AIMS Mathematics, 10(12): 28556–28582.
DOI: 10.3934/math.20251257
Received: 01 October 2025
Revised: 24 November 2025
Accepted: 01 December 2025
Published: 04 December 2025

https://www.aimspress.com/journal/Math

Research article

A Riemannian regularized Gauss–Newton method for low-rank matrix completion

Xiaojing Zhu*and Fengyi Yuan

College of Mathematics and Physics, Shanghai University of Electric Power, Shanghai 201306, China

* **Correspondence:** Email: xjzhu2013@shiep.edu.cn.

Abstract: In this paper, we propose and analyze a Riemannian Gauss–Newton method for the low-rank matrix completion problem. Different from the existing second-order Riemannian optimization methods for this problem, we adopt the approximate Riemannian Hessian of Gauss–Newton methods instead of the true Riemannian Hessian. This approximate Riemannian Hessian has a computationally efficient and symmetric positive semidefinite structure. Added with a regularization term, the corresponding Riemannian Gauss–Newton equation can be solved efficiently by the linear conjugate gradient method. Global and local convergence of the proposed method are established under mild assumptions. Preliminary numerical results on synthetic and image recovery problems with various dimensionalities and ranks are reported to demonstrate the efficiency of the proposed method.

Keywords: low-rank matrix completion; fixed-rank manifold; manifold optimization; Riemannian Gauss–Newton method
Mathematics Subject Classification: 49Q99, 65K05, 90C26, 90C30, 90C48

1. Introduction

This paper considers the problem of low-rank matrix completion, which aims to recover a target matrix $A \in \mathbb{R}^{m\times n}$ from a partially observed set of entries indexed by Ω. We focus on the mathematical formulation of this problem as given in [33]:

$$\min_X \; F(X) := \frac{1}{2}\|\mathcal{P}_\Omega(X - A)\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad X \in \mathcal{M}_l, \tag{1}$$

where $\mathcal{P}_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is the projection that preserves entries in the observed set Ω and sets others to zero, i.e.,

$$[\mathcal{P}_\Omega(X)]_{ij} = \begin{cases} X_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise,} \end{cases}$$

$\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, and $\mathcal{M}_l := \{X \in \mathbb{R}^{m\times n} : \mathrm{rank}(X) = l\}$ is the manifold of $m \times n$ matrices of rank l. This problem arises in numerous applications, and the reader is referred to the survey paper [24] for concrete examples.

Riemannian optimization or manifold optimization [3, 6, 28] is a class of methodologies for optimizing a function f over a Riemannian manifold $\mathcal{M}$, i.e.,

$$\min\ f(x)\ \text{ s.t. }\ x \in \mathcal{M}.$$

Riemannian optimization has become one of the mainstream methods for solving problem (1) because it can make full use of the manifold structure of the feasible set $\mathcal{M}_l$.

In the seminal work [33], an efficient Riemannian conjugate gradient (CG) method was developed based on the embedded submanifold geometry of $\mathcal{M}_l$. An adaptive regularized Newton method was developed in [14] for general optimization problems on Riemannian submanifolds, including problem (1) as a special case. The Riemannian Newton continuation method in [29] and the Riemannian inexact gradient descent method in [37] have also been shown to be promising for solving problem (1) based on submanifold geometry. Submanifold-based optimization methods for problem (1) can be extended to symmetric positive semidefinite fixed-rank matrix manifolds [34], low-rank matrix varieties [12, 25, 38], nonnegative fixed-rank matrix set [30], fixed-rank tensor manifolds [18, 19, 31], and low-rank tensor varieties [13].

By leveraging matrix factorizations, optimization methods on quotient manifolds were developed. Two Riemannian Newton methods based on Riemannian quotient metrics were developed in [1]. Riemannian gradient descent and trust-region methods based on various Riemannian quotient manifold structures were studied in [23]. A Riemannian CG method based on a Riemannian quotient metric with scaling information was studied in [22]. The Riemannian preconditioned CG and trust region methods in [7] and the stochastic Newton method with subsampling in [11] solved a Grassmann manfold reformulation [10] of problem (1). Quotient manifold-based optimization methods for problem (1) can be extended to Hermitian positive semidefinite fixed-rank matrix manifolds [36] and fixed-rank tensor manifolds [9].

Apart from manifold-based approaches, Euclidean optimization methods based on low-rank factorization and alternating minimization can also tackle problem (1), e.g., [16, 32, 35]. Additionally, instead of the fixed-rank formulation (1), models and algorithms based on nuclear norm relaxation have been extensively studied and widely applied [8, 21, 27].

In this paper, we develop a submanifold-based Riemannian Gauss–Newton method different from the existing second-order Riemannian optimization methods such as [14, 29]. Our method adopts the approximate Riemannian Hessian of Gauss–Newton methods instead of the true Riemannian Hessian. The approximate Riemannian Hessian has simple form and is symmetric positive semidefinite. Adding a regularization term, we solve the corresponding Riemannian Gauss–Newton equation by the linear CG method. After computing the search direction, a backtracking line search procedure with an 'optimal' initial guess of the stepsize is performed. Approximate Newton or Gauss–Newton methods have also been applied in other optimization problems on matrix or tensor manifolds such as [9, 19, 26].

The main contributions of this work are twofold. In the algorithmic aspect, we propose a Riemannian regularized Gauss–Newton method with implementation details specifically for the low-rank matrix completion problem. A particular CG procedure is exploited to solve the

corresponding regularized Gauss–Newton equation. Although similar approaches have appeared in Riemannian least-squares formulations, our approach is tailored to the matrix completion task, presenting a special treatment of subproblem solving and parameter settings. In the theoretical aspect, we analyze the properties of the Riemannian Hessian with its Gauss–Newton approximation and prove the global and local convergence of the proposed algorithm. Although the convergence proofs follow standard arguments from Riemannian optimization, we provide specific convergence analysis for this particular problem, relying on our parameter and stepsize designs. It also yields several special results, such as the properties of step size.

This paper is organized as follows. We review essential preliminaries on Riemannian geometry and Riemannian optimization related to our problem and method in Section 2. Our Riemannian regularized Gauss–Newton method with CG implementation for solving subproblems are presented in Section 3. Section 4 is devoted to global and local convergence analysis. In Section 5, we demonstrate the efficiency of the proposed algorithm via numerical experiments. Finally, we conclude the paper in Section 6.

