Computational Optimization and Applications (2022) 81:67–90
https://doi.org/10.1007/s10589-021-00328-w

A Riemannian rank-adaptive method for low-rank matrix completion

Bin Gao[1] · **P.-A. Absil**[1]

Received: 26 March 2021 / Accepted: 27 October 2021 / Published online: 13 November 2021

Abstract
The low-rank matrix completion problem can be solved by Riemannian optimization on a fixed-rank manifold. However, a drawback of the known approaches is that the rank parameter has to be fixed a priori. In this paper, we consider the optimization problem on the set of bounded-rank matrices. We propose a Riemannian rank-adaptive method, which consists of fixed-rank optimization, rank increase step and rank reduction step. We explore its performance applied to the low-rank matrix completion problem. Numerical experiments on synthetic and real-world datasets illustrate that the proposed rank-adaptive method compares favorably with state-of-the-art algorithms. In addition, it shows that one can incorporate each aspect of this rank-adaptive framework separately into existing algorithms for the purpose of improving performance.

Keywords Rank-adaptive · Fixed-rank manifold · Bounded-rank matrices · Riemannian optimization · Low-rank · Matrix completion

1 Introduction

The low-rank matrix completion problem has been extensively studied in recent years; see the survey [11]. The matrix completion model based on a Frobenius norm minimization over the manifold of fixed-rank matrices is formulated as follows,

This work was supported by the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no. 30468160.

✉ Bin Gao
gaobin@lsec.cc.ac.cn

P.-A. Absil
pa.absil@uclouvain.be

[1] ICTEAM Institute, UCLouvain, 1348 Louvain-la-Neuve, Belgium

$$\begin{aligned} \min \ & f(X) := \frac{1}{2}\left\|P_\Omega(X) - P_\Omega(A)\right\|_{\mathrm{F}}^2 \\ \text{s.t. } & X \in \mathcal{M}_k := \{X \in \mathbb{R}^{m\times n} : \mathit{rank}(X) = k\}, \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ is a data matrix only known on a subset $\Omega \subset \{1,\dots,m\} \times \{1,\dots,n\}$, $k \le \min(m,n)$ is a given rank parameter and $P_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ denotes the projection onto Ω, i.e., $\left[P_\Omega(X)\right]_{i,j} = X_{i,j}$ if $(i,j) \in \Omega$, otherwise $\left[P_\Omega(X)\right]_{i,j} = 0$.

As mentioned in [17, §5.5], in many applications, the (unknown) singular values of A decay but do not become exactly zero; or A has low rank, but its rank is unknown. In both cases, A is known to admit good low-rank approximations, but the relation between the rank and the quality of the approximation is unknown, making it challenging to choose an adequate rank parameter k. If k is chosen too low, then the feasible set $\mathcal{M}_k$ of (1) only contains poor approximations of A, hampering an accurate completion. On the other hand, when k gets too large, the dimension of the feasible set grows, leading to an increase of space and time complexity, and eventually again to an inaccurate completion, now because of overfitting. A brief overview of existing strategies to choose k can be found in [3, Remark 5.1]. This context motivates the development of techniques that endeavor to strike a suitable balance between the above-mentioned concerns by adapting the rank as the computation progresses.

We consider the following model for low-rank matrix completion:

$$\begin{aligned} \min \ & f(X) \\ \text{s.t. } & X \in \mathcal{M}_{\le k} := \{X \in \mathbb{R}^{m\times n} : \mathit{rank}(X) \le k\}. \end{aligned} \tag{2}$$

Several algorithms based on Riemannian optimization (e.g., see [1]) for this problem have been developed in [12, 13, 15]. Recently, a Riemannian rank-adaptive method for low-rank optimization has been proposed in [19], and problem (2) can be viewed as a specific application. This rank-adaptive algorithm mainly consists of two steps: Riemannian optimization on the fixed-rank manifold and adaptive update of the rank. When a nearly rank-deficient point is detected, the algorithm reduces the rank to save computational cost. Alternatively, it increases the rank to gain accuracy. However, there are several parameters that users need to tune.

In this paper, we propose a new Riemannian rank-adaptive method (RRAM); see Algorithm 1. In comparison with the RRAM method in [19], we stray from convergence analysis concerns in order to focus on the efficiency of the proposed method for low-rank matrix completion. Specifically, the contributions are as follows.

- We adopt a Riemannian gradient method with non-monotone line search and Barzilai–Borwein step size to solve the optimization problem on the fixed-rank manifold (subsect. 3.2).
- By detecting the most significant gap of singular values of iterates, we propose a novel rank reduction strategy such that the fixed-rank problem can be restricted to a dominant subspace (subsect. 3.4). In addition, we propose a normal correction strategy to increase the rank (subsect. 3.3). Note that the existing algorithms

may benefit from these rank-adaptive mechanisms to improve their numerical performance.
– We demonstrate the effectiveness of the proposed method applied to low-rank matrix completion (Sect. 4). The numerical experiments on synthetic and real-world datasets illustrate that the proposed rank-adaptive method is able to find the ground-truth rank and compares favorably with other state-of-the-art algorithms in terms of computational efficiency.

The rest of paper is organized as follows. The next section introduces related work based on rank-update mechanisms, and presents necessary ingredients of the proposed method. In Sect. 3, a new Riemannian rank-adaptive method is proposed and its implementation details are also provided. Numerical experiments are reported in Sect. 4. The conclusion is drawn in Sect. 5.

2 Related work and preliminaries

In this section, we start with related work and give the preliminaries regarding the geometric aspect.

2.1 Related work

The feasible set $\mathcal{M}_k$ of problem (1) is a smooth submanifold of dimension $(m+n-k)k$ embedded in $\mathbb{R}^{m\times n}$; see [9, Example 8.14]. A Riemannian conjugate gradient (RCG) method for solving problem (1) has been proposed in [17], which efficiently assembles ingredients of RCG by employing the low-rank structure of matrices. There has been other methods for the fixed-rank optimization including the Riemannian trust-region method (RTR) [10] and the Riemannian gradient-descent method (RGD) [19]. Mishra et al. [10] have considered a trace norm penalty model for low-rank matrix completion and have proposed a method that alternately performs a fixed-rank optimization and a rank-one update.

However, $\mathcal{M}_k$ is not closed in $\mathbb{R}^{m\times n}$, hence $\min_{X\in\mathcal{M}_k} f(X)$ may not have a solution even when f is continuous and coercive; moreover, if a Riemannian optimization algorithm has a limit point of rank less than k, then the classical convergence results in Riemannian optimization (e.g., [4]) do not provide guarantees about the limit point. As a remedy, one can resort to the set of bounded rank matrices, i.e., $\mathcal{M}_{\leq k}$. Recently, algorithms for solving problem (2), combining the fixed-rank Riemannian optimization with a rank-increase update, have been introduced in [13, 15]. Basically, these methods increase the rank with a constant by searching along the tangent cone of $\mathcal{M}_{\leq k}$ and projecting onto $\mathcal{M}_k$ or $\mathcal{M}_{\leq k}$. In addition, a general projected line-search method on $\mathcal{M}_{\leq k}$ has been developed in [12] whose convergence guarantee is based on the assumption that limit points of algorithm have rank k.

The related work is further discussed in subsect. 3.5 after we introduce the necessary geometric ingredients for Riemannian optimization.

2.2 Geometry of $\mathcal{M}_{\leq k}$

The geometry of $\mathcal{M}_{\leq k}$ has been well studied in [12]. In this subsection, we introduce several Riemannian aspects that will be used in the rank-adaptive method.

Given the singular value decomposition (SVD) of fixed-rank matrices, an equivalent expression of the manifold $\mathcal{M}_k$ is

$$\mathcal{M}_k = \left\{ U\Sigma V^\top : \begin{array}{c} U \in \mathrm{St}(k,m), V \in \mathrm{St}(k,n), \\ \Sigma = diag(\sigma_1, \dots, \sigma_k) \text{ with } \sigma_1 \geq \cdots \geq \sigma_k > 0 \end{array} \right\},$$

where

$$\mathrm{St}(k,m) := \left\{ X \in \mathbb{R}^{m\times k} : X^\top X = I_k \right\}$$

denotes the (compact) Stiefel manifold, and a diagonal matrix with $\{\sigma_i\}$ on its diagonal is denoted by $diag(\sigma_1, \dots, \sigma_k)$. This expression of $\mathcal{M}_k$ provides a convenient way to assemble other geometric tools. For instance, the tangent space of $\mathcal{M}_s$ at $X \in \mathcal{M}_s$ is given as follows; see [17, Proposition 2.1]

$$\mathrm{T}_X\mathcal{M}_s = \left\{ \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} \mathbb{R}^{s\times s} & \mathbb{R}^{s\times(n-s)} \\ \mathbb{R}^{(m-s)\times s} & 0_{(m-s)\times(n-s)} \end{bmatrix} \begin{bmatrix} V & V_\perp \end{bmatrix}^\top \right\},$$

where $U_\perp \in \mathbb{R}^{m\times(m-s)}$ denotes a matrix such that $U^\top U_\perp = 0$ and $U_\perp^\top U_\perp = I$. Moreover, the normal space of $\mathcal{M}_s$ at X associated with the Frobenius inner product, $\langle X, Y\rangle := tr(X^\top Y)$, has the following form

$$\left(\mathrm{T}_X\mathcal{M}_s\right)^\perp = \left\{ \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} 0_{s\times s} & 0_{s\times(n-s)} \\ 0_{(m-s)\times s} & \mathbb{R}^{(m-s)\times(n-s)} \end{bmatrix} \begin{bmatrix} V & V_\perp \end{bmatrix}^\top \right\}. \tag{3}$$

