

Article

Low-Rank Matrix Completion via QR-Based Retraction on Manifolds

Ke Wang ¹, Zhuo Chen ¹ , Shihui Ying ¹  and Xinjian Xu ^{2,*} ¹ Department of Mathematics, Shanghai University, Shanghai 200444, China² Qianweichang College, Shanghai University, Shanghai 200444, China

* Correspondence: xinjxu@shu.edu.cn

Abstract: Low-rank matrix completion aims to recover an unknown matrix from a subset of observed entries. In this paper, we solve the problem via optimization of the matrix manifold. Specially, we apply QR factorization to retraction during optimization. We devise two fast algorithms based on steepest gradient descent and conjugate gradient descent, and demonstrate their superiority over the promising baseline with the ratio of at least 24%.

Keywords: matrix completion; QR factorization; gradient algorithm; manifold

MSC: 90C29; 57R57; 49Q10; 90C30



Citation: Wang, K.; Chen, Z.; Ying, S.; Xu, X. Low-Rank Matrix Completion via QR-Based Retraction on Manifolds. *Mathematics* **2023**, *11*, 1155. <https://doi.org/10.3390/math11051155>

Academic Editors: Adrian Deaconu, Petru Adrian Cotfas and Daniel Tudor Cotfas

Received: 17 January 2023

Revised: 23 February 2023

Accepted: 23 February 2023

Published: 26 February 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The problem of matrix completion (MC) has generated a great deal of interest over the last decade [1], and several variant problems have been considered, such as non-negative matrix completion (NMC) [2], structured matrix completion [3,4] (including Hankel matrices [5]), and low-rank matrix completion (LRMC) [6,7]. Because of its wide applications in sensor network localization [8], system identification [9], machine learning [10,11], computer vision [12], recommendation systems [13], etc., LRMC has drawn a great deal of attention. Let $M \in \mathbb{R}^{n \times n}$ be an observed matrix and $\Omega \subset \{(i, j), i, j = 1, \dots, n\}$ be an index set of the observed position. Then, the desired low-rank matrix X can be recovered by solving the following rank minimization problem [14,15]:

$$\min_{X \in \mathbb{R}^{n \times n}} \text{rank}(X) \quad \text{s.t.} \quad P_{\Omega}(X) = P_{\Omega}(M), \quad (1)$$

where $[P_{\Omega}(X)]_{ij} = X_{ij}$, $(i, j) \in \Omega$ and 0, otherwise. Unfortunately, the calculation of the rank function is (non-deterministic polynomial) NP-hard, and thus all known algorithms need double exponential time on the dimension of n .

To overcome this limitation, many approaches have been proposed [13]. For instance, Candès and Recht [16] replaced the rank function with the nuclear norm, and (1) can be rewritten as

$$\min_X \|X\|_* \quad \text{s.t.} \quad P_{\Omega}(X) = P_{\Omega}(M), \quad (2)$$

where $\|X\|_* = \sum_i \sigma_i(X)$ and $\sigma_i(X)$ represents the i -th largest non-zero singular value. They proved that if the number of observed entries $m = |\Omega|$ obeys $m \geq cn^{1.2}r \log(n)$ with c being some positive constant and r being the rank of X , then most matrices of rank r can be perfectly recovered with very high probability by solving a simple convex optimization program. However, when the size of the matrix is large, the computation is still burdensome. To mitigate the computational burden, Cai et al. [17] introduced the singular value thresholding algorithm. The key idea of this approach is to place the regularization term into the objective function of the nuclear norm minimization problem.

On the other hand, given the rank of a matrix, Lee and Bresler [18] replaced the rank function with the Frobenius norm, and (1) can be rewritten as

$$\min_X \frac{1}{2} \|P_\Omega(M) - P_\Omega(X)\|_F^2 \quad \text{s.t.} \quad \text{rank}(X) \leq r. \quad (3)$$

According to matrix theory, the matrix $M \in \mathbb{R}^{n_1 \times n_2}$ of rank r can be decomposed into two matrices $X \in \mathbb{R}^{n_1 \times r}$ and $Y \in \mathbb{R}^{r \times n_2}$ such that $M = XY^T$. A straightforward method is to determine X and Y by minimizing the residual between the original M and the recovered one (that is, XY^T) on the sampling set [19,20]:

$$\min_{X,Y} \frac{1}{2} \|P_\Omega(M) - P_\Omega(XY^T)\|_F^2. \quad (4)$$

To solve this multiple objective optimization program, one can employ the alternating minimization technique: (i) set X first and determine Y via minimizing the residual and; (ii) fix Y and determine X in the same way.

To accelerate the completing process, a novel utilization of the rank information is definition of an inner product and a differentiable structure, which formulates a manifold-based optimization program [21,22]. Then, one can compute the Riemannian gradient and Hessian matrix to solve the following problem [14,23]:

$$\min_{X \in \mathcal{M}_r} \|P_\Omega(X - M)\|, \quad (5)$$

where $\mathcal{M}_r := \{X \in \mathbb{R}^{n_1 \times n_2}, \text{rank}(X) = r\}$. Specially, Mishra et al. [24] discussed singular value decomposition, rank factorization, and QR factorization on manifolds. Following this line, Cambier and Absil [25] simultaneously considered singular value decomposition and regularization:

$$\min_{X \in \mathcal{M}_r} \|P_\Omega(X - M)\|_{l_1} + \lambda \|P_\Omega(X)\|_F^2, \quad (6)$$

where $\|X\|_{l_1} = \sum_{i,j} |X_{i,j}|$ and λ is a regularization parameter. Yet, the improvement of the accuracy is not remarkable. More recently, Dong et al. [26] devised a preconditioned gradient descent algorithm for the rank factorization problem:

$$\min_{(G,H) \in \mathbb{R}^{m \times k} \times \mathbb{R}^{n \times k}} \frac{1}{2} \|P_\Omega(GH^T - M)\|_F^2, \quad (7)$$

which is a multiple objective problem on the product space that can be defined as a manifold. Although it shows good performance in comparison to single-objective problems, the algorithm hardly considers the structure on a per matrix basis.

In this paper, we consider QR factorization on manifolds. Different from single-objective optimization on the manifold [24,27,28], we study LRMC using multiple objective optimization in the product space $\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$. During iteration, we first obtain the gradient of the objective function in the tangent space and then retract it with QR factorization. Specially, we introduce a measure to characterize the degree of orthogonality of Q for retraction, based on which we design two fast algorithms and show their advantage in comparison to rank factorization [26].

The paper is organized as follows. In Section 2, we introduce some preliminaries, including basic notations, problem formulation, and the element of manifolds. In Section 3, we show algorithms based on the choice of initial point, descent direction, step size, and retraction. In Section 4, we prove convergence and analyze the reason why the proposed algorithms outperform those in [26]. In Section 5, we demonstrate the superior performance of the proposed algorithms using numerical experiments. Finally, in Section 6 we provide a conclusion.

2. Preliminary

Notation. The Euclidean inner product and norm for the product space $\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$, respectively, denoted with $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, are defined by

$$\langle x, y \rangle = \text{Tr}(Q_x^T Q_y) + \text{Tr}(R_x R_y^T) \quad \text{and} \quad \|x\| = \sqrt{\langle x, x \rangle}, \quad (8)$$

for any pair of points $x = (Q_x, R_x), y = (Q_y, R_y) \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$.

Problem statement. The purpose of this paper is to solve the problem (5). With QR factorization, it becomes:

$$\min_{(Q,R) \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}} f_{\Omega}(Q, R) := \frac{1}{2} \|P_{\Omega}(QR - M)\|_F^2. \quad (9)$$

QR factorization. QR factorization [29] can be carried out by using Householder transformation, Givens rotations, Gram–Schmidt process, and their variants. In this paper, we choose the modified Gram–Schmidt algorithm for a more reliable procedure. (see Algorithm 7 for details.)

Geometric element on $\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$. The tangent space (see Figure 1) at a point $x \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ is the finite Cartesian product of the tangent spaces of the two element matrix spaces. Then, $T_x(\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}) \simeq \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ (see Section 3.5.2, [21]) where $X \simeq Y$ indicates that there is a homeomorphism between the topological space X and Y .