We close this section by describing some notations used later. Let $\mathcal{M}$ be a Riemannian submanifold of a Euclidean space and f a twice differentiable function defined on $\mathcal{M}$. $T_x\mathcal{M}$ denotes the tangent space to $\mathcal{M}$ at a point $x \in \mathcal{M}$. $\langle\cdot,\cdot\rangle$ denotes the underlying Riemannian metric, and $\|\cdot\| := \sqrt{\langle\cdot,\cdot\rangle}$ the associated norm. ∇f, $\nabla^2 f$, $\mathrm{grad} f$, and $\mathrm{Hess} f$ denote the Euclidean gradient, Euclidean Hessian, Riemannian gradient, and Riemannian Hessian of f, respectively. ∇ also denotes the Riemannian connection on a manifold. $\mathrm{Tr}(\cdot)$ denotes the trace of a square matrix. $o(\cdot)$ and $O(\cdot)$ denote higher-order infinitesimal quantities and same-order or lower-order infinitesimal quantities, respectively. $\succ$ ($\succeq$) and $\prec$ ($\preceq$) are the symbols of generalized inequalities in the sense of (semi)definiteness of quadratic forms.

2. Preliminaries

These geometric foundations and optimization tools provide the necessary framework for developing efficient algorithms for low-rank matrix completion, which we will explore in the subsequent sections.

2.1. Basic geometry of the fixed-rank matrix manifold

The set $\mathcal{M}_l$ forms a Riemannian submanifold of $\mathbb{R}^{m\times n}$ when equipped with the standard metric

$$\langle \xi, \eta\rangle := \mathrm{Tr}(\xi^\top \eta), \quad \forall\, \xi, \eta \in T_X\mathcal{M}_l. \tag{2}$$

This metric induces a norm $\|\cdot\|$ for tangent vectors, i.e., the Frobenius norm $\|\cdot\|_{\mathrm{F}}$. Below we introduce some of its basic geometry.

At any $X \in \mathcal{M}_l$ with its thin singular value decomposition (SVD) $X = U\Sigma V^\top$, where $U \in \mathbb{R}^{m\times l}$, $V \in \mathbb{R}^{n\times l}$, and $\Sigma \in \mathbb{R}^{l\times l}$, the tangent space $T_X\mathcal{M}_l$ to $\mathcal{M}_l$ at X can be described as [33]

$$\begin{aligned} T_X\mathcal{M}_l &= \left\{ [U \;\; U_\perp] \begin{bmatrix} \mathbb{R}^{l\times l} & \mathbb{R}^{l\times(n-l)} \\ \mathbb{R}^{(m-l)\times l} & 0_{(m-l)\times(n-l)} \end{bmatrix} [V \;\; V_\perp]^\top \right\} \\ &= \left\{ UMV^\top + U_pV^\top + UV_p^\top : M \in \mathbb{R}^{l\times l}, U_p \in \mathbb{R}^{m\times l}, U_p^\top U = 0, V_p \in \mathbb{R}^{n\times l}, V_p^\top V = 0 \right\}, \end{aligned} \tag{3}$$

where $U_\perp$ and $V_\perp$ are orthogonal complements of U and V, respectively. The normal space $N_X\mathcal{M}_l$ to $\mathcal{M}_l$ at X with respect to the metric (2) is given by [33]

$$N_X\mathcal{M}_l = \left\{U_\perp \Lambda V_\perp^\top : \Lambda \in \mathbb{R}^{(m-l)\times(n-l)}\right\} = \{Z \in \mathbb{R}^{m\times n} : U^\top Z = 0, ZV = 0\}.$$

The orthogonal projection $\mathcal{P}_{T_X\mathcal{M}_l}$ onto $T_X\mathcal{M}_l$ is given by [33]

$$\mathcal{P}_{T_X\mathcal{M}_l} : \mathbb{R}^{m\times n} \to T_X\mathcal{M}_l,\ Z \mapsto P_U Z P_V + P_U^\perp Z P_V + P_U Z P_V^\perp, \tag{4}$$

where $P_U := UU^\top$, $P_U^\perp := I - P_U$, $P_V := VV^\top$, and $P_V^\perp := I - P_V$.

Let $\overline{\mathcal{M}}$ be a Riemannian manifold, $\mathcal{M}$ a submanifold, $\overline{f}$ a function defined on $\overline{\mathcal{M}}$ and f the restriction of $\overline{f}$ to $\mathcal{M}$. According to (3.37) in [3], the Riemannian gradient of f is the projection of the Riemannian gradient onto $T_x\mathcal{M}$. This gives the following explicit formula of the Riemannian gradient of the obective function in (1):

$$\mathrm{grad}F(X) = \mathcal{P}_{T_X\mathcal{M}_l}\nabla F(X) = \mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_\Omega(X - A). \tag{5}$$

According to Proposition 5.3.2 in [3], the Riemannian connection ∇ on a submanifold $\mathcal{M}$ of $\overline{\mathcal{M}}$ relates to the connection $\overline{\nabla}$ on the ambient manifold $\overline{\mathcal{M}}$ in the following manner:

$$\nabla_\eta \xi = \mathcal{P}_{T_x\mathcal{M}}\left(\overline{\nabla}_{\overline{\eta}}\overline{\xi}\right),$$

where ξ and η are smooth vector fields on $\mathcal{M}$ and $\overline{\xi}$ and $\overline{\eta}$ are smooth extensions of ξ and η, respectively, to some neighborhoods in $\overline{\mathcal{M}}$. If $\overline{\mathcal{M}}$ is a Euclidean space, the above formula of connections reduces to

$$\nabla_\eta \xi = \mathcal{P}_{T_x\mathcal{M}}\left(\mathrm{D}\overline{\xi}[\overline{\eta}]\right), \tag{6}$$

where $\mathrm{D}\overline{\xi}[\overline{\eta}]$ is the directional derivative of $\overline{\xi}$ along $\overline{\eta}$. According to Definition 5.5.1 in [3], the Riemannian Hessian of f at $x \in \mathcal{M}$ is a linear map from $T_x\mathcal{M}$ to itself defined as

$$\mathrm{Hess}f(x)[\eta] := \nabla_\eta \mathrm{grad}f(x),\ \ \forall\ \eta \in T_x\mathcal{M}. \tag{7}$$

Then, (5)–(7) together with (2) lead to the following explicit expression of the Riemannian Hessian of the objective function in (1).

Proposition 2.1. *The Riemannian Hessian of the objective function in (1) is given by*

$$\begin{aligned}\mathrm{Hess}F(X)[\xi] = {} & P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp(\mathcal{P}_\Omega(\xi) + \mathcal{P}_\Omega(X - A)V_p\Sigma^{-1}V^\top)P_V \\ & + P_U(\mathcal{P}_\Omega(\xi) + U\Sigma^{-1}U_p^\top\mathcal{P}_\Omega(X - A))P_V^\perp.\end{aligned} \tag{8}$$

Proof. This formula was first given by Proposition 2.2 in [33], where it was proved by property (11) of second-order retractions. But, for self-containedness, we give a different but much simpler proof using (5)–(7) directly in Appendix A. □

2.2. Basic knowledge in Riemannian Optimization

Riemannian optimization extends classical nonlinear optimization methods to Riemannian manifolds by utilizing geometric operations. The fundamental concept of retraction [3] is formally defined as follows.