Letting $P_U := UU^\top$ and $P_U^\perp := U_\perp U_\perp^\top = I - P_U$, the orthogonal projections onto the tangent space and normal space at X for $Y \in \mathbb{R}^{m\times n}$ are

$$\begin{aligned} \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(Y) &= P_U Y P_V + P_U^\perp Y P_V + P_U Y P_V^\perp, \\ \mathcal{P}_{(\mathrm{T}_X\mathcal{M}_s)^\perp}(Y) &= P_U^\perp Y P_V^\perp. \end{aligned} \tag{4}$$

The Riemannian gradient of f at $X \in \mathcal{M}_s$, denoted by $\mathrm{grad}_s f(X)$, is defined as the unique element in $\mathrm{T}_X\mathcal{M}_s$ such that $\langle \mathrm{grad}_s f(X), Z\rangle = \mathrm{D}f(X)[Z]$ for all $Z \in \mathrm{T}_X\mathcal{M}_s$, where $\mathrm{D}f(X)$ denotes the Fréchet derivative of f at X. It readily follows from [1, (3.37)] that

$$\mathrm{grad}_s f(X) = \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(\nabla f(X)),$$

where $\nabla f(X)$ denotes the Euclidean gradient of f at X.

When $s < k$, the tangent cone of $\mathcal{M}_{\leq k}$ at $X \in \mathcal{M}_s$ can be expressed as the orthogonal decomposition [12, Theorem 3.2]

$$\mathrm{T}_X\mathcal{M}_{\leq k} = \mathrm{T}_X\mathcal{M}_s \oplus \left(\mathrm{T}_X\mathcal{M}_s\right)^\perp_{\leq(k-s)}, \tag{5}$$

where $\oplus$ denotes the direct sum and

$$\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}_{\leq(k-s)} := \left\{ N \in \left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp} : rank(N) \leq (k-s) \right\}$$

is a subset of the normal space $\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}$. Furthermore, the projection onto the tangent cone has the form [12, Corollary 3.3]

$$\mathcal{P}_{\mathrm{T}_X\mathcal{M}_{\leq k}}(Y) \in arg\,min_{Z\in\mathrm{T}_X\mathcal{M}_{\leq k}} \|Y-Z\|_{\mathrm{F}} = \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(Y) + \mathcal{P}_{\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}_{\leq(k-s)}}(Y),$$

where $\mathcal{P}_{\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}_{\leq(k-s)}}(Y) \in \left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}_{\leq(k-s)}$ is a best rank-$(k-s)$ approximation of $\mathcal{P}_{\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}}(Y) = Y - \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(Y)$, i.e.,

$$\mathcal{P}_{\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}_{\leq(k-s)}}(Y) \in arg\,min_{rank(N)\leq(k-s)} \left\| Y - \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(Y) - N \right\|_{\mathrm{F}}. \tag{6}$$

Note that this projection is not unique when the singular values number $k-s$ and $k-s+1$ of $\mathcal{P}_{\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}}(Y)$ are equal. For simplicity, we denote

$$N_{k-s}(X) := \mathcal{P}_{\left(\mathrm{T}_X\mathcal{M}_s\right)^{\perp}_{\leq(k-s)}}(-\nabla f(X)), \tag{7}$$

and the projections of $-\nabla f(X)$ are denoted by

$$G_s(X) := \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(-\nabla f(X)) = -\mathrm{grad}_s f(X), \tag{8}$$

$$G_{\leq k}(X) := \mathcal{P}_{\mathrm{T}_X\mathcal{M}_{\leq k}}(-\nabla f(X)) = G_s(X) + N_{k-s}(X). \tag{9}$$

Consequently, the optimality condition of problem (2) can be defined as follows; see [12, Corollary 3.4].

Definition 1 $X^* \in \mathcal{M}_{\leq k}$ is called a *critical point* of optimization problem (2) if

$$\left\| G_{\leq k}(X) \right\|_{\mathrm{F}} = 0.$$

3 A Riemannian rank-adaptive method

We propose a new Riemannian rank-adaptive algorithm in this section. The Riemannian Barzilai–Borwein method with a non-monotone line search is proposed to solve the fixed-rank optimization problem. Rank increase and rank reduction strategies are developed.

3.1 Algorithmic framework

The proposed algorithmic framework for solving problem (2) is listed in Algorithm 1. Several comments are in order.

Algorithm 1: Riemannian rank-adaptive method (RRAM)

1 **Input:** $X_0 \in \mathcal{M}_k$, $p \leftarrow 1$, the update rank $s \leftarrow k$.
2 **Require:** rank reduction threshold $\Delta > 0$, rank increase parameter $\epsilon > 0$ and increase number l.
3 **Pre-process:** rank reduction $(\tilde{X}_0, s) \leftarrow (X_0, \Delta)$ by Algorithm 4.
4 **while** *stopping criteria are not achieved* **do**
5 Solve the fixed-rank optimization problem (11) on $\mathcal{M}_s$ by Algorithm 2 with the initial point $\tilde{X}_{p-1}$; obtain X_p.
6 **if** *large gap is detected* **then**
7 Rank reduction $(\tilde{X}_p, \tilde{r}) \leftarrow (X_p, \Delta)$ by Algorithm 4 and $s \leftarrow \tilde{r}$.
8 **else if** $s < k$ *and* $\|N_{k-s}(X_p)\|_{\mathrm{F}} > \epsilon \|G_s(X_p)\|_{\mathrm{F}}$ **then**
9 Rank increase $(\tilde{X}_p, s + \tilde{l}) \leftarrow (X_p, l)$ by Algorithm 3 and $s \leftarrow s + \tilde{l}$.
10 $p \leftarrow p + 1$.
11 **Output:** X_p with the rank s.

Algorithm 1 consists of the fixed-rank optimization and the rank adaptation. Except for its rank reduction feature, Algorithm 1 can be thought of as a descent method that aims at finding an approximate critical point of problem (2). In view of Definition 1 and (9), we consider the following measure of criticality at $X^* \in \mathcal{M}_{\leq k}$:

$$\left\|G_{\leq k}(X^*)\right\|_{\mathrm{F}}^2 = \left\|G_{s^*}(X^*)\right\|_{\mathrm{F}}^2 + \left\|N_{k-s^*}(X^*)\right\|_{\mathrm{F}}^2, \tag{10}$$

where X^* has rank s^* and the equality follows from the orthogonal decomposition (5) of $G_{\leq k}(X^*)$.

Algorithm 1 is initialized by a rank parameter k and an initial guess $X_0 \in \mathcal{M}_k$; see (17) for an initialization. After a pre-process on X_0 (line 3 of Algorithm 1), we obtain $\tilde{X}_0$ with the update (working) rank s; this process works favorably in numerical experiments (see Sect. 4).

During the iterations of Algorithm 1, line 5 aims for solving the fixed-rank optimization on $\mathcal{M}_s$, i.e.,

$$\min_{X \in \mathcal{M}_s} f(X), \tag{11}$$

by a globally convergent Riemannian optimization algorithm (Algorithm 2). Note that

$$\left\|G_s(X)\right\|_{\mathrm{F}} = \left\|\mathrm{grad}_s f(X)\right\|_{\mathrm{F}} = 0$$

is the first-order optimality condition of (11); see [1, §4.1].