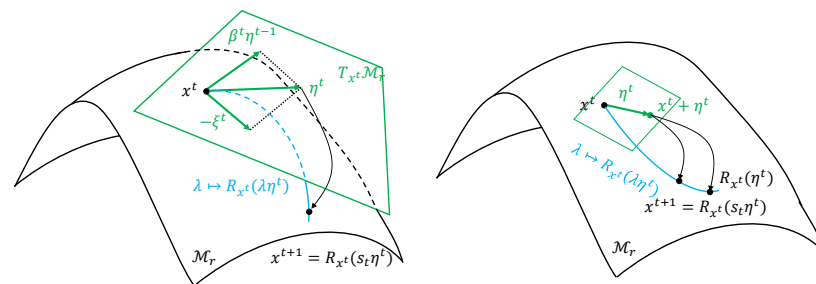


Figure 1. Illustration of the tangent space, the conjugate direction, and the retraction process.

Comparing the performance of several metrics, we consider the following. Given two tangent vectors $\xi, \eta \in T_x(\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2})$ (see Section 4, [26]) at point $x = (Q, R) \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$, the preconditioned metric is

$$M_x(\xi, \eta) = \text{Tr}(\xi_Q^T \eta_Q (RR^T + \delta I_r)) + \text{Tr}(\xi_R \eta_R^T (1 + \delta)), \quad (10)$$

where $\xi = (\xi_Q, \xi_R), \eta = (\eta_Q, \eta_R), \delta > 0$ is a constant, which keeps the metric well defined and positive definite if Q or R does not have full rank. Furthermore, if $\xi = \eta$, one can write $\|\xi\|_x^2 = M_x(\xi, \xi)$ as a kind of norm at point x .

Definition 1. For a point $x \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$, the gradient of f_{Ω} is the unique vector in $T_x(\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2})$, denoted with $\nabla f_{\Omega}(x)$, such that

$$M_x(\xi, \nabla f_{\Omega}(x)) = Df_{\Omega}(x)[\xi], \forall \xi \in T_x(\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}), \quad (11)$$

where $Df_{\Omega}(x)[\xi] = \text{Tr}(\xi_Q^T P_{\Omega}(QR - M)R^T) + \text{Tr}(\xi_R P_{\Omega}^T(QR - M)Q)$ is directional derivative defined [21] by

$$Df_{\Omega}(x)[\xi] = \lim_{t \rightarrow 0} \frac{f_{\Omega}(x + t\xi) - f_{\Omega}(x)}{t}.$$

Combining Equations (8) and (11), it follows that

$$\nabla f_{\Omega}(Q, R) = (\partial_Q f_{\Omega}(Q, R)(RR^T + \delta I_r)^{-1}, \partial_R f_{\Omega}(Q, R)(1 + \delta)^{-1}), \quad (12)$$

where $\partial_Q f_\Omega(Q, R) = P_\Omega(QR - M)R^T$ and $\partial_R f_\Omega(Q, R) = Q^T P_\Omega(QR - M)$.

3. Algorithms

Initial point x^0 . Following the widely used spectral initialization [30], we apply k -SVD to a zero-filled matrix $M_0 := P_\Omega(M)$ and yield three matrices U_0, Σ_0 , and V_0 such that $M_0 = U_0 \Sigma_0 V_0^T$. Then, the initial point $x^0 := (Q^0, R^0)$ is set as (see Algorithm 1 for details)

$$(Q^0, R^0) = (U_0 \Sigma_0^{1/2}, \Sigma_0^{1/2} V_0^T). \quad (13)$$

Algorithm 1 Initialization

Input: data M and rank r

Output: initialization x_0

- 1: Use singular value decomposition (SVD) to compute U, Σ, V (that satisfies $M = U \Sigma V^T$).
 - 2: Trim matrices $U_0 = U(:, 1:r), \Sigma_0 = \Sigma(1:r, 1:r), V_0 = V(:, 1:r)$.
 - 3: Set $x_0 = [U_0(\Sigma_0)^{1/2}; V_0(\Sigma_0^{1/2})^T]$.
-

Descent direction η^t . Here, we consider two kinds of directions, the steepest descent (SD) direction (see Algorithm 2) and the conjugate descent (CD) direction (see Algorithm 3 and Figure 1), defined, respectively, by

$$\eta^t = -\nabla f(x^t), \quad (14)$$

$$\eta^t = -\nabla f(x^t) + \beta_t \eta^{t-1}. \quad (15)$$

Although there are several calculations of β_t , we adopt $\beta_t^{DY} = \|\xi^t\|_x^2 / M_x(\eta^{t-1}, \Delta \xi^{t-1})$ from [31] because it outperforms the others.

Algorithm 2 Steepest descent (SD) direction of the function in (9) with orthogonality of Q

Input: Data M , iterate $x_t = [Q; R^T]$, rank r , and metric constant δ .

Output: SD direction ξ^t

- 1: $S = P_\Omega(QR - M)$.
 - 2: $\nabla f_\Omega(x^t) = (SR^T / (RR^T + \delta I_r), S^T Q / (1 + \delta)) (= \nabla f, \text{ for omitted})$.
 - 3: $\xi^t = -\nabla f$
-

Algorithm 3 Conjugate descent (CD) direction of the function in (9)

Input: Last conjugate direction η^{t-1} (Set $\eta^0 = 0$), conjugate direction β_t .

Output: Conjugate direction η^t .

- 1: Compute ∇f using Algorithm 2.
 - 2: $\eta^t = -\nabla f + \beta_t \eta^{t-1}$.
-

Stepsize s_t . For the SD direction, we apply exact line search (ELS) [22] (see Algorithm 4). Let $\eta = (\eta_Q, \eta_R) \in T_x(\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2})$ be a given descent direction; then,

$$\begin{aligned} & \arg \min_s f_\Omega(Q + s\eta_Q, R + s\eta_R) \\ &= \arg \min_s \frac{1}{2} \|P_\Omega((Q + s\eta_Q)(R + s\eta_R) - M)\|_F^2 \\ &= \arg \min_s \frac{1}{2} \|P_\Omega(\eta_Q \eta_R) s^2 + P_\Omega(Q \eta_R + \eta_Q R) s + P_\Omega(QR - M)\|_F^2 \\ &= \arg \min_s \frac{1}{2} \text{Tr}[(As^2 + Bs + O)^T (As^2 + Bs + O)] \\ &= \arg \min_s \frac{1}{2} [\text{Tr}(A^T A) s^4 + 2\text{Tr}(A^T B) s^3 + (2\text{Tr}(A^T O) + \text{Tr}(B^T B)) s^2 + 2\text{Tr}(B^T O) s + \text{Tr}(O^T O)], \end{aligned}$$

where $A = P_{\Omega}(\eta_Q \eta_R)$, $B = P_{\Omega}(Q \eta_R + \eta_Q R)$, and $O = P_{\Omega}(QR - M)$. The differential of the formula above reads as:

$$2\text{Tr}(A^T A)s^3 + 3\text{Tr}(A^T B)s^2 + (2\text{Tr}(A^T O) + \text{Tr}(B^T B))s + \text{Tr}(B^T O) = 0. \quad (16)$$

As a cubic equation, one can obtain its roots easily. The step size s_t is exactly the real positive root.

Algorithm 4 Exact line search

Input: Data M , iterate $x = [Q; R^T]$, Conjugate direction $\eta = [\eta_Q; \eta_R^T]$.

Output: Step size s .

- 1: Set $A = P_{\Omega}(\eta_Q \eta_R)$, $B = P_{\Omega}(Q \eta_R + \eta_Q R)$ and $O = P_{\Omega}(QR - M)$.
 - 2: Furthermore, solve the cubic equation $2\text{Tr}(A^T A)x^3 + 3\text{Tr}(A^T B)x^2 + (2\text{Tr}(A^T O) + \text{Tr}(B^T B))x + \text{Tr}(B^T O) = 0$.
 - 3: Let the smallest absolute value s of their solutions be the step size.
-

For the CD direction, we apply the inexact line search (IELS) [32] (see Algorithm 5). For this purpose, we set $s_0 = -M_x(\xi^t, \eta^t) / (L \|\eta^t\|_x^2)$, where $L > 0$ is constant, and $\sigma \in (0, 0.5]$. Then, the step size s_t at the t -th iteration is the largest one in the set $\{s_0, 0.5s_0, 0.5^2s_0, \dots\}$, and therefore

$$f(x^t) - f(x^t + s_t \eta^t) \geq -\sigma s_t M_x(\xi^t, \eta^t). \quad (17)$$

Algorithm 5 Inexact line search

Input: Data M , iterate x , constant $L > 0$, times limitation im , parameter $\sigma \in (0, 0.5]$, SD direction ξ , and CD direction η .