Definition 2.2 (Retraction)**.** A retraction R on a manifold $\mathcal{M}_l$ is a smooth map from the tangent bundle $T\mathcal{M} := \bigcup_{x\in\mathcal{M}} T_x\mathcal{M}$ to $\mathcal{M}$ such that

(1) $R_x(0) = x$ for all $x \in \mathcal{M}$.

(2) $\frac{\mathrm{d}}{\mathrm{d}t}R_x(t\xi)\Big|_{t=0} = \xi$ for all $x \in \mathcal{M}$ and $\xi \in T_x\mathcal{M}$.

If the curve $\gamma(t) = R_x(t\xi)$ has zero acceleration at $t = 0$, i.e., $\nabla_{\gamma'(0)}\gamma'(0) = 0$, then R is called a second-order retraction.

For the fixed-rank manifold $\mathcal{M}_l$, we use the projective retraction defined as [4, 5]

$$R_X(\xi) := \operatorname*{arg\,min}_{Y\in\mathcal{M}_l} \|X + \xi - Y\|_{\mathrm{F}},$$

which can be expressed as the truncated SVD of $X + \xi$, i.e.,

$$R_X(\xi) = \sum_{i=1}^{l} \sigma_i u_i v_i^{\top},$$

where σ_i are the singular values (in decreasing order) of $X + \xi$ and u_i and v_i are the left and right singular vectors of $X + \xi$, respectively.

If ξ is available in form (3), $R_X(\xi)$ can be efficiently computed as follows [5, 33]. First, perform the QR factorizations $U_p = Q_u R_u$ and $V_p = Q_v R_v$. Observe that

$$X + \xi = \begin{bmatrix} U & Q_u \end{bmatrix} \begin{bmatrix} \Sigma + M & R_v^{\top} \\ R_u & 0 \end{bmatrix} \begin{bmatrix} V & Q_v \end{bmatrix}^{\top}.$$

Obtain (U_s, Σ_s, V_s) as an SVD of the $2l$-by-$2l$ matrix $\begin{bmatrix} \Sigma + M & R_v^{\top} \\ R_u & 0 \end{bmatrix}$. Then, we have

$$R_X(X + \xi) = U_+ S_+ V_+^{\top},$$

where $U_+ = [U\ Q_u]U_s(:, 1 : l)$, $V_+ = [V\ Q_v]V_s(:, 1 : l)$, and $S_+ = \Sigma_s(1 : l, 1 : l)$.

Given a retraction R, the Riemannian Newton method takes the following iterative scheme [3]:

$$x_{k+1} = R_{x_k}(t_k\xi_k),$$

where $t_k > 0$ is a stepsize and $\xi_k \in T_{x_k}\mathcal{M}$ is the search direction computed by solving the Riemannian Newton equation

$$\mathrm{Hess} f(x_k)[\xi] = -\mathrm{grad} f(x_k), \quad \xi \in T_{x_k}\mathcal{M}. \tag{9}$$

When $\mathrm{Hess} f(x_k)$ is positive definite, the Riemannian Newton step ξ_k actually solves the quadratic subproblem

$$\min_{\xi\in T_{x_k}\mathcal{M}} f(x_k) + \langle \mathrm{grad} f(x_k), \xi\rangle + \frac{1}{2}\langle \mathrm{Hess} f(x_k)[\xi], \xi\rangle.$$

The pullback function $\widehat{f_x}(\xi) := f(R_x(\xi))$ is a useful tool that sometimes enables the analysis and computation of Riemannian optimization methods to be more convenient. By Definition 2.1, it is easy to see that

$$\nabla\widehat{f_x}(0) = \mathrm{grad} f(x). \tag{10}$$

Moreover, if R is a second-order retraction, if follows from Proposition 5.5.5 in [3] that

$$\nabla^2\widehat{f_x}(0) = \mathrm{Hess} f(x). \tag{11}$$

According to Theorem 22 in [4], the above projective retraction on $\mathcal{M}_l$ is a second-order retraction, so (11) holds in our case.

3. The proposed method

In this section, we propose and explain our new method. In Section 3.1, a framework of Riemannian regularized Gauss–Newton method is presented. In Section 3.2, specific implemention details on solving subproblems are given.

3.1. A Riemannian regularized Gauss–Newton method

According to (9), the Riemannian Newton method for problem (1) iteratively solves the Riemannian Newton equation

$$\mathrm{Hess}F(X_k)[\xi] = -\mathrm{grad}F(X_k), \quad \xi \in T_{X_k}\mathcal{M}_l.$$

For the sake of saving computational cost, we adopt the following approximation. An observation on (8) reveals that if $\mathcal{P}_\Omega(X) \approx \mathcal{P}_\Omega(A)$, then

$$\mathrm{Hess}F(X)[\xi] \approx P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp.$$

Then a natural idea is to replace the exact Riemannian Hessian by the right-hand side of the above equation. For convenience, we define the linear map $\mathcal{H}_X : T_X\mathcal{M}_l \to T_X\mathcal{M}_l$ that closely resembles $\mathrm{Hess}F(X)$ as

$$\mathcal{H}_X[\xi] := \mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_\Omega(\xi) = P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp. \tag{12}$$

According to Section 8.4.1 in [3], $\mathcal{H}_X$ can actually be interpreted as the approximate Riemannian Hessian of Gauss–Newton methods. Specifically, if we write $F(X) = \frac{1}{2}\|\phi(X)\|_{\mathrm{F}}^2$, where $\phi(X) := \mathcal{P}_\Omega(X - A)$, then we have

$$\mathcal{H}_X = \mathrm{D}\phi(X)^* \circ \mathrm{D}\phi(X),$$

where $\mathrm{D}\phi(X)^*$ is the adjoint operator of $\mathrm{D}\phi(X)$.

The approximate Riemannian Hessian $\mathcal{H}_X$ not only approximates the exact Hessian, but also has nice algebraic properties according to the following proposition.