Subsequently, we consider adapting the rank. On the one hand, if the obtained X_p from line 5 of Algorithm 1 has a large gap among singular values, we consider the working rank is too large, which leads to high space and time complexity. Thus, the rank is reduced by Algorithm 4. On the other hand, we check the condition in line 8 of Algorithm 1. Specifically, if $s = k$, we do not increase the rank; otherwise we check the condition

$$\left\|N_{k-s}(X_p)\right\|_{\mathrm{F}} > \epsilon \left\|G_s(X_p)\right\|_{\mathrm{F}}, \tag{12}$$

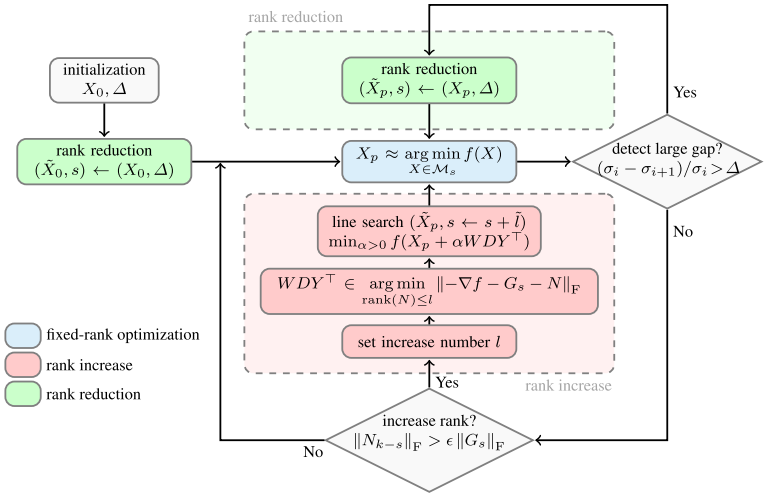


Fig. 1 Flowchart of the Riemannian rank-adaptive method (RRAM, Algorithm 1); see Algorithm 2 for the fixed-rank optimization, Algorithm 3 for the rank increase, and Algorithm 4 for the rank reduction

where $\epsilon > 0$ is a parameter that determines how much we crave for increasing the rank. If the condition (12) holds, in view of (10), it means that a significant part of $G_{\le k}(X_p)$ is in the normal space to $\mathcal{M}_s$, in which case we consider that the current rank s is too low, hampering an accurate completion. To this end, we increase the rank by Algorithm 3. Note that (12) is more apt to hold with smaller ϵ.

Note that Algorithm 1 endeavors to strike a balance between the rank and the quality of the completion by updating the rank adaptively. In order to have a more intuitive look at the rank-adaptive mechanism, Algorithm 1 is also presented as a flowchart in Fig. 1. There are three major parts in the framework: optimization on the fixed-rank manifold; rank increase; rank reduction. In the rest of this section, we introduce these features in detail.

3.2 Riemannian optimization on $\mathcal{M}_s$

Very recently, the Riemannian gradient method with non-monotone line search and Barzilai–Borwein (BB) step size [2] has been shown to be efficient in various applications; see [6–8]. In addition, its global convergence has been established (e.g., see [6, Theorem 5.7]). We adopt this method on the fixed-rank manifold $\mathcal{M}_s$ to address line 5 of Algorithm 1. Given the initial point $\tilde{X}_{p-1} \in \mathcal{M}_s$, the detailed method called RBB is listed in Algorithm 2.

Algorithm 2: Riemannian gradient method with non-monotone line search and Barzilai–Borwein step size (RBB)

1 **Input:** $X^{(0)} = \tilde{X}_{p-1} \in \mathcal{M}_s$.

2 **Require:** $\beta, \delta \in (0,1)$, $\theta \in [0,1]$, $0 < \gamma_{\min} < \gamma_{\max}$, $c_0 = f(X^{(0)})$, $q_0 = 1$, $\gamma_0 > 0$.

3 **for** $j = 0, 1, 2, \ldots, j_{\max}$ **do**

4 Choose $Z^{(j)} = -\mathrm{grad}_s f(X^{(j)})$.

5 If $j > 0$, choose a trial step size

$$\gamma_j = \frac{\langle S^{(j-1)}, S^{(j-1)} \rangle}{|\langle S^{(j-1)}, K^{(j-1)} \rangle|} \text{ for odd } j; \quad \gamma_j = \frac{|\langle S^{(j-1)}, K^{(j-1)} \rangle|}{\langle K^{(j-1)}, K^{(j-1)} \rangle} \text{ for even } j,$$

where

$$S^{(j-1)} = t_{j-1} \mathcal{T}_{X^{(j-1)} \to X^{(j)}}(Z^{(j-1)}),$$
$$K^{(j-1)} = \mathcal{T}_{X^{(j-1)} \to X^{(j)}}(Z^{(j-1)}) - Z^{(j)}.$$

Set $\gamma_j = \max(\gamma_{\min}, \min(\gamma_j, \gamma_{\max}))$.

6 Find the smallest integer h such that the non-monotone condition

$$f\left(\mathcal{P}_{\mathcal{M}_s}(X^{(j)} + \gamma_j \delta^h Z^{(j)})\right) \leq c_j + \beta \gamma_j \delta^h \left\langle \mathrm{grad}_s f(X^{(j)}), Z^{(j)} \right\rangle$$

holds. Set $t_j = \gamma_j \delta^h$.

7 Set $X^{(j+1)} = \mathcal{P}_{\mathcal{M}_s}(X^{(j)} + t_j Z^{(j)})$.

8 Update

$$q_{j+1} = \theta q_j + 1,$$
$$c_{j+1} = \frac{\theta q_j}{q_{j+1}} c_j + \frac{1}{q_{j+1}} f(X^{(j+1)}).$$

9 **Output:** Final iterate $X_p = X^{(j_0)}$.

In line 5 of Algorithm 2, we calculate a step size based on the Riemannian BB method [8], and the vector transport on $\mathcal{M}_s$ is defined as

$$\mathcal{T}_{X \to Y} : \mathrm{T}_X \mathcal{M}_s \to \mathrm{T}_Y \mathcal{M}_s, \quad Z \mapsto \mathcal{P}_{\mathrm{T}_Y \mathcal{M}_s}(Z).$$

The non-monotone line search is presented in line 6 and line 8 of Algorithm 2. The metric projection $\mathcal{P}_{\mathcal{M}_s}$ is chosen as the retraction map in line 7 of Algorithm 2, which can be calculated by a truncated SVD. Note that this projection is not necessarily unique, but we can always choose one in practice. All these calculations in Algorithm 2 can be efficiently achieved by exploiting the low-rank structure of $X^{(j)} = U \Sigma V^{\mathsf{T}}$. The interested readers are referred to [17] for detailed implementations.

We terminate Algorithm 2 when the maximum iteration number $j_{\max}$ is reached. In practice, the stopping criteria of Algorithm 1 (the relative errors introduced in Sect. 4) will also be checked during the iterations of Algorithm 2. Regarding the computational efficiency of Algorithm 2, it often compares favorably with other methods for fixed-rank optimization; see numerical examples in subsect. 4.1.

3.3 Rank increase

In order to increase the rank of X_p, we propose the "normal correction" step defined in Algorithm 3. The principle is to make an update along the normal vector $N_l(X_p)$ as defined in (6)-(7) provided $l \leq k - s$. Specifically, the normal correction steps consists of a line search along the direction of

$$N_l(X_p) = WDY^\top \in arg\,min_{rank(N)\leq l} \left\| -\nabla f(X_p) - G_s(X_p) - N \right\|_{\mathrm{F}} \tag{13}$$

such that $\tilde{l} \leq l$, $W \in \mathrm{St}(\tilde{l}, m)$, $Y \in \mathrm{St}(\tilde{l}, n)$, and $D \in \mathbb{R}^{\tilde{l}\times\tilde{l}}$ is a diagonal matrix that has full rank. The factors W, D and Y can be obtained by an l-truncated SVD of $-\nabla f(X_p) - G_s(X_p)$. The next result relates $WDY^\top$ to $N_l(X_p)$ and confirms that it yields a descent direction whenever it does not vanish.

Proposition 1 *Given $X = U\Sigma V^\top \in \mathcal{M}_s$, every best rank-$l$ approximation $WDY^\top$ of $-\nabla f(X) - G_s(X)$ satisfies that*

$$W^\top U = 0 \quad \text{and} \quad Y^\top V = 0.$$

More precisely, $WDY^\top$ belongs to $\mathcal{P}_{(\mathrm{T}_X\mathcal{M}_s)^\perp_{\leq l}}(-\nabla f(X))$ as defined in (6). Moreover, if $-\nabla f(X) - G_s(X) \neq 0$ (i.e., if $\nabla f(X) \notin \mathrm{T}_X\mathcal{M}_s$) , then $WDY^\top$ is a descent direction for f, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t} f(X + tWDY^\top)\Big|_{t=0} = -\|D\|_{\mathrm{F}}^2 < 0.$$

Proof It follows from (4) and (8) that

$$\begin{aligned} H(X) &:= -\nabla f(X) - G_s(X) = -\nabla f(X) - \mathcal{P}_{\mathrm{T}_X\mathcal{M}_s}(-\nabla f(X)) \\ &= \mathcal{P}_{(\mathrm{T}_X\mathcal{M}_s)^\perp}(-\nabla f(X)) = -P_U^\perp \nabla f(X) P_V^\perp \\ &= -U_\perp U_\perp^\top \nabla f(X) V_\perp V_\perp^\top. \end{aligned}$$