Output: Stepsize s .

- 1: Set $s_0 = -M_x(\xi, \eta) / (L \|\eta\|_x^2)$.
 - 2: $k \leftarrow 0$
 - 3: **while** $f_{\Omega}(x) - f_{\Omega}(x + s\eta) + \sigma s M_x(\xi, \eta) < 0$ **&&** $k < im$ **do**
 - 4: $s \leftarrow 0.5s$
 - 5: $k \leftarrow k + 1$
 - 6: **end while**
-

Retraction. With the descent direction $\eta = (\eta_Q, \eta_R)$ and stepsize s , one can apply retraction (see Figure 1 and Algorithm 6). For this purpose, we introduce the concept of the degree of orthogonality.

Definition 2. For a matrix $Q \in \mathbb{R}^{n_1 \times r}$, we definite its degree of orthogonality as:

$$\text{Orth}(Q) = \left| \frac{\text{Tr}(Q^T Q) - r}{r} \right|. \quad (18)$$

Algorithm 6 Retraction with QR factorization

Input: Iteration $x = [Q; R^T]$, direction $\eta = [\eta_Q; \eta_R^T]$, stepsize s_t and parameter θ .

Output: Next iterate $x^{t+1} = (Q^{t+1}, R^{t+1})$.

- 1: **if** $\text{Orth}(Q^t + s_t \eta_Q^t) < \theta$ (see (18)) **then**
 - 2: $x^{t+1} = x^t$
 - 3: **else** {obtain $\tilde{Q}$ and $\tilde{R}$ from $Q^t + s_t \eta_Q^t$ using Algorithm 7}
 - 4: $x^{t+1} = [\tilde{Q}; ((R^t)^T + s_t (\eta_R^t)^T)(\tilde{R})^T]$
 - 5: **end if**
-

Algorithm 7 Modified Gram–Schmidt algorithm**Input:** $A \in \mathbb{R}^{n_1 \times r}$ with $\text{rank}(A) = r$.**Output:** $Q \in \mathbb{R}^{n_1 \times r}$ and $R \in \mathbb{R}^{r \times r}$.

```

1: for  $k = 1 : r$  do
2:    $R_{k,k} = \sqrt{\sum_{i=1}^{n_1} A_{i,k}^2}$ 
3:   for  $i = 1 : n_1$  do
4:      $Q_{i,k} = A_{i,k} / R_{k,k}$ 
5:   end for
6:   for  $k = k + 1 : r$  do
7:      $R_{k,j} = \sum_{i=1}^{n_1} Q_{i,k} A_{i,j}$ 
8:     for  $i = 1 : n_1$  do
9:        $A_{i,j} = A_{i,j} - Q_{i,k} R_{k,j}$ 
10:    end for
11:  end for
12: end for

```

Given a small parameter θ , we say that the matrix $Q + s\eta_Q$ has *good orthogonality* if $\text{Orth}(Q + s\eta_Q) < \theta$. Then, we adopt $(Q + s\eta_Q, R + s\eta_R)$ as the value for the next iterate. On the contrary, we have to decompose $Q + s\eta_Q$ [33] and obtain $\tilde{Q}$, $\tilde{R}$, and hence, the next iteration point $(\tilde{Q}, \tilde{R}(R + s\eta_R))$.

In summary, we present Algorithms 8 and 9 as the whole process of solving the optimization problem (9), respectively.

Algorithm 8 QR Riemannian gradient descent (QRRGD)**Input:** Function $f : \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2} \rightarrow \mathbb{R}$ (see (9)), initial point $x^0 \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ (generated by Algorithm 1), tolerance parameter $\epsilon > 0$ **Output:** x^f

```

1:  $t \leftarrow 0$ 
2: Compute the gradient by Algorithm 2
3: while  $\|\zeta^t\|_x > \epsilon$  do
4:   Find step size  $s_t$  by Algorithm 4
5:   Update via retraction (Algorithm 6):  $x^{t+1} = R_{x^t}(s_t \eta^t)$ 
6:    $t \leftarrow t + 1$ 
7:   Compute the steepest direction by Algorithm 2
8: end while

```

Algorithm 9 QR Riemannian conjugate gradient (QRRCG)**Input:** Function $f : \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2} \rightarrow \mathbb{R}$ (see (9)), initial point $x^0 \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$, tolerance parameter $\epsilon > 0$ **Output:** x^f

```

1:  $t \leftarrow 0$ 
2: Compute the gradient using Algorithm 2
3: while  $\|\zeta^t\|_x > \epsilon$  do
4:   Find step size  $s_t$  using Algorithm 4
5:   Update via retraction (Algorithm 6):  $x^{t+1} = R_{x^t}(s_t \eta^t)$ 
6:    $t \leftarrow t + 1$ 
7:   Compute the conjugate direction using Algorithm 3
8: end while

```

4. Analysis**4.1. Convergence**

We conduct analysis of Algorithm 8 as an instance and Algorithm 9 can be proved using a similar method. First, we prove that the objective function (9) is Lipschitz continuously differentiable [34] on the product space $\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ over the Euclidean geometry.

Then, we demonstrate that the proposed Riemannian gradient descent direction (14) has a sufficient decrease in the function value provided that the step size is selected properly depending on the local geometry at each iteration.

The Lipschitz continuity of the gradient of f in the sublevel set [35]

$$S^0 = \{x \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}, f(x) \leq f(x^0)\}. \quad (19)$$

with respect to a point $x^0 \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ is shown below.

Lemma 1. (Lipschitz continuous). Given a point $x^0 \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$, there exists a Lipschitz constant $L_0 > 0$ such that the gradient of f in $\mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ is L_0 -Lipschitz continuous for any $x, y \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$ belonging to the sublevel set S^0 (19) (see [36] for details to this lemma),

$$f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L_0}{2} \|y - x\|^2. \quad (20)$$

Proof. By the definition of function (9), set S^0 , where S^0 is bounded with respect to any $x < \infty$. Furthermore, let B be a closed ball that contains S^0 . For all $x, y \in B$, according to f is C^∞ , we have

$$\begin{aligned} f(y) &= f(x) + \int_0^1 \langle \nabla f(x + \tau(y - x)), y - x \rangle d\tau \\ &= f(x) + \langle \nabla f(x), y - x \rangle + \int_0^1 \langle \nabla f(x + \tau(y - x)) - \nabla f(x), y - x \rangle d\tau \end{aligned}$$

Then,

$$\begin{aligned} |f(y) - f(x) - \langle \nabla f(x), y - x \rangle| &= \left| \int_0^1 \langle \nabla f(x + \tau(y - x)) - \nabla f(x), y - x \rangle d\tau \right| \\ &\leq \int_0^1 \|\nabla f(x + \tau(y - x)) - \nabla f(x)\| \|y - x\| d\tau \\ &\leq \int_0^1 L_0 \|y - x\| d\tau \\ &\leq L_0 \|y - x\| \\ &= \frac{L_0}{2} \|y - x\|^2. \end{aligned}$$

This means that (20) is true on B , and it functions on its subset S^0 . $\square$

Next, we obtain the following sufficient decrease property with Lemma 1.

Lemma 2. At any iterate $x^t = (Q^t, R^t)$ produced by Algorithm 8 before stopping, the following sufficient decrease property holds, provided that the step size s satisfies $0 < s < 2H_t/L_0$ for a positive value $H_t > 0$,

$$f(x^{t+1}) - f(x^t) \leq -C_t(s) \|\nabla f(x^t)\|^2, \quad (21)$$

where $C_t(s) = s(H_t - \frac{L_0 s}{2}) > 0$ and H_t is defined by

$$H_t = \delta + \min(1, \sigma_{\min}^2(R^t)), \quad (22)$$

under the gradient setting (14).