Proposition 3.1. *The map $\mathcal{H}_X$ in (12) is self-adjoint (symmetric) and positive semidefinite. Moreover, $\langle\xi, \mathcal{H}_X[\xi]\rangle \le \langle\xi, \xi\rangle$ for all $X \in \mathcal{M}_l$ and $\xi \in T_X\mathcal{M}_l$.*

Proof. Let ξ and η be any two tangent vectors in $T_X\mathcal{M}_l$. Then it follows from (12) that

$$\begin{aligned}\langle \mathcal{H}_X[\xi], \eta\rangle &= \langle \mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_\Omega(\xi), \eta\rangle = \langle \mathcal{P}_\Omega(\xi), \eta\rangle = \langle \mathcal{P}_\Omega(\xi), \mathcal{P}_\Omega(\eta)\rangle \\ &= \langle \xi, \mathcal{P}_\Omega(\eta)\rangle = \langle \xi, \mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_\Omega(\eta)\rangle = \langle \xi, \mathcal{H}_X[\eta]\rangle,\end{aligned}$$

$$\langle \mathcal{H}_X[\xi], \xi\rangle = \langle \mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_\Omega(\xi), \xi\rangle = \langle \mathcal{P}_\Omega(\xi), \xi\rangle = \langle \mathcal{P}_\Omega(\xi), \mathcal{P}_\Omega(\xi)\rangle \geq 0,$$

and

$$\langle \mathcal{H}_X[\xi], \xi\rangle = \langle \mathcal{P}_\Omega(\xi), \mathcal{P}_\Omega(\xi)\rangle \leq \langle \xi, \xi\rangle.$$

Thus, all the desired properties have been proved. □

Because $\mathcal{H}_X$ is only positive semidefinite according to the above proposition, it is possibly ill-conditioned. Thus, we regularize it as

$$\mathcal{H}_{X,\delta} := \mathcal{H}_X + \delta\mathcal{I}, \tag{13}$$

where $\delta > 0$ is a regularization parameter and $\mathcal{I}$ is the identity map on $T_X\mathcal{M}_l$. Then, we solve in each iteration the equation

$$\mathcal{H}_{X_k,\delta_k}[\xi] = -\text{grad}F(X_k). \tag{14}$$

Equation (14) is called the regularized Riemannian Gauss–Newton equation. The following corollary is an immediate consequence of Proposition 3.1, showing a satisfactory property of $\mathcal{H}_{X,\delta}$.

Corollary 3.2. *The map $\mathcal{H}_{X,\delta}$ in (13) is self-adjoint and positive definite. Moreover,*

$$\delta\mathcal{I} \preceq \mathcal{H}_{X,\delta} \preceq (1+\delta)\mathcal{I}$$

for all $X \in \mathcal{M}_l$ and $\delta > 0$.

Proof. The results are straightforward from Proposition 3.1 and (13). □

Since $\mathcal{H}_{X_k,\delta_k}$ is symmetric postive definite for all $\delta_k > 0$, we can solve (14) efficiently by the linear CG method. Specific implementation details of CG for solving (14) are given in Section 3.2. Compared to the approximate Riemannian Hessian $\mathcal{H}_X$, the true Riemannian Hessian Hess$F(X_k)$ has a drawback that it may be non-positive semidefinite according to Proposition 3.3. This makes Hess$F(X_k)$ rely more heavily on the regularization parameter δ_k or use trust-region techniques [2].

Proposition 3.3. *The Riemannian Hessian* Hess$F(X)$ *given by (8) may be non-positive semidefinite.*

Proof. Define $F_a(X) := \frac{1}{2}\|\mathcal{P}_\Omega(X - A)\|_{\text{F}}^2$ with $A = aX$. Then, it follows from (8), (12), and the proof of Proposition 3.1 that

$$\begin{aligned}\langle \text{Hess}F_a(X)[\xi], \xi\rangle &= \langle \mathcal{H}_X[\xi], \xi\rangle + \langle P_U^\perp\mathcal{P}_\Omega((1-a)X)V_p\Sigma^{-1}V^\top + U\Sigma^{-1}U_p^\top\mathcal{P}_\Omega((1-a)X)P_V^\perp, \xi\rangle \\ &= \langle \mathcal{P}_\Omega(\xi), \mathcal{P}_\Omega(\xi)\rangle + (1-a)\langle P_U^\perp\mathcal{P}_\Omega(X)V_p\Sigma^{-1}V^\top + U\Sigma^{-1}U_p^\top\mathcal{P}_\Omega(X)P_V^\perp, \xi\rangle.\end{aligned}$$

So, $\langle \text{Hess}F_a(X)[\xi], \xi\rangle$ is affine with respect to a, implying that the Riemannian Hessian Hess$F(X)$ may not be positive semidefinite. □

Here we show an example of non-positive semidefinite Hessian following the constructive proof of Proposition 3.3. Let

$$\Omega = \{(1,1)\},\ A = \begin{bmatrix} 6 & 6 \\ 6 & 6 \end{bmatrix},\ X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},\ \xi = \begin{bmatrix} 3 & 1 \\ 1 & -1 \end{bmatrix}.$$

Then, $\langle \mathrm{Hess}F(X)[\xi], \xi \rangle = -1$.

After obtaining ξ_k, we need to compute an appropriate stepsize. Following (3.3) in [33], we use the following 'optimal' strategy for the initial stepsize:

$$t_k^* := \arg\min_t F(X_k + t\xi_k) = -\frac{\langle \mathcal{P}_\Omega(\xi_k), \mathcal{P}_\Omega(X_k - A) \rangle}{\langle \mathcal{P}_\Omega(\xi_k), \mathcal{P}_\Omega(\xi_k) \rangle}. \tag{15}$$

Then, a backtracking procedure is performed to choose a stepsize that satisfies the Armijo-type condition

$$F(R_{X_k}(t_k\xi_k)) \leq F(X_k) + \sigma t_k \langle \mathrm{grad}F(X_k), \xi_k \rangle, \tag{16}$$

where $\sigma \in (0, \frac{1}{2})$.

Now we can formally present a framework of our Riemannian regularized Gauss–Newton method as follows.

