. Nanjing University (2010)
39. Yuan, X., Yang, J.: Sparse and low-rank matrix decomposition via alternating direction methods. tech. rep. Dept. of Mathematics, Hong Kong Baptist University (2009)
40. Zhang, Y.: LMaFit Low-rank matrix fitting (2009). http://www.caam.rice.edu/optimization/L1/LMaFit/
41. Zhang, Y.: An alternating direction algorithm for nonnegative matrix factorization. tech. rep. Rice University (2010)
42. Zhu, Z., So, A.M.-C., Ye, Y.: Fast and near-optimal matrix completion via randomized basis pursuit, tech. rep. Stanford University (2009)