Proof. In Algorithm 8, at iterate $x^t \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$, the Riemannian gradient descent step is $\eta^t = -\nabla f(x^t)$. Let $s > 0$ denote the step size for producing the next iterate: $x^{t+1} = x^t + s\eta^t = (\eta_Q, \eta_R)$. In the gradient setting (14), the partial differentials are

$$\partial_Q f(x) = \eta_Q(RR^T + \delta I_r) \quad \text{and} \quad \partial_R f(x) = \eta_R(1 + \delta). \quad (23)$$

According to Lemma 1, it follows that

$$\begin{aligned}
 f(x^{t+1}) - f(x^t) &\leq \langle \nabla f(x^t), x^{t+1} - x^t \rangle + \frac{L_0}{2} \|x^{t+1} - x^t\|^2 \\
 &= -s \langle \nabla f(x^t), \nabla f(x^t) \rangle + \frac{L_0 s^2}{2} \|\eta^t\|^2 \\
 &= -s (\text{Tr}(\eta_Q^T \eta_Q (RR^T + \delta I_r)) + \text{Tr}(\eta_R \eta_R^T (Q^T Q + \delta I_r))) + \frac{L_0 s^2}{2} \|\eta^t\|^2 \\
 &\leq -s (\delta \|\eta^t\|^2 + \sigma_{\min}^2(R) \|\eta_Q\|_F^2 + \sigma_{\min}^2(Q) \|\eta_R\|_F^2) + \frac{L_0 s^2}{2} \|\eta^t\|^2 \\
 &\leq -s \|\eta^t\|^2 (\delta + \min(1, \sigma_{\min}^2(R))) + \frac{L_0 s^2}{2} \|\eta^t\|^2 \\
 &= -C_t(s) \|\nabla f(x^t)\|^2.
 \end{aligned}$$

□

Next, we prove that Algorithm 8 with the step size selected by the exact line search (16) ensures sufficient decrease at each iteration.

Lemma 3. *The iterates produced by Algorithm 8, with step size chosen by the exact line search (see Algorithm 4) satisfy the following sufficient decrease property,*

$$f(x^{t+1}) - f(x^t) \leq -(H_t^2/2L_0) \|\nabla f(x^t)\|^2. \quad (24)$$

Proof. In Algorithm (8), let $\eta = -\nabla f(x^t)$ denote the Riemannian gradient descent direction at the iterate $x^t \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{r \times n_2}$. From Lemmas 2 and 3, one obtains

$$f(x^t + s\eta) \leq f(x^t) - C_t(s) \|\nabla f(x^t)\|^2.$$

for $s \in [0, 2H_t/L_0]$ with H_t defined in (22). On the other hand, let $\hat{s}$ be the step size computed using Algorithm 4, and the next iterate $x^{t+1} = x^t + \hat{s}\eta$ is the minimum of f along the direction η : $f(x^{t+1}) \leq f(x^t + s\eta)$ by procession, for all $s \geq 0$. Therefore,

$$f(x^{t+1}) \leq \min_{s \in [0, 2H_t/L_0]} f(x^t + s\eta) \quad (25)$$

$$\leq \min_{s \in [0, 2H_t/L_0]} (f(x^t) - C_t(s) \|\nabla f(x^t)\|^2) \quad (26)$$

$$= f(x^t) - (H_t^2/2L_0) \|\nabla f(x^t)\|^2. \quad (27)$$

In Equation (27), the conclusion $\max_{s \in [0, 2H_t/L_0]} C_t(s) = \max_{s \in [0, 2H_t/L_0]} s(H_t - \frac{L_0 s}{2}) = H_t^2/2L_0$ is applied. □

In both Lemmas 2 and 3, the sufficient decrease quantity depends on the local parameter H_t . The quality H_t is useful only when it is a strictly positive number. We address this in Proposition 1 for the gradient setting (12).

Proposition 1. *Under the same settings as in Lemmas 2 and 3, there exists a positive numerical constant $H_* > 0$ such that the quantities (22) are lower bounded,*

$$\inf_{t \geq 0} H_t \geq H_*. \quad (28)$$

Proof. In the gradient setting (12),

$$H_t = \delta + \min(\sigma_{\min}^2(Q^t), \sigma_{\min}^2(R^t)) \geq \sigma > 0.$$

It is easy to find the result (28) can be ensured by $H_* := \sigma$ as claimed. □

Now, we reach the main result using the following theorem.

Theorem 1. Under the problem statement (9), given the initial point x^0 and the gradient setting (12), the sequence generated by Algorithm 8 with the step size (16) converges and a upper bound of the gradient norm shows as follows,

$$\|\nabla f(x^N)\| \leq \sqrt{\frac{2L_0(f(x^0) - f^*)}{H_*N}} \quad (29)$$

after N iterations, where $L_0 > 0$ is the Lipschitz constant in Lemma 1, the numerical constant $H_* > 0$ is given in Proposition (28), and f^* is a lower bound of the function value of (9).

Proof. The convergence of the sequence $(x^t)_{t \geq 0}$ is a direct result of the sufficient decrease property (21) in Lemma 2 and the boundedness of the sequence of function values $f(x^t)_{t \geq 0}$.

Let $N \geq 1$ denote the number of iterations needed for reaching an iterate x^N such that $\|\nabla f(x^N)\| \leq \epsilon$, for a tolerance parameter $\epsilon > 0$.

Because Algorithm 8 does not terminate at $t \leq N - 1$, the gradient norms $\|\nabla f(x^t)\| > \epsilon$ for all $t \leq N - 1$. Adding up the right hand sides of (24) for $t = 0, \dots, N - 1$ follows

$$f(x^N) - f(x^0) \leq - \sum_{t=0}^{N-1} (H_t^2 / 2L_0) \|\nabla f(x^t)\|^2 \quad (30)$$

$$\leq -(\epsilon^2 / 2L_0) \sum_{t=0}^{N-1} H_t^2 \quad (31)$$

$$= -(H_* / 2L_0) \epsilon^2 N. \quad (32)$$

In Equation (32), Proposition 1 is applied. Therefore, the number of iterations satisfies

$$N \leq \frac{2L_0(f(x^0) - f(x^N))}{H_*\epsilon^2} \leq \frac{2L_0(f(x^0) - f^*)}{H_*\epsilon^2}.$$

In other words, the iterate produced by the algorithm after N iteration obeys

$$\|\nabla f(x^N)\| \leq \sqrt{\frac{2L_0(f(x^0) - f^*)}{H_*N}}.$$

□

4.2. Computational Cost

In this subsection, we analyze the computation cost of our QR-based method with the other. It demonstrates the reason that we can obtain better performance than the compared method. After we make a QR factorization to matrix Q , it leads to some computational cost, but what the factorization reaps, the benefits greatly exceeds what it costs using an ingenious trick.

Cost increase. In Figure 2, we mark three parts including computations in retraction as C_1 , C_2 , and C_3 , respectively. For C_1 , we compute $\text{Tr}(Q^T Q)$ as the leading part where $Q \in \mathbb{R}^{n_1 \times r}$, the cost of which is $2n_1r^2$. For C_2 , we compute the QR decomposition of a matrix in $\mathbb{R}^{n_1 \times r}$ by the MGS algorithm, which costs $2n_1r^2$. For C_3 , it just computes the product of matrices $\tilde{R} \in \mathbb{R}^{r \times r}$ and $R \in \mathbb{R}^{r \times n_2}$; hence, it costs $2n_2r^2$. Assume that the rate of the good orthogonality is $1 - \theta_0$, and the iteration number is k_{iter} ; then, the whole increasing cost is $k_{\text{iter}}(C_1 + \theta_0(C_2 + C_3))$.

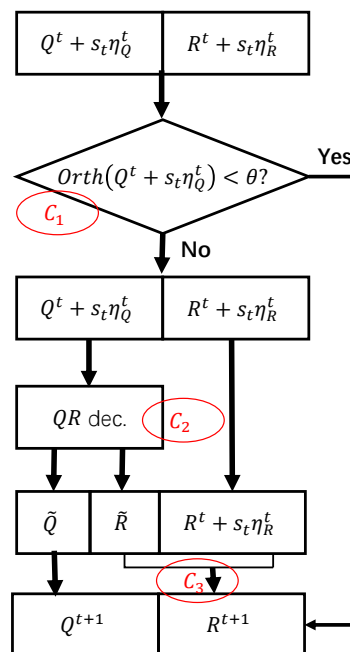


Figure 2. Illustration of the QR factorization in retraction.