Algorithm 1 Riemannian regularized Gauss–Newton method (RRGN)

1: **require:** $X_0 \in \mathcal{M}_l$, $\mu > 0$, $\tau \in (0,2)$, $\{\theta_k\}_{k\in\mathbb{N}} \subset (0,1)$, $\underline{t}, \rho \in (0,1)$, $\sigma \in (0, \frac{1}{2})$.
2: **for** $k = 0, 1, \ldots$ **do**
3: Compute the Riemannian gradient $\mathrm{grad}F(X_k)$.
4: Set the regularization parameter

$$\delta_k = \mu \|\mathrm{grad}F(X_k)\|^\tau. \tag{17}$$

5: Solve equation (14) inexactly by CG to obtain a search direction ξ_k satisfying

$$\|r_k\| \leq \theta_k \|\mathrm{grad}F(X_k)\|, \tag{18}$$

6: where r_k is the residual defined as

$$r_k := \mathcal{H}_{X_k,\delta_k}[\xi^k] + \mathrm{grad}F(X_k). \tag{19}$$

7: Compute $t_k = t_k^{\mathrm{ini}} := \max\{\underline{t}, t_k^*\}$ where t_k^* is defined by (15).
8: **while** the Armijo condition (16) does not hold **do**
9: Set $t_k = \rho t_k$.
10: **end while**
11: Update $X_{k+1} = R_{X_k}(t_k\xi_k)$.
12: **end for**

Remark 1. The motivation of the regularization parameter formula (17) is threefold. First, the regularized Hessian should converge to the true Hessian so as to ensure a fast local convergence rate. Second, the regularization parameter should adapt conveniently to the global convergence analysis (see the proof of Theorem 4.3). Third, such a parameter setting should be simple.

3.2. Solving subproblem

In this subsection, we show how to solve the subproblem, i.e., the regularized Gauss–Newton equation (14). Since (14) is a symmetric positive definite linear system on the tangent space $T_{X_k}\mathcal{M}_l$, we can implement CG to solve it. Below we present two versions of CG algorithms, which are equivalent.

Algorithm 2 is the original CG for solving (14), and Algorithm 3 is a variant of Algorithm 2. Motivated by [33], Algorithm 3 solves (14) while preserving the compact structure in (3), meaning that it computes M, U_p, and V_p separately such that

$$\xi_k = U_k M V_k^\top + U_k V_p^\top + U_p V_k^\top,$$

where $X_k = U_k \Sigma_k V_k^\top$ forms an SVD of X_k. Algorithm 3 requires M_g, U_{pg}, and V_{pg} as input, where

$$\mathrm{grad}F(X_k) = U_k M_g V_k^\top + U_k V_{pg}^\top + U_{pg} V_k^\top$$

is an expression of $\mathrm{grad}F(X_k)$ in form (3). Also note that during the process of Algorithm 3, $M_{Hp,j}$, $U_{pHp,j}$, and $V_{pHp,j}$ can be computed separately such that

$$\begin{aligned}\mathcal{H}_{X_k,\delta_k}[p_j] - \delta_k p_j = \mathcal{H}_{X_k}[p_j] &= \mathcal{P}_{T_k\mathcal{M}_l}\mathcal{P}_\Omega(p_j) \\ &= U_k(M_{Hp,j} - \delta_k M_{p,j})V_k^\top + U_k(V_{pHp,j} - \delta_k V_{pp,j})^\top + (U_{pHp,j} - \delta_k U_{pp,j})V_k^\top.\end{aligned}$$

According to Algorithm 2 in [33], given $X = U\Sigma V^\top \in \mathcal{M}_l$ and $Z \in \mathbb{R}^{m\times n}$, the matrices M, U_p, and V_p satisfying

$$\mathcal{P}_{T_X\mathcal{M}_l}(Z) = UMV^\top + U_p V^\top + UV_p^\top$$

can be computed efficiently as follows: $M = U^\top(ZV)$, $U_p = ZV - UM$, and $V_p = Z^\top U - VM^\top$.

Algorithm 2 CG on $T_{X_k}\mathcal{M}_l$ (original)

require: $\xi_{k,0} = 0$, $r_{k,0} = -\mathrm{grad}F(X_k)$, $p_0 = r_{k,0}$, $J \in \mathbb{N}$.
for $j = 0, 1, \ldots, J$ **do**
  $\alpha_j = \frac{\langle r_{k,j}, r_{k,j}\rangle}{\langle p_j, \mathcal{H}_{X_k,\delta_k}[p_j]\rangle}$.
  $\xi_{k,j+1} = \xi_{k,j} + \alpha_j p_j$.
  $r_{k,j+1} = r_{k,j} - \alpha_j \mathcal{H}_{X_k,\delta_k}[p_j]$.
  if $\|r_{k,j+1}\|$ satisfies (18) **then**
    $\xi_k = \xi_{k,j+1}$.
    return
  end if
  $\beta_{j+1} = \frac{\langle r_{k,j+1}, r_{k,j+1}\rangle}{\langle r_{k,j}, r_{k,j}\rangle}$.
  $p_{j+1} = r_{k,j+1} + \beta_{j+1} p_j$.
end for

Algorithm 3 CG on $T_{X_k}\mathcal{M}_l$ (variant)

require: $M_0 = 0$, $U_{p,0} = 0$, $V_{p,0} = 0$, $M_{r,0} = -M_g$, $U_{pr,0} = -U_{pg}$, $V_{pr,0} = -V_{pg}$, $M_{p,0} = -M_g$, $U_{pp,0} = -U_{pg}$, $V_{pp,0} = -V_{pg}$, $J \in \mathbb{N}$.
$s_0 = \langle M_{r,0}, M_{r,0}\rangle + \langle U_{pr,0}, U_{pr,0}\rangle + \langle V_{pr,0}, V_{pr,0}\rangle$. ▷ $s_0 = \langle r_{k,0}, r_{k,0}\rangle$.
for $j = 0, 1, \ldots, J$ **do**
  $\alpha_j = \frac{s_j}{\langle M_{p,j}, M_{Hp,j}\rangle + \langle U_{pp,j}, U_{pHp,j}\rangle + \langle V_{pp,j}, V_{pHp,j}\rangle}$. ▷ $\alpha_j = \frac{\langle r_{k,j}, r_{k,j}\rangle}{\langle p_j, \mathcal{H}_{X_k,\delta_k}[p_j]\rangle}$.
  $M_{j+1} = M_j + \alpha_j M_{p,j}$.
  $U_{p,j+1} = U_{p,j} + \alpha_j U_{pp,j}$.
  $V_{p,j+1} = V_{p,j} + \alpha_j V_{pp,j}$.
  $M_{r,j+1} = M_{r,j} - \alpha_j M_{Hp,j}$.
  $U_{pr,j+1} = U_{pr,j} - \alpha_j U_{pHp,j}$.
  $V_{pr,j+1} = V_{pr,j} - \alpha_j V_{pHp,j}$.
  $s_{j+1} = \langle M_{r,j+1}, M_{r,j+1}\rangle + \langle U_{pr,j+1}, U_{pr,j+1}\rangle + \langle V_{pr,j+1}, V_{pr,j+1}\rangle$. ▷ $s_{j+1} = \langle r_{k,j+1}, r_{k,j+1}\rangle$.
  if $\sqrt{s_{j+1}}$ satisfies (18) **then**
    $M = M_{j+1}$, $U_p = U_{p,j+1}$, $V_p = V_{p,j+1}$.
    $\xi_k = U_k M V_k^\top + U_k V_p^\top + U_p V_k^\top$.
    return
  end if
  $\beta_{j+1} = \frac{s_{j+1}}{s_j}$. ▷ $\beta_{j+1} = \frac{\langle r_{k,j+1}, r_{k,j+1}\rangle}{\langle r_{k,j}, r_{k,j}\rangle}$.
  $M_{p,j+1} = M_{r,j+1} + \beta_{j+1} M_{p,j}$.
  $U_{pp,j+1} = U_{pr,j+1} + \beta_{j+1} U_{pp,j}$.
  $V_{pp,j+1} = V_{pr,j+1} + \beta_{j+1} V_{pp,j}$.
end for