Therefore, any SVD of $H(X)$ has the form $U_\perp \bar{U}\bar{\Sigma}\bar{V}^\top V_\perp^\top$, where

$$\begin{aligned} \bar{U} &= [\bar{u}_1, \ldots, \bar{u}_r] \in \mathrm{St}(r, m-s), \\ \bar{V} &= [\bar{v}_1, \ldots, \bar{v}_r] \in \mathrm{St}(r, n-s), \\ \bar{\Sigma} &= diag(\bar{\sigma}_1, \ldots, \bar{\sigma}_r) \end{aligned}$$

with $\bar{\sigma}_1 \geq \cdots \geq \bar{\sigma}_r > 0$ and $r = rank(H(X))$. It follows that

$$H(X) = -U_\perp U_\perp^\top \nabla f(X) V_\perp V_\perp^\top = U_\perp \bar{U}\bar{\Sigma}\bar{V}^\top V_\perp^\top = \sum_{i=1}^{r} \bar{\sigma}_i \big(U_\perp \bar{u}_i\big)\big(V_\perp \bar{v}_i\big)^\top,$$

which is any compact SVD of $H(X)$. Note that $WDY^\top$ is a best rank-l approximation of $H(X)$ iff $WDY^\top$ is a l-truncated SVD of $\sum_{i=1}^{r} \bar{\sigma}_i \big(U_\perp \bar{u}_i\big)\big(V_\perp \bar{v}_i\big)^\top$, i.e., $\sum_{i=1}^{\tilde{l}} \bar{\sigma}_i \big(U_\perp \bar{u}_i\big)\big(V_\perp \bar{v}_i\big)^\top$ where $\tilde{l} := \min(l, r)$. Let

$$W = U_\perp[\bar{u}_1, \dots, \bar{u}_{\tilde{l}}],$$
$$Y = V_\perp[\bar{v}_1, \dots, \bar{v}_{\tilde{l}}],$$
$$D = diag(\bar{\sigma}_1, \dots, \bar{\sigma}_{\tilde{l}}).$$

It yields that $W^\top U = 0$ and $Y^\top V = 0$.

In addition, it follows from $U_\perp^\top \nabla f(X) V_\perp = -\bar{U}\bar{\Sigma}\bar{V}^\top$ that

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} f(X + tWDY^\top)\Big|_{t=0} &= \left\langle \nabla f(X), WDY^\top \right\rangle \\
&= \left\langle \nabla f(X), U_\perp \left(\sum_{i=1}^{\tilde{l}} \bar{\sigma}_i \bar{u}_i \bar{v}_i^\top \right) V_\perp^\top \right\rangle \\
&= \left\langle U_\perp^\top \nabla f(X) V_\perp, \sum_{i=1}^{\tilde{l}} \bar{\sigma}_i \bar{u}_i \bar{v}_i^\top \right\rangle \\
&= -\left\langle \bar{U}\bar{\Sigma}\bar{V}^\top, \sum_{i=1}^{\tilde{l}} \bar{\sigma}_i \bar{u}_i \bar{v}_i^\top \right\rangle \\
&= -\sum_{i=1}^{\tilde{l}} \bar{\sigma}_i^2 = -\|D\|_{\mathrm{F}}^2 < 0.
\end{aligned}$$

Thus, $WDY^\top$ is a descent direction. □

Hence, we can perform a line-search

$$\min_{\alpha > 0} f(X_p + \alpha WDY^\top).$$

As the objective function $f(X)$ in (1) is quadratic, this problem has the closed-form solution

$$\alpha^* = -\frac{\left\langle P_\Omega(WDY^\top), P_\Omega(X - A) \right\rangle}{\left\| P_\Omega(WDY^\top) \right\|_{\mathrm{F}}^2}. \tag{14}$$

Moreover, unless $\nabla f(X_p) \in T_X \mathcal{M}_s$, the normal correction step is guaranteed to increase the rank.

Proposition 2 *Let X and $WDY^\top$ be as in Proposition 1. Then, for all $\alpha \neq 0$, $X + \alpha WDY^\top$ has rank $s + \tilde{l}$, where $\tilde{l} = rank(D)$.*

Proof In view of Proposition 1, it readily follows from $W^\top U = 0$ and $Y^\top V = 0$ that

$$X + \alpha WDY^\top = U\Sigma V^\top + \alpha WDY^\top = \begin{bmatrix} U & W \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & \alpha D \end{bmatrix} \begin{bmatrix} V & Y \end{bmatrix}^\top$$

has rank $s + \tilde{l}$. □

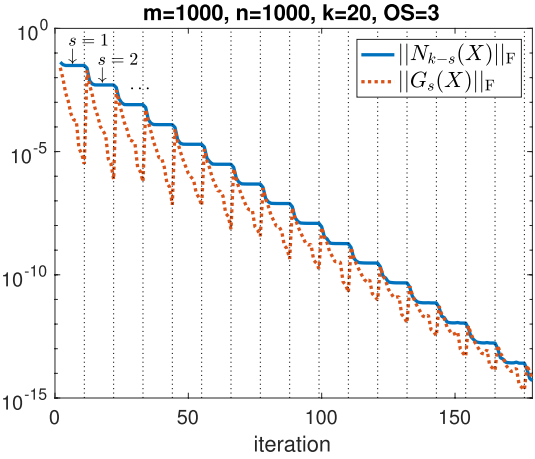


Fig. 2 Effect of the rank increase

We summarize these steps in Algorithm 3.

Algorithm 3: Rank increase

1 **Input:** $X_p = U\Sigma V^\top \in \mathcal{M}_s$, a rank increase number $l \leq k - s$.
2 Compute W, D and Y by (13).
3 Compute the step size α by (14).
4 Sort the columns of $[U\ W]$, $\left[\begin{smallmatrix}\Sigma & 0\\ 0 & \alpha D\end{smallmatrix}\right]$, $[V\ Y]$ by descending the diagonal entries of $\left[\begin{smallmatrix}\Sigma & 0\\ 0 & \alpha D\end{smallmatrix}\right]$, and accordingly obtain U_+, Σ_+, V_+.
5 **Output:** $\tilde{X}_p := U_+\Sigma_+V_+^\top$ with the rank $s + \tilde{l}$, where $\tilde{l} := \mathrm{rank}(\tilde{X}_p) - s$.

The effect of the rank increase is illustrated in Fig. 2. We consider a method that combines the fixed-rank optimization (RBB) with the rank-one increase. Figure 2 reports the evolution of $\|N_{k-s}(X)\|_F$ and $\|G_s(X)\|_F$ for solving problem (2) with a rank-one initial point. It is observed that when the stage of fixed-rank optimization is finished, $\|G_s(X)\|_F$ dramatically degrades while $\|N_{k-s}(X)\|_F$ does not. The peaks on the curve of $\|G_s(X)\|_F$ are caused by the change from $G_s(X_p)$ to $G_{s+1}(\tilde{X}_p)$.

3.4 Rank reduction

As $\mathcal{M}_s$ is not closed, an iterate sequence in $\mathcal{M}_s$ may have limit points of rank less than s. If the iterates are found to approach $\mathcal{M}_{\leq(s-1)}$, then it makes sense to solve the optimization problem on the set of smaller fixed-rank matrices. In addition, it can reduce the dimension of problem and thereby save the memory.

One possible strategy to decrease the rank has been proposed in [19]. Specifically, given $X_p = U\Sigma V^\top$ with $\Sigma = diag(\sigma_1, \ldots, \sigma_s)$ and $\sigma_1 \geq \cdots \geq \sigma_s > 0$, given a threshold $\Delta > 0$, we can replace the singular values smaller than $\sigma_1\Delta$ by zero. This rank reduction step returns a matrix $\tilde{X}_p \in \mathcal{M}_{\tilde{r}}$ by the best rank-$\tilde{r}$ approximation of X_p, where $\tilde{r} := \max\{i : \sigma_i \geq \sigma_1\Delta\}$.

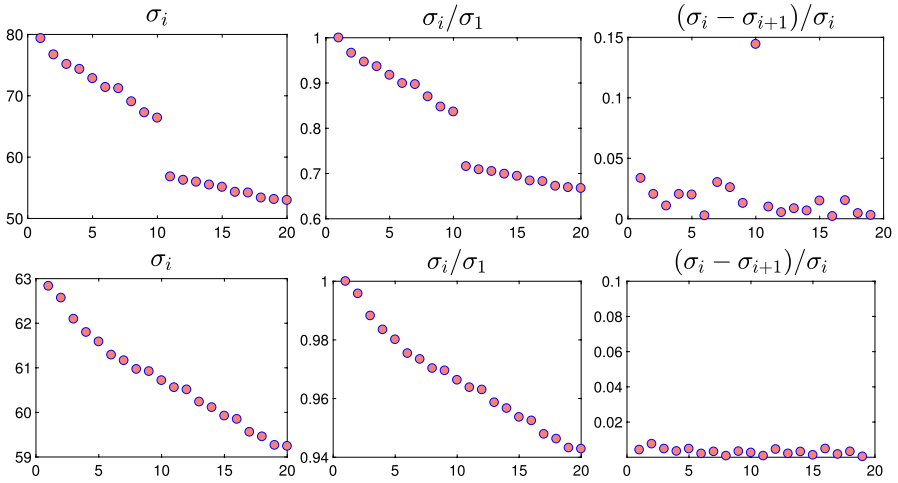


Fig. 3 Singular values of $X = \mathcal{P}_{\mathcal{M}_{20}}(P_\Omega(A))$. First row: $A = LR^\top \in \mathbb{R}^{1000\times1000}$, where $L, R \in \mathbb{R}^{1000\times10}$. Second row: $A = \texttt{randn}(1000, 1000) \in \mathbb{R}^{1000\times1000}$

In practice, we observe that the gap of singular values also plays an important role in fixed-rank optimization; see the numerical examples in subsect. 4.2. Therefore, we propose to check if there is a large gap in the singular values and, if so, to decrease the rank accordingly. To this end, we consider a criterion based on the relative change of singular values

$$\frac{\sigma_i - \sigma_{i+1}}{\sigma_i} > \Delta, \tag{15}$$

where $\Delta \in (0, 1)$ is a given threshold. Figure 3 presents two typical distributions of singular values, and σ_i/σ_1, $(\sigma_i - \sigma_{i+1})/\sigma_i$ are also computed. We observe that the large gap of singular values in the first row can be detected by (15) with setting $\Delta = 0.1$, while the condition $\sigma_i < \sigma_1\Delta$ is not activated.