Cost decrease. A simple thought is to reduce the cost in each iteration process. First, we consider the gradient of the objective function (9), and the computational costs are summarized in Table 1, where $C_{\text{chol}} = 1/3$ is a coefficient in the Cholesky decomposition while computing the inverse. Therefore, once we compute the gradient we have $D_1 = 2(n_1 + n_2)r^2 + C_{\text{chol}}r^3$ reductions directly with respect to algorithms without QR. Second, we consider the metric (10), and the computational costs are summarized in Table 2. When we compute the metric, the reduction is $D_2 = 2(n_1 + r)r^2$ under the QR method than those without it.

Table 1. Computational costs of the gradients of the objective function.

Computation	Cost
$P_{\Omega}(QR - M)R^T(RR^T + \delta I_r)^{-1}$	$(4r + 1) \Omega + 2(n_1 + n_2)r^2 + C_{\text{chol}}r^3$
$Q^T P_{\Omega}(QR - M)(Q^T Q + \delta I_r)^{-1}$	$(4r + 1) \Omega + 2(n_1 + n_2)r^2 + C_{\text{chol}}r^3$
$Q^T P_{\Omega}(QR - M)/(1 + \delta)$	$(4r + 1) \Omega $

Table 2. Computation costs of the metric.

Computation	Cost
$\text{Tr}(\xi_Q^T \eta_Q (RR^T + \delta I_r))$	$2(n_1 + n_2 + r)r^2$
$\text{Tr}(\xi_R^T \eta_R^T (Q^T Q + \delta I_r))$	$2(n_1 + n_2 + r)r^2$
$\text{Tr}(\xi_R^T \eta_R^T (1 + \delta))$	$2n_2 r^2$

In Algorithm 8, if we find the step size s_t using the exact line search (Algorithm 4), then the reduction is $k_{\text{iter}}D_1$, because the exact line search needs not to compute the metric. Furthermore, the reduction in the computational cost at one iteration is $D_1 - (C_1 + \theta_0(C_2 + C_3)) = \frac{1}{3}r^3 + 2(1 - \theta_0)n_2r^2 - 2n_1r^2$.

In Algorithm 9, if we find the step size s_t using the inexact line search (Algorithm 5), the reduction in the worst case is $k_{\text{iter}}(D_1 + imD_2)$. Furthermore, the reduction in the computational cost at one iteration is $(D_1 + imD_2) - (C_1 + \theta_0(C_2 + C_3)) = (2im + \frac{1}{3})r^3 + 2(im - r)n_1r^2 + 2(1 - \theta_0)n_2r^2$.

5. Numerical Experiments

This section shows a numerical comparison of our algorithms with the recent RGD/RCG algorithms [26], which outperforms existing matrix factorization models on manifolds. The experiments are divided into two parts: in the first part, we test our algorithm on synthetic data, whereas in the second part, we provide the results on an empirical dataset PeMS Traffic[37].

To assess the algorithmic performance, we use the root mean square error (RMSE). Given a matrix $M \in \mathbb{R}^{n_1 \times n_2}$ observed on Ω , the RMSE of $X \in \mathbb{R}^{n_1 \times n_2}$ with respect to M is defined by

$$\text{RMSE}(X; \Omega) = \sqrt{\sum_{(i,j) \in \Omega} (X_{ij} - M_{ij})^2 / |\Omega|}. \quad (33)$$

Other parameters used in experiments are as follows: (1) p is the probability of an entry being observed; (2) the stopping parameter $\epsilon = 10^{-10}$ is one of the two parameters stop the iteration process when RMSE reaches it; (3) the iteration budget parameter $\lambda = 250$ is another parameter that stops the iteration process when iterating a specific amount of times over it; (4) the metric parameter $\delta = 10^{-4}$ helps the metric be well defined; (5) the orthogonality parameter $\theta = 0.01$ is used to judge whether a matrix has good orthogonality; and (6) the oversampling factor $OSF \in (2.5, 3)$ according to [14], defined by $OSF = |\Omega| / (r(n_1 + n_2 - r))$, which decides the difficulty of the problem.

In our experiment, we first fix the values of n_1 , n_2 , and p . Next, we determine the difficulty of recovery, which can be characterized by the over sampling factor (OSF). Following [14], we set the OSF in $(2.5, 3)$. Finally, we determine the value of the rank by $r = \lfloor 11/30 n_1 n_2 p / (n_1 + n_2) \rfloor$. To ensure that the matrix M is low ranked (e.g., $r = 10$), there are two methods. One is setting n_1 and n_2 as small as possible given the values of p . For example, given $p = 0.2$, the values of n_1 and n_2 are about 250. Because of the small size, the problem is trivial. The other is letting p be smaller given the larger values of n_1 and n_2 . This is what was performed in our experiment. For example, given $n_1 = n_2 = 2000$ in Figure 2, we set $p = 0.05$ and obtain $r = 18$.

All numerical experiments were performed on a desktop with 16-core Intel i7-10700F CPUs and 32GB of memory running Windows10 and MATLAB R2022b. The source code is available at <https://github.com/Cz1544252489/qrcode> (accessed on 14 February 2023).

5.1. Synthetic Data

Initially, we provide some comments about the chosen rank on synthetic data. We first fix the values of n_1 , n_2 , and p . Next, we determine the difficulty of recovery, which can be characterized by the oversampling factor (OSF). Following [14], we set the OSF in $(2.5, 3)$. Finally, we can determine the value of the rank by $r = \lfloor 11/30 n_1 n_2 p / (n_1 + n_2) \rfloor$.

We generate two observed matrices M_1 and M_2 with probability p , which is the ratio of an entry being observed defined by $M_1 = FQ$ with $(F, Q^T) \in \mathbb{R}^{n_1 \times r} \times \mathbb{R}^{n_2 \times r}$ and $M_2 = M_1 / (\max(M_1) - \min(M_1))$, where (F, Q^T) are composed of columns that are i.i.d. Gaussian vectors. The reason why we generate them is to test our algorithm on different scale of entries, and it will be measured by $E(M)$ that is the average of random entries.

Table 3 and Table 4 show the results with matrices size of 2000×2000 and 4000×4000 . And Table 5 shows the results with matrices size ranging from 2000×2000 to 8000×2000 .

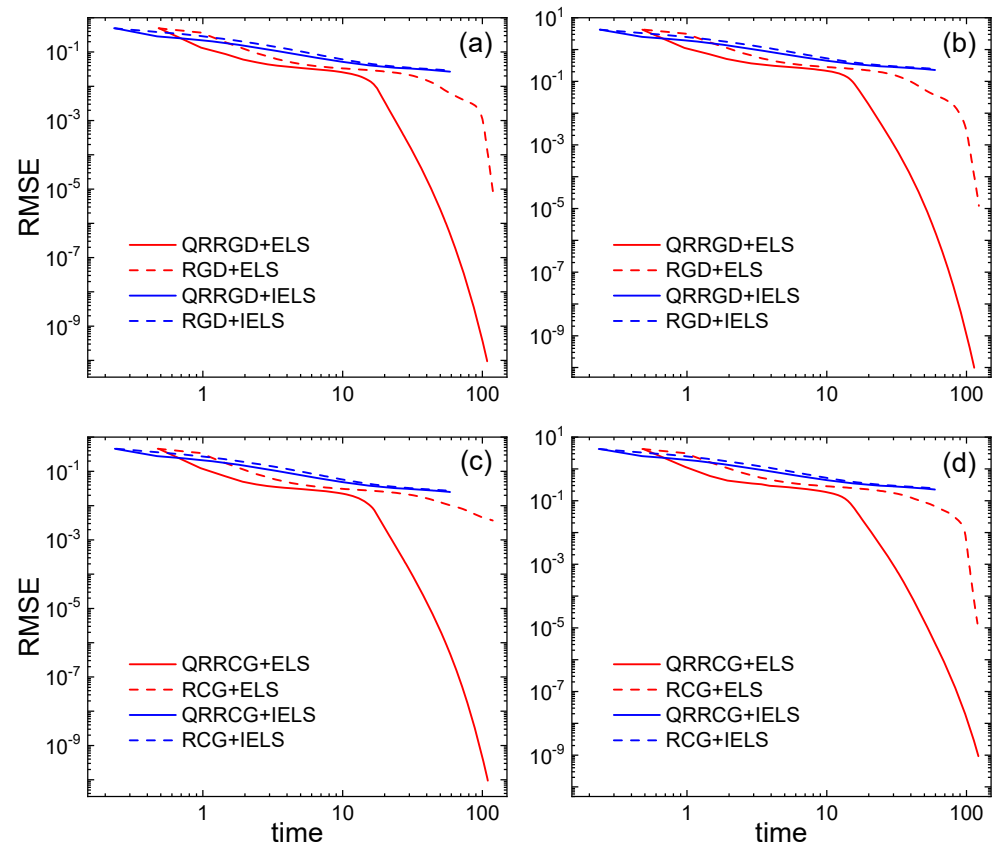


Figure 3. Performance comparison of RGD (a,b) and RCG (c,d) with and without QR factorization. Simulations were carried out on square matrices M_1 (a,c) and M_2 (b,d). We apply the ELS and IELS to find the step size. The parameter values are $n_1 = 2000$, $n_2 = 2000$, $r = 18$, and $p = 0.05$.