We close this section by comparing the computational complexities of Algorithms 2 and 3. Algorithm 2 costs $6mnl + 2(m + n)l^2 + 4|\Omega|l$ flops in each iteration, while Algorithm 3 costs $2(2m + n)l^2 + 10|\Omega|l$ flops in each iteration. Therefore, Algorithm 3 is preferable in practice.

4. Convergence

In this section, we analyze the convergence properties of Algorithm 1. Section 4.1 focuses on global convergence, and Section 4.2 is devoted to local convergence.

4.1. Global convergence

In this subsection, we consider the global convergence property of Algorithm 1. Before our proof, we make the following assumption.

Assumptiom 1. There is a constant $L > 0$ such that for $\{(X_k, \xi_k)\}_{k\in\mathbb{N}}$ generated by Algorithm 1, it holds that

$$F(R_{X_k}(\xi_k)) \leq F(X_k) + \langle \mathrm{grad}F(X_k), \xi_k\rangle + \frac{L}{2}\|\xi_k\|^2.$$

Assumption 1 is used in Lemma 4.2 and Theorem 4.3 below. This kind of assumption is commonly used in global convergence analysis of Riemannian optimization methods.

Lemma 4.1. *If the parameter θ_k in (18) satisfies*

$$\theta_k \leq (1-\omega)\sqrt{\frac{\delta_k}{1+\delta_k}}, \tag{20}$$

then

$$\langle \text{grad}F(X_k), \xi_k \rangle \leq -\frac{\omega}{1+\delta_k}\|\text{grad}F(X_k)\|^2, \tag{21}$$

where δ_k is the regularization parameter and $\omega \in (0,1)$.

Proof. It follows from (18)–(20), and Corollary 3.2 that

$$\begin{aligned}
-\langle \text{grad}F(X_k), \xi_k \rangle &= \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k) - r_k] \rangle \\
&= \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k)] \rangle - \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[r_k] \rangle \\
&= \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k)] \rangle - \langle \mathcal{H}^{-\frac{1}{2}}_{X_k,\delta_k}[\text{grad}F(X_k)], \mathcal{H}^{-\frac{1}{2}}_{X_k,\delta_k}[r_k] \rangle \\
&\geq \left(1 - \frac{\|\mathcal{H}^{-\frac{1}{2}}_{X_k,\delta_k}[r_k]\|}{\|\mathcal{H}^{-\frac{1}{2}}_{X_k,\delta_k}[\text{grad}F(X_k)]\|}\right) \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k)] \rangle \\
&\geq \left(1 - \sqrt{\frac{1+\delta_k}{\delta_k}}\theta_k\right) \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k)] \rangle \\
&\geq \omega \langle \text{grad}F(X_k), \mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k)] \rangle \\
&\geq \frac{\omega}{1+\delta_k}\|\text{grad}F(X_k)\|^2.
\end{aligned}$$

Thus, (21) holds. □

Lemma 4.2. *Suppose Assumption 1 holds. If (20) holds, then the Armijo condition (16) holds for all sufficiently small stepsizes, and, moreover, we have for all k that*

$$t_k \geq \min\left\{\underline{t}, \frac{\rho(1-\sigma)\omega\delta_k^2}{2L(1+\delta_k)}\right\}. \tag{22}$$

Proof. If $t_k \leq \frac{(1-\sigma)\omega\delta_k^2}{2L(1+\delta_k)}$, it follows from (18), (19), (21), Corollary 3.2, and $\theta_k \leq 1$ that

$$\begin{aligned}
\frac{1}{2}Lt_k^2\|\xi_k\|^2 &= \frac{1}{2}Lt_k^2\left\|\mathcal{H}^{-1}_{X_k,\delta_k}[\text{grad}F(X_k) - r_k]\right\|^2 \\
&\leq \frac{1}{2\delta_k^2}Lt_k^2(\|\text{grad}F(X_k)\| + \|r_k\|)^2 \\
&\leq \frac{L(1+\theta_k)^2t_k^2}{2\delta_k^2}\|\text{grad}F(X_k)\|^2 \\
&\leq \frac{2Lt_k^2}{\delta_k^2}\|\text{grad}F(X_k)\|^2 \\
&\leq \frac{(1-\sigma)\omega t_k}{1+\delta_k}\|\text{grad}F(X_k)\|^2
\end{aligned}$$

$$\leq (\sigma - 1)t_k\langle \mathrm{grad}F(X_k), \xi_k\rangle.$$

Combining this and Assumption 1 yields

$$F(R_{X_k}(t_k\xi_k)) - F(X_k) - t_k\langle \mathrm{grad}F(X_k), \xi_k\rangle \leq (\sigma - 1)t_k\langle \mathrm{grad}F(X_k), \xi_k\rangle.$$

So, the Armijo condition (16) is satisfied. Then, (22) follows immediately from the backtracking procedure (lines 8–10) in Algorithm 1. □

Theorem 4.3. *Suppose Assumption 1 holds and let $\{X_k\}_{k\in\mathbb{N}} \subset \mathcal{M}_l$ be a sequence generated by Algorithm 1. If (20) holds, then*

$$\lim_{k\to\infty} \|\mathrm{grad}F(X_k)\| = 0.$$

Moreover, every accumulation point of $\{X_k\}_{k\in\mathbb{N}}$ is a stationary point of f.

Proof. Since Assumption 1 holds, Lemma 4.2 is valid. Using Lemmas 4.1 and 4.2, we have

$$\begin{aligned} F(X_{k+1}) - F(X_k) &\leq \sigma t_k \langle \mathrm{grad}F(X_k), \xi_k\rangle \leq -\frac{\sigma\omega t_k}{1+\delta_k}\|\mathrm{grad}F(X_k)\|^2 \\ &\leq -\frac{\sigma\omega}{1+\delta_k}\min\left\{\underline{t}, \frac{c\delta_k^2}{1+\delta_k}\right\}\|\mathrm{grad}F(X_k)\|^2, \end{aligned} \tag{23}$$

where $c := \frac{\rho(1-\sigma)\omega}{2L}$. Then, we consider the following two cases.