In order to avoid losing too much information, we do not reduce the rank aggressively. The large gap of singular values is only detected by finding the index $\tilde{r}$ such that

$$\tilde{r} := \begin{cases} s, & \text{if } \max_i \left\{ \frac{\sigma_i - \sigma_{i+1}}{\sigma_i} \right\} \le \Delta; \\ arg\,max_i \left\{ \frac{\sigma_i - \sigma_{i+1}}{\sigma_i} \right\}, & \text{otherwise}. \end{cases} \tag{16}$$

In other words, if all the gaps $(\sigma_i - \sigma_{i+1})/\sigma_i$ are small, then we do not reduce the rank, i.e., $\tilde{r} = s$; otherwise, we choose the index that maximizes the gap $(\sigma_i - \sigma_{i+1})/\sigma_i$. A benefit of taking maximum of $(\sigma_i - \sigma_{i+1})/\sigma_i$ is to avoid aggressive rank reductions. For instance, given a threshold $\Delta = 0.1$, if singular values are distributed as $\{100, 80, 1, 0.8\}$, the condition (15) always holds. Nevertheless, we are not in favor of reducing the rank at the first gap (between 100 and 80), which is too

aggressive in this case. The condition (16) that takes maximal gap is able to return a suitable detection between 80 and 1.

The proposed rank reduction step is listed in Algorithm 4.

Algorithm 4: Rank reduction

1 **Input:** $X_p = U\Sigma V^\top \in \mathcal{M}_s$, $\Delta \in (0, 1)$.
2 Choose the index $\tilde{r}$ by (16).
3 Choose $U_{\tilde{r}}, V_{\tilde{r}}$ as the first $\tilde{r}$ columns of U, V, and set $\Sigma_{\tilde{r}} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\tilde{r}})$.
4 **Output:** $\tilde{X}_p := U_{\tilde{r}}\Sigma_{\tilde{r}}V_{\tilde{r}}^\top$ with the rank $\tilde{r}$.

Algorithm 4 is just one among many possible rank reduction strategies that retain the "largest" singular values and set the other ones to zero. The performance of those strategies is highly problem-dependent. Nevertheless, without such a strategy, the rank-adaptive method may miss opportunities to reduce the rank and thereby benefit from a reduced computational cost. Moreover, in order to address the issue, mentioned in subsect. 2.1, that $\mathcal{M}_k$ is not closed, some rank reduction mechanism is required to rule out sequences with infinitely many points in $\mathcal{M}_{>s}$ and a limit point in $\mathcal{M}_s$.

3.5 Discussion

In this subsection, we discuss the differences and improvements between the proposed rank-adaptive method and other rank-related algorithms. These methods with their corresponding features (fixed-rank algorithm, rank increase, and rank reduction) are listed in Table 1.

Note that the proposed rank-adaptive method in Algorithm 1 has several novel aspects. Firstly, as an inner iteration for the fixed-rank optimization, the RBB method with non-monotone line search proposed in [6] is applied to low-rank matrix completion. The numerical comparisons show that RBB tends to outperform other Riemannian methods such as RCG; see subsect. 4.1 and 4.4. Secondly, we search along the normal space to increase the rank, which is supported by Proposition 2. This contrasts with [19], where the update direction is obtained by projecting the antigradient onto a tangent cone. Moreover, in contrast with [15, §III.A], we do not assume the fixed-rank algorithm (Algorithm 2) to return a point X_p that satisfies $G_s(X_p) = 0$; however, if it does, then the proposed rank increase step coincides with the update $X_+ = X + \alpha G_{\leq(s+l)}(X)$ of [15] in view of (10). Thirdly, the proposed rank reduction mechanism differs from the one in [19]. It detects the most significant gap of singular values of iterates, while the rank reduction in [19] removes the singular values that are smaller than a threshold. The performance of the new rank reduction is illustrated in subsect. 4.2.

Table 1 Rank-related algorithms based on the geometry of the feasible set

ALGORITHM	FIXED-RANK	RANK	
		INCREASE	REDUCTION
[10]	RTR	$X_+ = X + \alpha w y^\top$	-
[12]	-	$X_+ = \mathcal{P}_{\mathcal{M}_{\leq k}}(X + \alpha G_{\leq k})$	-
[13]	RCG	$X_+ = \mathcal{P}_{\mathcal{M}_{s+l}}(X + \alpha G_{\leq (s+l)})$	-
[15]	RCG	$X_+ = X + \alpha G_{\leq (s+l)}$	-
[19]	RGD	$X_+ = \mathcal{P}_{\mathcal{M}_{\leq (s+l^*)}}(X + \alpha G_{\leq (s+l^*)})$	$\sigma_i/\sigma_1 < \Delta$
Algorithm 1	RBB	$X_+ = X + \alpha W D Y^\top$	$(\sigma_i - \sigma_{i+1})/\sigma_i > \Delta$

4 Numerical experiments

In this section, we first demonstrate the effectiveness of the proposed rank-adaptive algorithm, and then compare it with other methods on low-rank matrix completion problems. For simplicity, we restrict our comparisons to manifold-based methods since they usually perform well on this problem and are comparable with other low-rank optimization methods; see [17].

Unless otherwise mentioned, the low-rank matrix

$$A = LR^\top \in \mathbb{R}^{m\times n}$$

in (2) is generated by two random matrices $L \in \mathbb{R}^{m\times r}$, $R \in \mathbb{R}^{n\times r}$ with i.i.d. standard normal distribution. The sample set Ω is randomly generated on $\{1, \ldots, m\} \times \{1, \ldots, n\}$ with the uniform distribution of $|\Omega|/(mn)$. The oversampling rate (OS, see [17]) for A is defined as the ratio of $|\Omega|$ to the dimension of $\mathcal{M}_r$, i.e.,

$$\text{OS} := \frac{|\Omega|}{(m+n-r)r}.$$

Note that OS represents the difficulty of the completion problem, and it should be larger than 1.

The stopping criteria for different algorithms are based on the relative gradient and relative residual of their solutions, also the relative change of function value. Specifically,

$$\text{relative gradient} : \quad \frac{\left\|\text{grad}_s f(X)\right\|_{\text{F}}}{\max(1, \|X\|_{\text{F}})} < \epsilon_g,$$

$$\text{relative residual} : \quad \frac{\left\|P_\Omega(X) - P_\Omega(A)\right\|_{\text{F}}}{\left\|P_\Omega(A)\right\|_{\text{F}}} < \epsilon_\Omega,$$

$$\text{relative change} : \quad \left| 1 - \frac{\left\|P_\Omega(X_i) - P_\Omega(A)\right\|_{\text{F}}}{\left\|P_\Omega(X_{i-1}) - P_\Omega(A)\right\|_{\text{F}}} \right| < \epsilon_f.$$

Once one of the above criteria or the maximum iteration number 1000 is reached, we terminate the algorithms. Note that these criteria are introduced in [17]. The default tolerance parameters are chosen as $\epsilon_g = 10^{-12}$, $\epsilon_\Omega = 10^{-12}$, $\epsilon_f = 10^{-4}$. The rank increase parameter ϵ in (12) is set to 10, and the rank increase number l is 1. The rank reduction threshold Δ in (16) is set to 0.1. The inner maximum iteration number $j_{\max}$ is set to 100, γ_0 is computed by [17, Algorithm 5], and other parameters in Algorithm 2 are the same as those in [6]. Specifically, $\beta = 10^{-4}, \delta = 0.1, \theta = 0.85$, $\gamma_{\min} = 10^{-15}$, and $\gamma_{\max} = 10^{15}$.

All experiments are performed on a laptop with 2.7 GHz Dual-Core Intel i5 processor and 8GB of RAM running MATLAB R2016b under macOS 10.15.2. The code that produces the result is available from https://github.com/opt-gaobin/RRAM.

4.1 Comparison on the fixed-rank optimization

Before we test the rank-adaptive method, we first illustrate the performance of the RBB method proposed in Algorithm 2 on the fixed-rank optimization problem (1). We compare RBB with a state-of-the-art fixed-rank method called `LRGeomCG`[1] [17], which is a Riemannian CG method for fixed-rank optimization.