Table 3. Computational results for Figure 3 with matrices size of 2000×2000 . The time is rounded to three decimal places and RMSE is rounded to five decimal places.

Subfigure	Method	Time	RMSE	Iteration
(a)	QRRGD+ELS	108.282	9.42416×10^{-11}	223
	RGD+ELS	120.644	5.65317×10^{-6}	250
	QRRGD+IELS	58.763	2.66777×10^{-2}	250
	RGD+IELS	57.905	2.88969×10^{-2}	250
(b)	QRRGD+ELS	113.655	9.89826×10^{-11}	237
	RGD+ELS	122.815	1.21759×10^{-5}	250
	QRRGD+IELS	59.640	2.27510×10^{-1}	250
	RGD+IELS	59.400	2.45927×10^{-1}	250
(c)	QRRCG+ELS	109.363	9.39990×10^{-11}	227
	RCG+ELS	118.726	3.69309×10^{-3}	250
	QRRCG+IELS	58.677	2.50264×10^{-2}	250
	RCG+IELS	59.849	2.68750×10^{-2}	250
(d)	QRRCG+ELS	121.740	9.28316×10^{-10}	250
	RCG+ELS	119.905	1.36830×10^{-5}	250
	QRRCG+IELS	59.449	2.24703×10^{-1}	250
	RCG+IELS	58.432	2.45275×10^{-1}	250

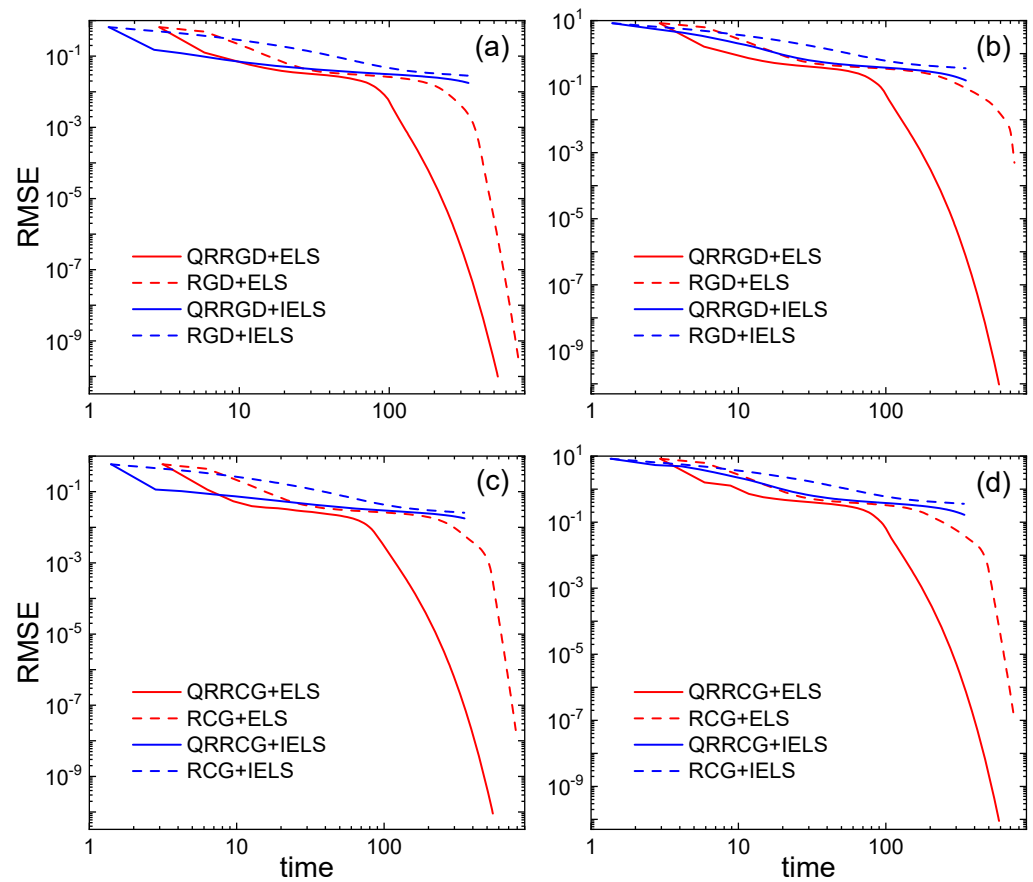


Figure 4. Performance comparison of the RCG with and without QR factorization. Simulations were carried out on square matrices M_1 (a,c) and M_2 (b,d). We apply ELS and IELS to find the step size. The parameter values are $n_1 = 4000$, $n_2 = 4000$, $r = 36$, and $p = 0.05$.

Table 4. Computational results for Figure 4 with matrices size of 4000×4000 . The time is rounded to three decimal places and RMSE is rounded to five decimal places.

Subfigure	Method	Time	RMSE	Iteration
(a)	QRRGD+ELS	529.851	9.89657×10^{-11}	181
	RGD+ELS	733.993	2.40792×10^{-10}	250
	QRRGD+IELS	337.730	1.76829×10^{-2}	250
	RGD+IELS	335.950	2.82502×10^{-2}	250
(b)	QRRGD+ELS	586.522	9.55942×10^{-11}	201
	RGD+ELS	745.970	4.96694×10^{-4}	250
	QRRGD+IELS	349.708	1.53603×10^{-1}	250
	RGD+IELS	348.213	3.59285×10^{-1}	250
(c)	QRRCG+ELS	547.370	9.03539×10^{-11}	173
	RCG+ELS	782.545	1.99382×10^{-8}	250
	QRRCG+IELS	351.655	1.77931×10^{-2}	250
	RCG+IELS	350.344	2.56178×10^{-2}	250
(d)	QRRCG+ELS	586.297	9.00940×10^{-11}	199
	RCG+ELS	738.357	1.38944×10^{-7}	250
	QRRCG+IELS	341.237	1.66425×10^{-1}	250
	RCG+IELS	338.835	3.59287×10^{-1}	250

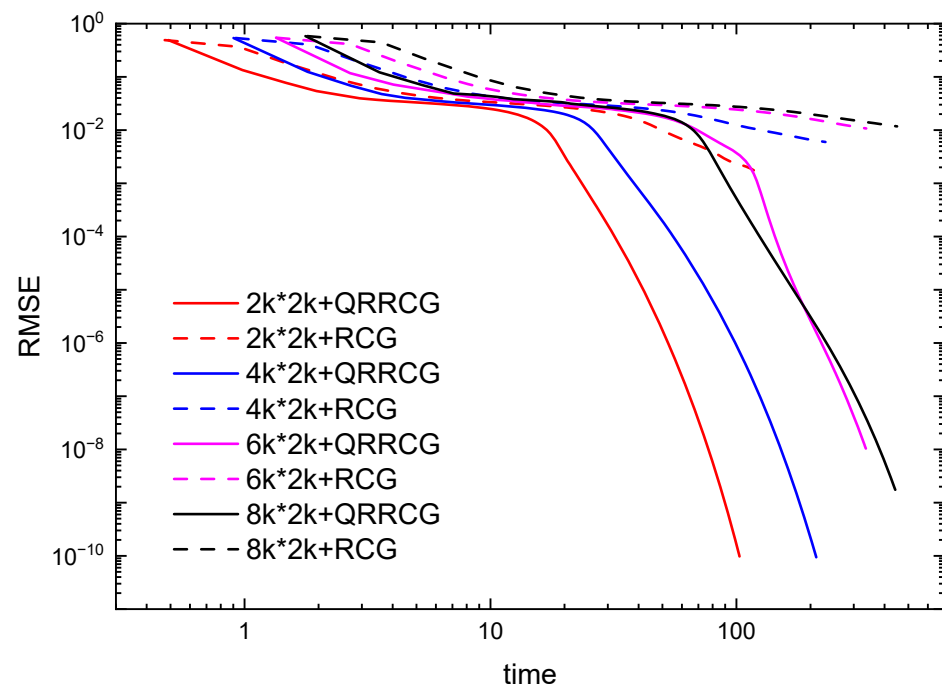


Figure 5. Performance comparison of RCG with and without QR factorization. Simulations were carried out on square matrices M_1 . We applied ELS. The parameter values are $n_1 = 2000, 4000, 6000, 8000, n_2 = 2000, p = 0.05$, then $r = 18, 24, 27, 29$.