If $\delta_k \geq 1$, it follows from (23) and (17) that

$$\begin{aligned} F(X_{k+1}) - F(X_k) &\leq -\frac{\sigma\omega}{2\delta_k}\min\left\{\underline{t}, \frac{1}{2}c\delta_k\right\}\|\mathrm{grad}F(X_k)\|^2 \\ &= -\frac{\sigma\omega}{2\mu}\min\left\{\underline{t}, \frac{1}{2}c\mu\|\mathrm{grad}F(X_k)\|^\tau\right\}\|\mathrm{grad}F(X_k)\|^{2-\tau}. \end{aligned} \tag{24}$$

If $\delta_k \leq 1$, it follows from (23) and (17) again that

$$\begin{aligned} F(X_{k+1}) - F(X_k) &\leq -\frac{\sigma\omega}{2}\min\left\{\underline{t}, \frac{1}{2}c\delta_k^2\right\}\|\mathrm{grad}F(X_k)\|^2 \\ &= -\frac{\sigma\omega}{2}\min\left\{\underline{t}, \frac{1}{2}c\mu^2\|\mathrm{grad}F(X_k)\|^{2\tau}\right\}\|\mathrm{grad}F(X_k)\|^2. \end{aligned} \tag{25}$$

Combining (24) and (25), we obtain

$$F(X_{k+1}) - F(X_k) \leq -\frac{\sigma\omega}{2}\min\left\{\frac{1}{\mu}\underline{t}\|\mathrm{grad}F(X_k)\|^{2-\tau}, \min\left\{\frac{1}{2}c, \underline{t}\right\}\|\mathrm{grad}F(X_k)\|^2, \frac{1}{2}c\mu^2\|\mathrm{grad}F(X_k)\|^{2+2\tau}\right\}.$$

Since f is bounded below and $\tau \in (0, 2)$, it holds that $\lim_{k\to\infty} \|\mathrm{grad}F(X_k)\| = 0$. □

4.2. Local convergence

In this subsection, we consider the local convergence rate of Algorithm 1. Before our proof, we need the following assumption.

Assumption 2. The sequence $\{X_k\}_{k\in\mathbb{N}}$ generated by Algorithm 1 converges to $X^* \in \mathcal{M}_l$, which satisfies $\mathcal{P}_\Omega(X^* - A) = 0$ and $\mathcal{H}_{X^*} = \mathcal{P}_{T_{X^*}\mathcal{M}_l}\mathcal{P}_\Omega \succ 0$. These conditions on X^* are equivalent to $\nabla F(X^*) = 0$ and $\mathrm{Hess}F(X^*) \succ 0$.

Assumption 2 is used in Lemma 4.5 and Theorem 4.8 below.

For convenience, we define the difference of the true and approximate Hessians at $X_k = U_k\Sigma_k V_k^\top$ as

$$E_k[\xi] := \mathrm{Hess}F(X_k)[\xi] - \mathcal{H}_{X_k}[\xi] = P_{U_k}^\perp \mathcal{P}_\Omega(X_k - A)V_p\Sigma_k^{-1}V_k^\top + U_k\Sigma_k^{-1}U_p^\top\mathcal{P}_\Omega(X_k - A)P_{V_k}^\perp. \tag{26}$$

According to [17], $U_p = P_{U_k}^\perp \xi V_k$ and $V_p = P_{V_k}^\perp \xi^\top U_k$. Therefore, by Assumption 2 and (26), we have

$$\|E_k[\xi]\| = o(\|\xi\|), \tag{27}$$

showing that $\mathcal{H}_{X_k}[\xi]$ approximates $\mathrm{Hess}F(X_k)[\xi]$ well if X_k is close enough to X^* and ξ is small enough.

Lemma 4.4. *The step size t_k^* defined in (15) can be rewritten as*

$$t_k^* = \frac{\langle \xi_k, \mathrm{grad}F(X_k)\rangle}{\langle \xi_k, \mathrm{grad}F(X_k) + \delta_k\xi_k\rangle}. \tag{28}$$

Proof. It follows from (12), (13), (15), and (19) that

$$\begin{aligned} t_k^* &= -\frac{\langle \mathcal{P}_\Omega(\xi_k), \mathcal{P}_\Omega(X_k - A)\rangle}{\langle \mathcal{P}_\Omega(\xi_k), \mathcal{P}_\Omega(\xi_k)\rangle} = -\frac{\langle \xi_k, \mathcal{P}_\Omega(X_k - A)\rangle}{\langle \xi_k, \mathcal{P}_\Omega(\xi_k)\rangle} \\ &= -\frac{\langle \xi_k, \mathcal{P}_{T_{X_k}\mathcal{M}_l}\mathcal{P}_\Omega(X_k - A)\rangle}{\langle \xi_k, \mathcal{P}_{T_{X_k}\mathcal{M}_l}\mathcal{P}_\Omega(\xi_k)\rangle} = \frac{\langle \xi_k, \mathrm{grad}F(X_k)\rangle}{\langle \xi_k, -\mathcal{H}_{X_k}[\xi_k]\rangle} \\ &= \frac{\langle \xi_k, \mathrm{grad}F(X_k)\rangle}{\langle \xi_k, -\mathcal{H}_{X_k,\delta_k}[\xi_k] + \delta_k\xi_k\rangle} \\ &= \frac{\langle \xi_k, \mathrm{grad}F(X_k)\rangle}{\langle \xi_k, \mathrm{grad}F(X_k) - r_k + \delta_k\xi_k\rangle}. \end{aligned}$$

Because ξ_k is initialized as 0 according to our CG implementation (Algorithms 2 and 3), a basic property of CG shows that

$$\langle \xi_k, r_k\rangle = 0. \tag{29}$$

Hence, (28) follows immediately. □

Lemma 4.5. *Suppose Assumption 2 holds. Then, $t_k^{\mathrm{ini}} = t_k^*$ for all sufficiently large k and $\lim_{k\to\infty} t_k^* = 1$.*

Proof. It follows from Assumption 2 that $\left\|(\mathcal{H}_{X_k} + \delta_k\mathcal{I})^{-1}\right\| = O(1)$. Then, by (13) and (17)–(19), we have

$$\|\xi_k\| = \left\|(\mathcal{H}_{X_k} + \delta_k\mathcal{I})^{-1}[-\mathrm{grad}F(X_k) + r_k]\right\| = O(\|\mathrm{grad}F(X_k)\|) \tag{30}$$

and

$$\langle \xi_k, \mathrm{grad}F(X_k)\rangle = \left\langle (\mathcal{H}_{X_k} + \delta_k\mathcal{I})^{-1}[-\mathrm{grad}F(X_k) + r_k], \mathrm{grad}F(X_k)\right\rangle = O(\|\mathrm{grad}F(X_k)\|^2). \tag{31}$$