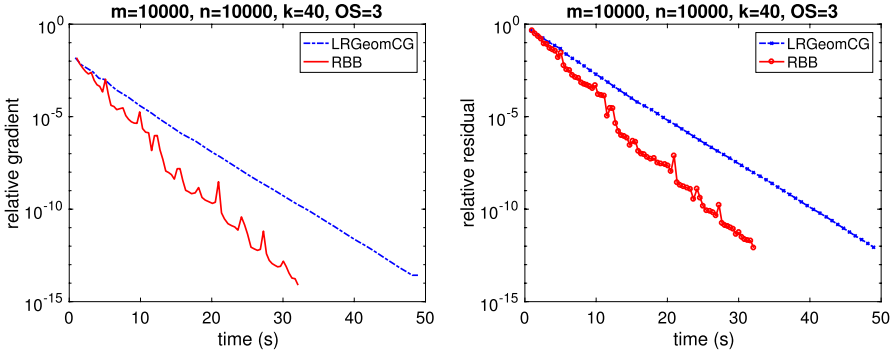
The problem is generated with $m = n = 10000, r = 40$ and OS $= 3$. The rank parameter k in (1) is set to $k = r$, which means the true rank of A is provided. The initial point is generated by

$$X_0 := \mathcal{P}_{\mathcal{M}_k}(P_\Omega(A)). \tag{17}$$

Figure 4 reports the numerical results. It is observed that the RBB method performs better than LRGeomCG in terms of computational efficiency to achieve comparable accuracy. In addition, one can observe the non-monotone property of RBB that stems from the non-monotone line search procedure in Algorithm 2.

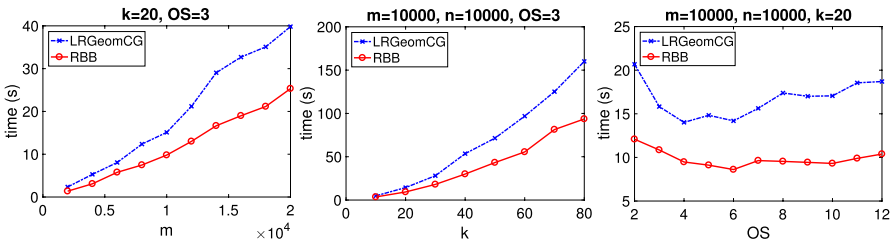
In order to investigate the performance of RBB on different problems, we test on three datasets with varying m, the rank parameter k, and OS, respectively. Specifically, we fix the oversampling rate OS $= 3$, $k = 20$, and chose $m = n$ from the set $\{2000j : j = 1, \ldots, 10\}$. Alternatively, we choose k from $\{10j : j = 1, \ldots, 8\}$ and fix $m = n = 10000$. In addition, the last dataset is varying OS from $\{1, \ldots, 10\}$, and

[1] Available from https://www.unige.ch/math/vandereycken/matrix_completion.html.



(a) Relative gradient **(b)** Relative residual

Fig. 4 A comparison on the fixed-rank optimization



(a) Test on varying m **(b)** Test on varying k **(c)** Test on varying OS

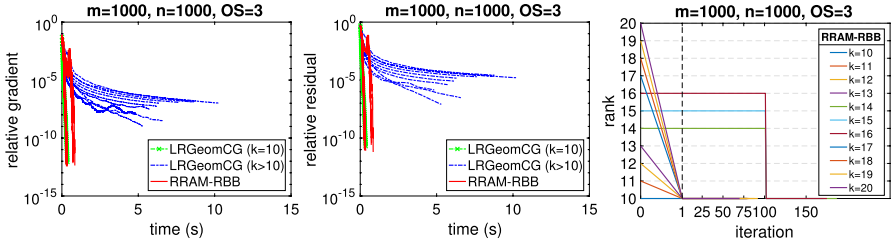
Fig. 5 A comparison on the fixed-rank optimization with varying m, k and OS

choosing $m = n = 10000$, $k = 20$. The running times of RBB and LRGeomCG are reported in Fig. 5. Notice that RBB has less running time than LRGeomCG when the size of problem (m and k) increases. Additionally, we observe that RBB outperforms LRGeomCG among all different oversampling settings.

4.2 Comparison on the rank reduction

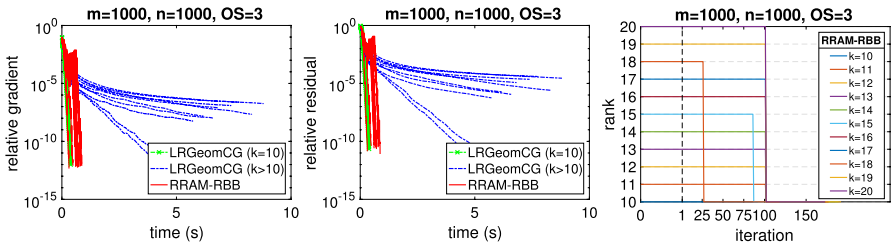
The effectiveness of the rank reduction step in RRAM is verified in this subsection. RRAM combines with the RBB method as the fixed-rank optimization, and we call it `RRAM-RBB`. For comparison, we also test LRGeomCG to illustrate that the rank-adaptive method is more suitable than fixed-rank methods for low-rank matrix completion. We generate problem (2) with $m = n = 1000$ and OS $= 3$. The data matrix $A = LR^\top$ is randomly generated by two rank 10 matrices. The following comparison is twofold based on different initial guesses that have similar singular value distributions in Fig. 3.

In a first set of experiments, the methods are initialized by (17), i.e., the best rank-k approximation of $P_\Omega(A)$. Given the rank parameter $k > rank(A) = 10$, the distribution of this type of initial points is similar to the one in the first row of Fig. 3, which



(a) Relative gradient **(b)** Relative residual **(c)** Update rank

Fig. 6 A comparison with different rank parameters k. The initial point is generated by $X_0 = \mathcal{P}_{\mathcal{M}_k}(P_\Omega(A))$



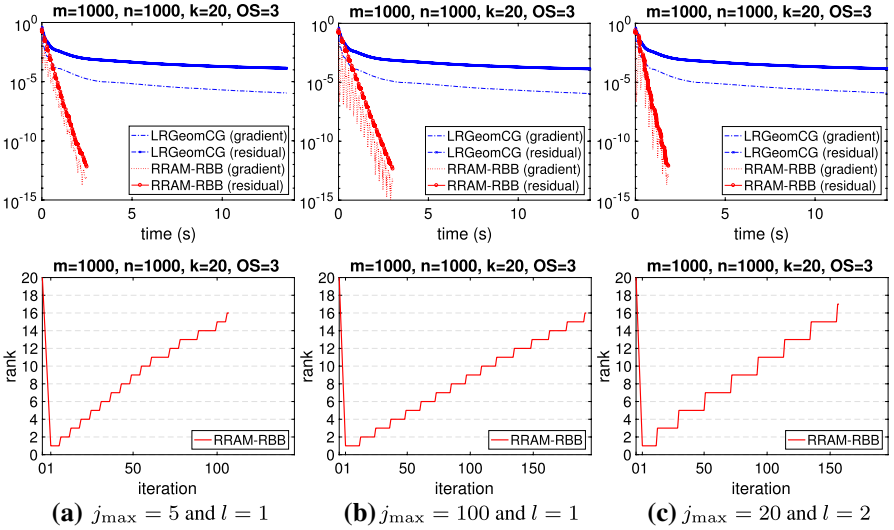
(a) Relative gradient **(b)** Relative residual **(c)** Update rank

Fig. 7 A comparison with different rank parameters k. The initial point is randomly generated

has a large gap of singular values. We make a test on different rank parameters k chosen from the set $\{10, 11, \ldots, 20\}$. The numerical results are presented in Fig. 6, and observations are summarized as follows.

- In Fig. 6a, b, it is observed that for LRGeomCG, the best choice of k is by far $k = 10$, which is the true rank of data matrix A. It reveals that the performance of the fixed-rank optimization method LRGeomCG highly depends on the choice of rank parameter, while the proposed rank-adaptive method has comparable results among all choices.
- The update rank of RRAM is listed in Fig. 6c. Notice that the rank reduction step is invoked in the initialization stage (line 3 of Algorithm 1) for most choices of the initial rank, and it reduces the rank to 10. In the cases of $k = 14, 15, 16$, although a initial rank reduction is not activated, the algorithm can detect the large gap of singular values when the first call of the fixed-rank method (Algorithm 2) terminates (at its maximum iteration number 100) and reduces k to the true rank 10.
- It is worth mentioning that in Fig. 6, when a rank reduction step completes, it often increases the function value at the very beginning, but the algorithm quickly converges once the true rank is detected.

Another class of initial points is randomly generated by a low-rank matrix $LR^\top$ that has rank k. It has a uniform singular values distribution that is the same as the second row of Fig. 3. Similarly, we compare RRAM-RBB with LRGeomCG on the



(a) $j_{\max} = 5$ and $l = 1$ **(b)** $j_{\max} = 100$ and $l = 1$ **(c)** $j_{\max} = 20$ and $l = 2$

Fig. 8 A comparison on the rank increase with three settings. First row: evolution of errors. Second row: update rank

problems with different rank parameters, and the results are reported in Fig. 7. We observe that RRAM-RBB can reduce the rank among all choices of $k > 10$ even when the singular values of the initial point do not have a large gap. Note that in the cases of $k = 15$ and 18, the first fixed-rank optimization stops with the iteration number less than 100 since the relative change is achieved.