Table 5. Computational results for Figure 5 with matrices size ranging from 2000×2000 to 8000×2000 . The time is rounded to three decimal places and RMSE is rounded to five decimal places.

r, OSF	Size	Method	Time	RMSE	Iteration
18,2.79	2000×2000	QRRCG	102.939	9.78333×10^{-11}	210
		RCG	118.397	1.71532×10^{-3}	250
24,2.79	4000×2000	QRRCG	211.122	9.43873×10^{-11}	234
		RCG	230.015	5.98057×10^{-3}	250
27,2.79	6000×2000	QRRCG	335.745	1.03872×10^{-8}	250
		RCG	336.917	1.07408×10^{-2}	250
29,2.77	8000×2000	QRRCG	442.133	1.74723×10^{-9}	250
		RCG	449.752	1.17567×10^{-1}	250

5.2. Empirical Data

In this part, we test our algorithm on the PeMS Traffic [37] dataset. It is a matrix with a size of 963×10560 containing traffic occupancy rates (between 0 and 1) recorded across time by $m = 963$ sensors placed along different lanes of freeways in the San Francisco Bay Area. The recordings are sampled every 10 minutes, covering a period of 15 months. The column index set corresponds to the time domain and the row index set corresponds to geographical points (sensors), which are referred to as the spatial domain. In the experiment, we use the

part of test dataset; it has 173 rows and 6837 columns with $p = 0.05$. Table 6 shows the results on the empirical data.

Table 6. Computational results for Figure 6 on PeMS Traffic. The time is rounded to three decimal places and RMSE is rounded to five decimal places.

Subfigure	Method	Time	RMSE	Iteration
(a)	QRG+ELS	189.945	1.57521×10^{-2}	250
	RGD+ELS	192.104	1.57584×10^{-2}	250
	QRRGD+IELS	116.213	1.62539×10^{-2}	250
	RGD+IELS	116.576	1.62615×10^{-2}	250
(b)	QRRGD+ELS	198.433	1.59324×10^{-2}	250
	RGD+ELS	196.615	1.59392×10^{-2}	250
	QRRGD+IELS	115.070	1.63650×10^{-2}	250
	RGD+IELS	113.418	1.63721×10^{-2}	250
(c)	QRRCG+ELS	196.291	1.54291×10^{-2}	250
	RCG+ELS	194.373	1.54170×10^{-2}	250
	QRRCG+IELS	112.798	1.59846×10^{-2}	250
	RCG+IELS	122.820	1.59796×10^{-2}	250
(d)	QRRCG+ELS	201.373	1.44327×10^{-2}	250
	RCG+ELS	196.029	1.44283×10^{-2}	250
	QRRCG+IELS	112.663	1.48999×10^{-2}	250
	RCG+IELS	112.358	1.48827×10^{-2}	250

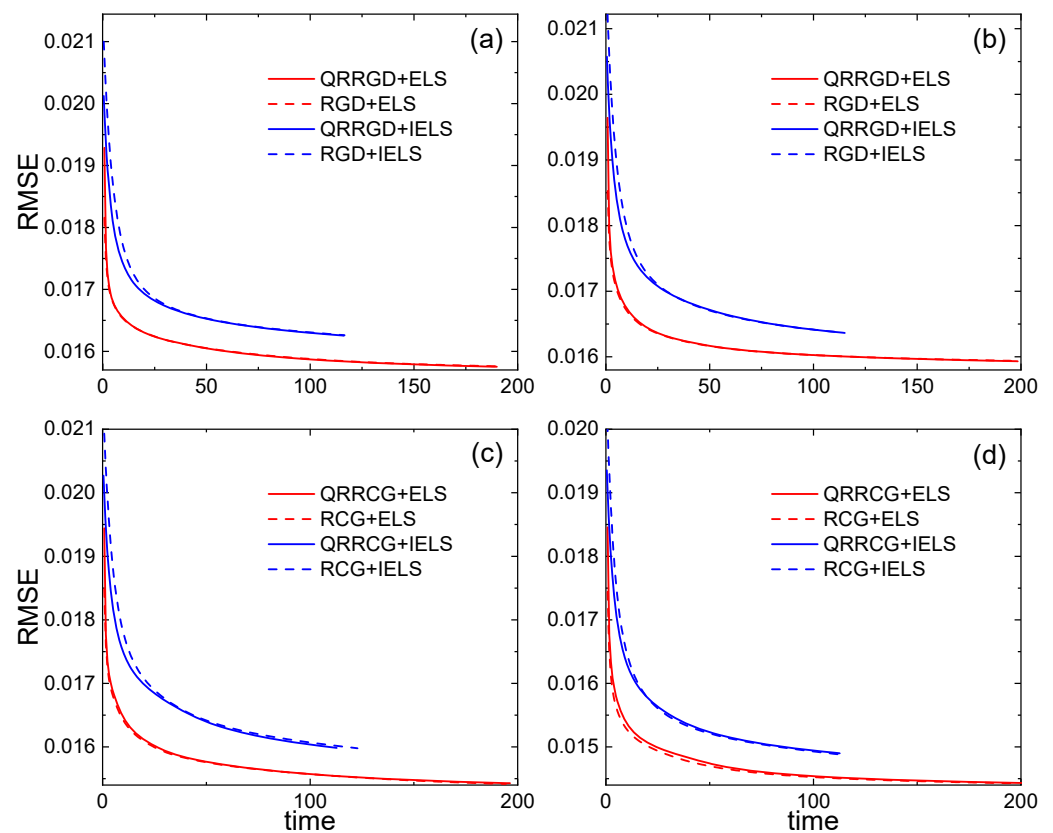


Figure 6. Performance comparison of RGD (a,b) and RCG (c,d) with and without QR factorization. Simulations were carried out on the same matrices M built by PeMS Traffic. We applied ELS and IELS to find the step size. The parameter values are $n_1 = 173$, $n_2 = 6837$, $p = 0.05$, $r = 3$, and $OSF = 2.82$.

As shown above, solid lines represent the results of our algorithms with QR factorization, whereas dashed lines correspond to those of the algorithms with rank factorization [26]. For synthetic data, our algorithms either yield better solutions or run with less time in comparison to [26] on most cases. Whereas for the empirical dataset, it shows a slight advantage for weak structures on earth. It has been demonstrated that the algorithms in [26] outperform the state-of-the-art methods using alternating minimization and the manifold concept.

Furthermore, we briefly measure the ratio of speedup from the compared algorithm. It can be defined by the means of speedup on all our experiment, that is, $SU = \sum_{i \in E} SU_i / |E|$, where E is the set of experiments. Furthermore, a single speedup SU_i defined as below:

$$SU_i = \begin{cases} 0 & \text{if } t_1 > t_2 \text{ and } \epsilon_1 > \epsilon_2 \\ |\frac{t_2 \epsilon_2}{t_1 \epsilon_1} - 1| & \text{otherwise} \end{cases} \quad (34)$$

where t_1 and ϵ_1 are the time in seconds and the RMSE of the QR method, respectively, whereas t_2 and ϵ_2 are the time in seconds and the RMSE of the compared method. Finally, we obtain $SU = 24.00\%$.

6. Conclusions

We have proposed two LRMC algorithms, QRRGD and QRRCG, for reconstruction of an observed matrix via QR-based retraction on manifolds. These two algorithms are computationally efficient and have higher accuracy, demonstrated by theoretical analysis of computational costs and numerical experiments with synthetic data and a real-world dataset PeMS Traffic. To improve efficacy, one could adjust the values of other parameters such as using smaller θ for orthogonality, a larger OSF , and a more suitable δ value for the metric. On the other hand, different conjugate methods [38], as well as the rank adaptive method [39], can be considered.

Author Contributions: Conceptualization, S.Y. and X.J.X.; Methodology, S.Y. and X.J.X.; Validation, X.J.X.; Formal analysis, K.W. and Z.C.; Investigation, K.W. and Z.C.; Writing—original draft, K.W. and Z.C.; Writing—review & editing, S.Y. and X.J.X.; Project administration, X.J.X.; Funding acquisition, S.Y. and X.J.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of China under Grant Nos. 11971296 and 12071281.

Data Availability Statement: Data available in a publicly accessible repository that does not issue DOIs Publicly available datasets were analyzed in this study. This data can be found here: <https://file.cz123.top/DatainManQR/>.

Acknowledgments: The authors thank anonymous referees for their valuable comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Candes, E.J.; Plan, Y. Matrix Completion With Noise. *Proc. IEEE* **2010**, *98*, 925–936. [\[CrossRef\]](#)
2. Xu, Y.; Yin, W.; Wen, Z.; Zhang, Y. An alternating direction algorithm for matrix completion with nonnegative factors. *Front. Math. China* **2012**, *7*, 365–384. [\[CrossRef\]](#)
3. Markovsky, I.; Usevich, K. Structured Low-Rank Approximation with Missing Data. *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 814–830. [\[CrossRef\]](#)
4. Markovsky, I. Recent progress on variable projection methods for structured low-rank approximation. *Signal Process.* **2014**, *96*, 406–419. [\[CrossRef\]](#)
5. Usevich, K.; Comon, P. Hankel Low-Rank Matrix Completion: Performance of the Nuclear Norm Relaxation. *IEEE J. Sel. Top. Signal Process.* **2016**, *10*, 637–646. [\[CrossRef\]](#)
6. Davenport, M. An overview of low-rank matrix recovery from incomplete observations. *IEEE J. Sel. Top. Signal Process.* **2016**, *10*, 608–622. [\[CrossRef\]](#)
7. Chi, Y. Low-rank matrix completion [lecture notes]. *IEEE Signal Process. Mag.* **2018**, *35*, 178–181. [\[CrossRef\]](#)

8. Ding, Y.; Krislock, N.; Qian, J.; Wolkowicz, H. Sensor Network Localization, Euclidean Distance Matrix completions, and graph realization. *Optim. Eng.* **2010**, *11*, 45–66. [\[CrossRef\]](#)
9. Liu, Z.; Vandenberghe, L. Interior-Point Method for Nuclear Norm Approximation with Application to System Identification. *SIAM J. Matrix Anal. Appl.* **2010**, *31*, 1235–1256. [\[CrossRef\]](#)
10. Jacob, M.; Mani, M.P.; Ye, J.C. Structured Low-Rank Algorithms: Theory, Magnetic Resonance Applications, and Links to Machine Learning. *IEEE Signal Process. Mag.* **2020**, *37*, 54–68. [\[CrossRef\]](#)
11. Jawanpuria, P.; Mishra, B. Structured low-rank matrix learning: Algorithms and applications. *arXiv* **2017**, arXiv:1704.07352.
12. Lu, S.; Ren, X.; Liu, F. Depth Enhancement via Low-rank Matrix Completion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 3390–3397.
13. Nguyen, L.T.; Kim, J.; Shim, B. Low-rank matrix completion: A contemporary survey. *IEEE Access* **2019**, *7*, 94215–94237. [\[CrossRef\]](#)
14. Vandereycken, B. Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.* **2013**, *23*, 1214–1236. [\[CrossRef\]](#)
15. Wang, H.; Zhao, R.; Cen, Y.; Liang, L.; He, Q.; Zhang, F.; Zeng, M. Low-rank matrix recovery via smooth rank function and its application in image restoration. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1565–1576. [\[CrossRef\]](#)
16. Candès, E.J.; Recht, B. Exact matrix completion via convex optimization. *Found. Comput. Math.* **2009**, *9*, 717–772. [\[CrossRef\]](#)
17. Cai, J.F.; Candès, E.J.; Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **2010**, *20*, 1956–1982. [\[CrossRef\]](#)
18. Lee, K.; Bresler, Y. Admira: Atomic decomposition for minimum rank approximation. *IEEE Trans. Inf. Theory* **2010**, *56*, 4402–4416. [\[CrossRef\]](#)
19. Jain, P.; Netrapalli, P.; Sanghavi, S. Low-rank matrix completion using alternating minimization. In Proceedings of the 45th Annual ACM Symposium on Theory of Computing, Palo Alto, CA, USA, 1–4 June 2013; pp. 665–674.
20. Tanner, J.; Wei, K. Low rank matrix completion by alternating steepest descent methods. *Appl. Comput. Harmon. Anal.* **2016**, *40*, 417–429. [\[CrossRef\]](#)
21. Absil, P.A.; Mahony, R.; Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*; Princeton University Press: Princeton, NJ, USA, 2009.
22. Boumal, N. *An Introduction to Optimization on Smooth Manifolds*; Cambridge University Press: Cambridge, UK, 2023.
23. Guglielmi, N.; Scalone, C. An efficient method for non-negative low-rank completion. *Adv. Comput. Math.* **2020**, *46*, 31. [\[CrossRef\]](#)
24. Mishra, B.; Meyer, G.; Bonnabel, S.; Sepulchre, R. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Comput. Stat.* **2014**, *29*, 591–621. [\[CrossRef\]](#)
25. Cambier, L.; Absil, P.A. Robust low-rank matrix completion by Riemannian optimization. *SIAM J. Sci. Comput.* **2016**, *38*, S440–S460. [\[CrossRef\]](#)
26. Dong, S.; Absil, P.A.; Gallivan, K. Riemannian gradient descent methods for graph-regularized matrix completion. *Linear Algebra Its Appl.* **2021**, *623*, 193–235. [\[CrossRef\]](#)
27. Zhu, X. A Riemannian conjugate gradient method for optimization on the Stiefel manifold. *Comput. Optim. Appl.* **2017**, *67*, 73–110. [\[CrossRef\]](#)
28. Sato, H.; Aihara, K. Cholesky QR-based retraction on the generalized Stiefel manifold. *Comput. Optim. Appl.* **2019**, *72*, 293–308. [\[CrossRef\]](#)
29. Golub, G.H.; Van Loan, C.F. *Matrix Computations*; JHU Press: Baltimore, MD, USA, 2013.
30. Keshavan, R.H.; Montanari, A.; Oh, S. Matrix completion from noisy entries. *J. Mach. Learn. Res.* **2010**, *11*, 2057–2078.
31. Dai, Y.H.; Yuan, Y. A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optim.* **1999**, *10*, 177–182. [\[CrossRef\]](#)
32. Armijo, L. Minimization of functions having Lipschitz continuous first partial derivatives. *Pac. J. Math.* **1966**, *16*, 1–3. [\[CrossRef\]](#)
33. Björck, Å.; Paige, C.C. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.* **1992**, *13*, 176–190. [\[CrossRef\]](#)
34. O’Seacoid, M. *Metric Spaces*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
35. Boyd, S.; Boyd, S.P.; Vandenberghe, L. *Convex optimization*; Cambridge University Press: Cambridge, UK, 2004.
36. Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2003.
37. Dua, D.; Graff, C. UCI Machine Learning Repository, 2017. Available online: <http://archive.ics.uci.edu/ml> (accessed on 10 February 2019).
38. Liu, J.; Feng, Y.; Zou, L. Some three-term conjugate gradient methods with the inexact line search condition. *Calcolo* **2018**, *55*, 1–16. [\[CrossRef\]](#)
39. Zhou, G.; Huang, W.; Gallivan, K.A.; Van Dooren, P.; Absil, P.A. A Riemannian rank-adaptive method for low-rank optimization. *Neurocomputing* **2016**, *192*, 72–80. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.