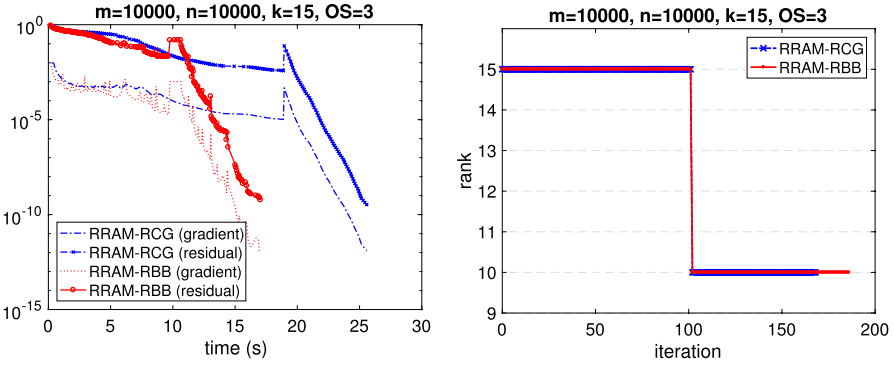
4.3 Comparison on the rank increase

In this subsection, we consider a class of problems for which the data matrix A is ill-conditioned. This type of problem has been numerically studied in [15]. Specifically, we construct

$$A = Udiag(1, 10^{-1}, \ldots, 10^{-r+1})V^{\top},$$

where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. Note that A has exponentially decaying singular values. We generate the problem with $m = n = 1000$, $k = r = 20$ and OS $= 3$. The initial point is generated by (17). We choose the rank increase parameter $\epsilon = 2$ such that RRAM-RBB is inclined to increase the rank. The tolerance parameter ϵ_g is set to 10^{-15}.

We test on three different settings: (I) the maximum iteration number $j_{\max}$ for the fixed-rank optimization is set to 5, and the rank increase number $l = 1$; (II) $j_{\max} = 100$ and $l = 1$; (III) $j_{\max} = 20$ and $l = 2$. Figure 8 reports the evolution of errors and the update rank of RRAM-RBB. The observations are as follows.



(a) Relative gradient and relative residual **(b)** Update rank

Fig. 9 An ablation comparison on the rank reduction

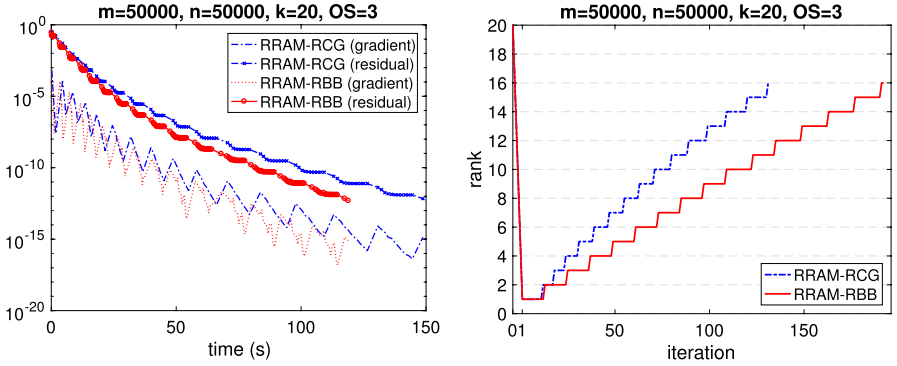
– In this ill-conditioned problem, RRAM-RBB performs better than the fixed-rank optimization method LRGeomCG ($k = 20$). In addition, we observe that the rank reduction step is invoked at the initial point for three settings, and RRAM-RBB increases the rank by a number l after each fixed-rank optimization.
– Note that the oscillation of relative gradient in RRAM-RBB stems from the rank increase step. From the first two columns of Fig. 8, it is observed that if the fixed-rank problem is inexactly solved ($j_{\max} = 5$), the performance of RRAM-RBB is still comparable with the "exactly" solved algorithm ($j_{\max} = 100$).

4.4 Ablation comparison on the proposed rank-adaptive mechanism

In this subsection, we produce an ablation study by incorporating Algorithm 1 into the fixed-rank optimization LRGeomCG [17] (Riemannian CG method). The resulting algorithm is called `RRAM-RCG`. Note that RRAM-RCG and RRAM-RBB differ only for the inner iteration (Algorithm 2). The purpose of this ablation study is to understand how the choice of the fixed-rank optimization method impacts the entire rank-adaptive framework.

In the first test, we compare RRAM-RCG with RRAM-RBB on problem instances generated as in subsect. 4.2, with the random initial guess described therein. Specifically, the problem is generated with $m = n = 10000$, $k = 15$, $r = 10$ and OS $= 3$. For both fixed-rank methods (RBB and RCG), the maximum iteration number $j_{\max}$ is set to 100. In Fig. 9, the numerical results illustrate that RCG still enjoys the benefit of the rank reduction step (16) that reduces the rank from 15 to the true rank 10. Moreover, it indicates that using RBB instead of RCG yields a considerable improvement on this problem instance. This observation can be explained by the comparison in subsect. 4.1.

Another test is generated as in subsect. 4.3 with $m = n = 50000$, $k = r = 20$ and OS $= 3$. Figure 10 reports the performance comparison of RRAM-RBB and RRAM-RCG. It shows that the proposed rank increase strategy is also effective for



(a) Relative gradient and relative residual **(b)** Update rank

Fig. 10 An ablation comparison on the rank increase

RCG. Notice that the performance of RRAM-RBB is slightly better than RRAM-RCG in terms of computational efficiency.

4.5 Test on real-world datasets

In this subsection, we evaluate the performance of RRAM on low-rank matrix completion with real-world datasets. The MovieLens[2] dataset contains movie rating data from users on different movies. In the following experiments, we choose the dataset MovieLens100K that consists of 100000 ratings from 943 users on 1682 movies, and MovieLens 1M that consists of one million movie ratings from 6040 users on 3952 movies.

For comparison, we test RRAM-RBB with several state-of-the-art methods that particularly target low-rank matrix completion, namely, LRGeomCG[1] [17] (Riemannian CG method), NIHT[3] and CGIHT[3] [18] (iterative hard thresholding algorithms), ASD[3] and ScaledASD[3] [14] (alternating steepest descent methods). Note that all these methods are based on the fixed-rank problem where the rank parameter has to be given a priori. For these two real-world datasets, we randomly choose 80% of the known ratings as the training set Ω and the rest as the test set Ω^c. The rank parameter k is set to 10 for all tested algorithms, and we terminate these algorithms once the maximum allowed running time (10s for MovieLens100K and 70s for MovieLens 1M) is reached or their own stopping criteria are achieved.

Figure 11 shows the singular values of the initial point (17), namely the rank-10 approximation of the zero-filled MovieLens dataset. It is observed that the largest gap can be detected between the first two singular values by the rank reduction (16) for both examples. According to the pre-process (line 3 of Algorithm 1), RRAM-RBB will

[2] Available from https://grouplens.org/datasets/movielens/.
[3] Available from http://www.sdspeople.fudan.edu.cn/weike/publications.html.

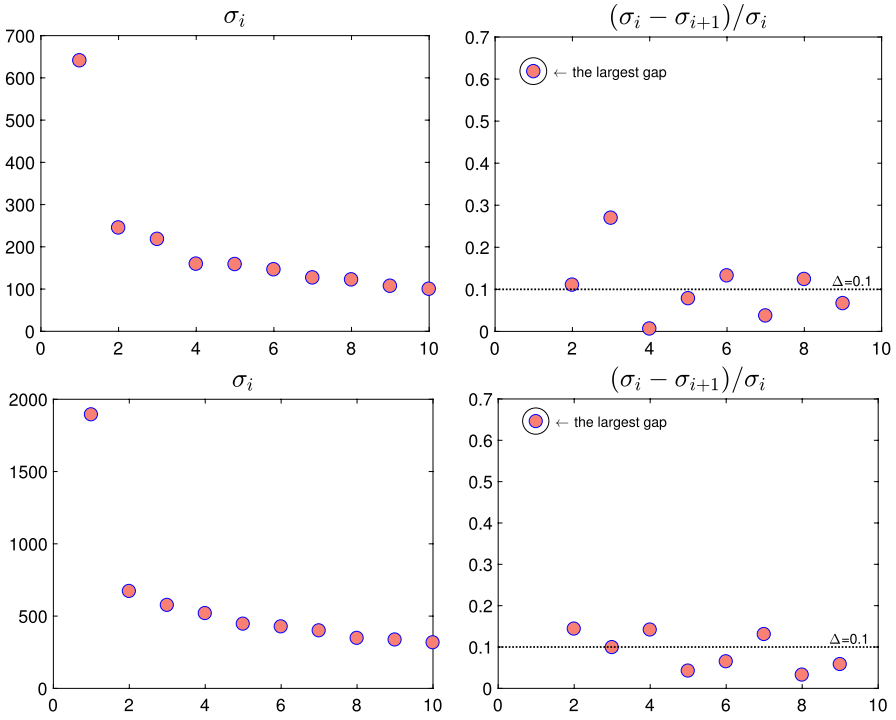
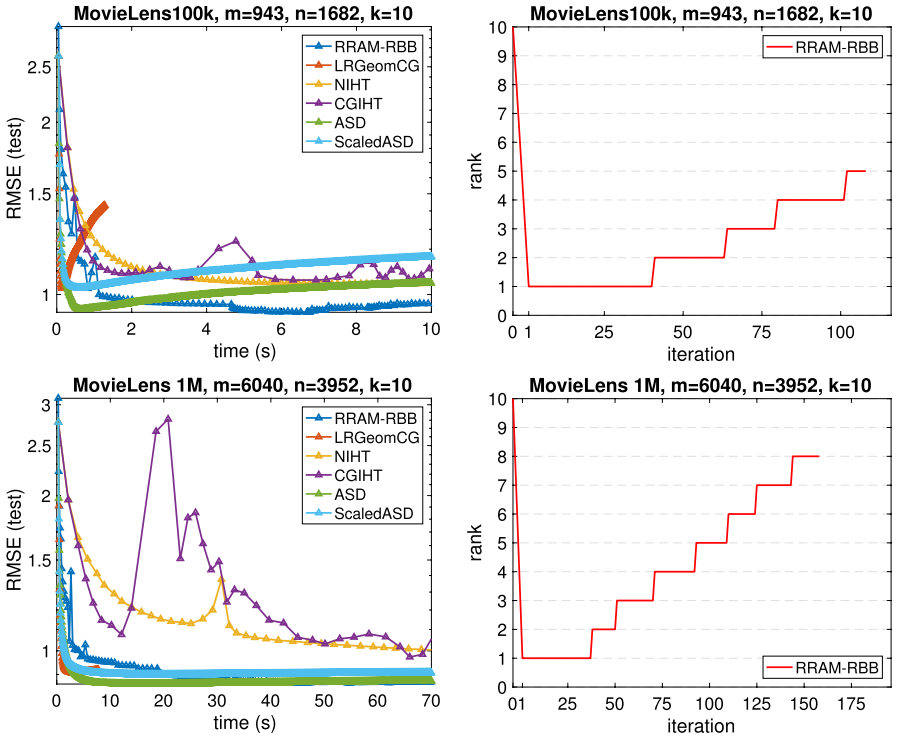


Fig. 11 Singular values of the initial points for the MovieLens dataset. First row: MovieLens100K. Second row: MovieLens 1M

thereby reduce the rank to one just after the initialization (17), which explains the first rank reduction in the following figures for RRAM-RBB.

The numerical results are illustrated in Fig. 12. The quality of matrix completion is evaluated by the root-mean-square error (RMSE) for a matrix X and a given index set Ω', i.e., $\mathrm{RMSE}(\Omega') := \|P_{\Omega'}(X) - P_{\Omega'}(A)\|_{\mathrm{F}}/\sqrt{|\Omega'|}$. The training RMSE and test RMSE are defined as $\mathrm{RMSE}(\Omega)$ and $\mathrm{RMSE}(\Omega^c)$, respectively. Note that RRAM-RBB achieves the best final RMSE (test) among all methods in the MovieLens100K dataset ($m = 943$, $n = 1682$), and is comparable with other algorithms in terms of computational efficiency. The evolution of update rank of RRAM-RBB shows that RRAM-RBB adaptively increases the rank and automatically finds a rank that is lower than the rank given to the other methods but with a smaller RMSE (test). In the larger dataset MovieLens 1M ($m = 6040$, $n = 3952$), RRAM-RBB still has a comparable RMSE (test). In summary, the rank-adaptive method accepts the flexible choices of rank parameter, and is able to search for a suitable rank.



(a) RMSE (test) **(b)** Update rank

Fig. 12 A comparison on real-world datasets. First row: MovieLens100K. Second row: MovieLens 1M

5 Conclusion

This paper concerns the low-rank matrix completion problem that can be modeled with a bounded-rank constraint. A Riemannian rank-adaptive method is proposed, featuring a Riemannian gradient method with non-monotone line search, a new rank increase strategy by searching along the normal space, and a new rank reduction scheme by detecting large gaps among singular values.

This paper explores the numerical behavior of rank-adaptive methods on synthetic and real-world problems. Although the value of the new rank-adaptive method is application dependent, our observations provide insight on the kind of data matrices for which rank-adaptive mechanisms play a valuable role. This suggests that the proposed method might also perform well on other low-rank optimization problems, such as those mentioned in [5, 12, 16, 19].

Acknowledgements We are grateful to the reviewers for helpful comments. We would like to thank Bart Vandereycken for helpful discussions on the code "LRGeomCG" and Shuyu Dong for providing his code of the comparison on real-world datasets.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Absil, P.-A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton (2008)
2. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA J. Numer. Anal. **8**(1), 141–148 (1988). https://doi.org/10.1093/imanum/8.1.141
3. Boumal, N., Absil, P.-A.: Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. Linear Algebra Appl. **475**, 200–239 (2015). https://doi.org/10.1016/j.laa.2015.02.027
4. Boumal, N., Absil, P.-A., Cartis, C.: Global rates of convergence for nonconvex optimization on manifolds. IMA J. Numer. Anal. **39**(1), 1–33 (2018). https://doi.org/10.1093/imanum/drx080
5. Chi, Y., Lu, Y.M., Chen, Y.: Nonconvex optimization meets low-rank matrix factorization: an overview. IEEE T. Signal Proces. **67**(20), 5239–5269 (2019). https://doi.org/10.1109/TSP.2019.2937282
6. Gao, B., Son, N.T., Absil, P.-A., Stykel, T.: Riemannian optimization on the symplectic Stiefel manifold. SIAM J. Optim. **31**(2), 1546–1575 (2021). https://doi.org/10.1137/20M1348522
7. Hu, J., Liu, X., Wen, Z.W., Yuan, Y.X.: A brief introduction to manifold optimization. J. Oper. Res. Soc. China (2020). https://doi.org/10.1007/s40305-020-00295-9
8. Iannazzo, B., Porcelli, M.: The Riemannian Barzilai-Borwein method with nonmonotone line search and the matrix geometric mean computation. IMA J. Numer. Anal. **38**(1), 495–517 (2018). https://doi.org/10.1093/imanum/drx015
9. Lee, J.M.: Introduction to Smooth Manifolds. Graduate Texts in Mathematics. Springer (2003). https://books.google.be/books?id=eqfgZtjQceYC
10. Mishra, B., Meyer, G., Bach, F., Sepulchre, R.: Low-rank optimization with trace norm penalty. SIAM J. Optim. **23**(4), 2124–2149 (2013). https://doi.org/10.1137/110859646
11. Nguyen, L.T., Kim, J., Shim, B.: Low-rank matrix completion: a contemporary survey. IEEE Access **7**, 94215–94237 (2019). https://doi.org/10.1109/ACCESS.2019.2928130
12. Schneider, R., Uschmajew, A.: Convergence results for projected line-search methods on varieties of low-rank matrices via Łojasiewicz inequality. SIAM J. Optim. **25**(1), 622–646 (2015). https://doi.org/10.1137/140957822
13. Tan, M., Tsang, I.W., Wang, L., Vandereycken, B., Pan, S.J.: Riemannian pursuit for big matrix recovery. pp. 1539–1547. PMLR, Bejing, China (2014). http://proceedings.mlr.press/v32/tan14.html
14. Tanner, J., Wei, K.: Low rank matrix completion by alternating steepest descent methods. Appl. Comput. Harmon. A. **40**(2), 417–429 (2016). https://doi.org/10.1016/j.acha.2015.08.003
15. Uschmajew, A., Vandereycken, B.: Greedy rank updates combined with Riemannian descent methods for low-rank optimization. In: 2015 International Conference on Sampling Theory and Applications (SampTA), pp. 420–424. IEEE (2015). https://doi.org/10.1109/SAMPTA.2015.7148925
16. Uschmajew, A., Vandereycken, B.: Geometric Methods on Low-Rank Matrix and Tensor Manifolds, pp. 261–313. Springer, Cham (2020)
17. Vandereycken, B.: Low-rank matrix completion by Riemannian optimization. SIAM J. Optim. **23**(2), 1214–1236 (2013). https://doi.org/10.1137/110845768
18. Wei, K., Cai, J.F., Chan, T.F., Leung, S.: Guarantees of Riemannian optimization for low rank matrix recovery. SIAM J. Matrix Anal. A. **37**(3), 1198–1222 (2016). https://doi.org/10.1137/15M1050525

19. Zhou, G., Huang, W., Gallivan, K.A., Van Dooren, P., Absil, P.-A.: A Riemannian rank-adaptive method for low-rank optimization. Neurocomputing **192**, 72–80 (2016). https://doi.org/10.1016/j.neucom.2016.02.030