Additionally, (30) together with (17) implies

$$\langle \xi_k, \delta_k \xi_k \rangle = o(\|\text{grad}F(X_k)\|^2). \tag{32}$$

Then, it follows from (28), (31), and (32) that

$$\lim_{k\to\infty} t_k^* = \lim_{k\to\infty} \frac{\langle \xi_k, \text{grad}F(X_k)\rangle}{\langle \xi_k, \text{grad}F(X_k) + \delta_k \xi_k\rangle} = 1.$$

Since $t_k^{\text{ini}} = \max\{\underline{t}, t_k^*\}$ and $\underline{t} \in (0,1)$ according to Algorithm 1, it holds that $t_k^{\text{ini}} = t_k^*$ for all sufficiently large k. □

For technical use, we need to introduce a kind of Newton–Leibniz formula for gradient fields, which involves the following basic concept in Riemannian geometry [20].

Definition 4.6 (parallel transport)**.** Let $\mathcal{M}$ be a Riemannian manifold and ∇ the Riemannian connection. A smooth vector field V along a smooth curve γ is said to be parallel along γ if $\nabla V[\gamma'(t)] = 0$. Given $\xi_a = T_{\gamma(a)}\mathcal{M}$, there is a unique parallel vector field ξ along γ such that $\xi(a) = \xi_a$. The operator $P_\gamma^{b\leftarrow a}$ sending $\xi(a)$ to $\xi(b)$ is called the parallel transport along γ.

Parallel transport has many good geometric properties, e.g., it preserves the norm of a vector field along a curve.

Lemma 4.7. *Let γ be the curve defined by $\gamma(t) := R_{X_k}(t\xi_k)$. Then,*

$$P_\gamma^{0\leftarrow t_k}\text{grad}F(X_{k+1}) - \text{grad}F(X_k) = \int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[\gamma'(t)]\text{d}t.$$

Proof. This result can be found in the literature, e.g., in the proof of Lemma 8 in [15]. But, for self-containedness, we provide a clear proof in Appendix B. □

Theorem 4.8. *Suppose Assumption 2 holds and let $\{X_k\}_{k\in\mathbb{N}} \subset \mathcal{M}_l$ be a sequence generated by Algorithm 1. If $\theta_k = o(1)$, then $\{\|\text{grad}F(X_k)\|\}_{k\in\mathbb{N}}$ converges superlinearly to 0 and $\{X_k\}_{k\in\mathbb{N}}$ converges superlinearly to X^*, i.e.,*

$$\|\text{grad}F(X_{k+1})\| = o\left(\|\text{grad}F(X_k)\|\right)$$

and

$$\text{dist}(X_{k+1}, X^*) = o\left(\text{dist}(X_k, X^*)\right).$$

If $\{\theta_k\}_{k\in\mathbb{N}} \subset (0, \bar{\theta}]$ for some constant $\bar{\theta} \in (0,1)$, then $\{\|\text{grad}F(X_k)\|\}_{k\in\mathbb{N}}$ converges at least Q-linearly to 0 and $\{X_k\}_{k\in\mathbb{N}}$ converges at least R-linearly to X^*, i.e.,

$$\|\text{grad}F(X_{k+1})\| \le \widetilde{\theta}\|\text{grad}F(X_k)\|$$

and

$$\text{dist}(X_k, X^*) \le C\widetilde{\theta}^k \text{dist}(X_0, X^*)$$

for all sufficiently large k, where $\widetilde{\theta} \in (\bar{\theta}, 1)$ and $C > 0$ are constants.

Proof. First, we prove that the initial stepsize t_k^{ini} is always accepted for all sufficiently large k. Since Assumption 4.2 holds, Lemma 4.5 is valid. Then, it follows from Lemma 4.5, (10), (11), (13), (19), (26), (27), and (29)–(32) that

$$\begin{aligned}
F(R_{X_k}(t_k^{\text{ini}}\xi_k)) - F(X_k) =& t_k^*\langle \text{grad}F(X_k), \xi_k\rangle + \frac{1}{2}(t_k^*)^2\langle \text{Hess}F(X_k)[\xi_k], \xi_k\rangle + o(\|\xi_k\|^2)\\
=& t_k^*\langle \text{grad}F(X_k), \xi_k\rangle + \frac{1}{2}(t_k^*)^2\langle \mathcal{H}_{X_k,\delta_k}[\xi_k] - \delta_k\xi_k + E_k[\xi_k], \xi_k\rangle + o(\|\xi_k\|^2)\\
=& t_k^*\langle \text{grad}F(X_k), \xi_k\rangle + \frac{1}{2}(t_k^*)^2\langle -\text{grad}F(X_k) + r_k - \delta_k\xi_k + E_k[\xi_k], \xi_k\rangle + o(\|\xi_k\|^2)\\
=& t_k^*\langle \text{grad}F(X_k), \xi_k\rangle + \frac{1}{2}(t_k^*)^2\langle -\text{grad}F(X_k) - \delta_k\xi_k + E_k[\xi_k], \xi_k\rangle + o(\|\xi_k\|^2)\\
=& (t_k^* - \frac{1}{2}(t_k^*)^2)\langle \text{grad}F(X_k), \xi_k\rangle + \frac{1}{2}(t_k^*)^2\langle E_k[\xi_k] - \delta_k\xi_k, \xi_k\rangle + o(\|\xi_k\|^2)\\
=& \frac{1}{2}\langle \text{grad}F(X_k), \xi_k\rangle + o(\|\text{grad}F(X_k)\|^2).
\end{aligned}$$

The above result together with $\sigma \in (0, \frac{1}{2})$ yields

$$F(R_{X_k}(t_k^{\text{ini}}\xi_k)) - F(X_k) \le \sigma t_k^*\langle \text{grad}F(X_k), \xi_k\rangle,$$

implying that t_k^{ini} satisfies the Armijo condition (16), for all sufficiently large k.

Second, we prove the local convergence rate of $\{\|\text{grad}F(X_k)\|\}_{k\in\mathbb{N}}$ and $\{X_k\}_{k\in\mathbb{N}}$. Let $\gamma(t)$ be the curve in Lemma 4.7. Then, it follows from (13), (17)–(19), (26), (27), (30), and Lemmas 4.5 and 4.7 that

$$\begin{aligned}
&\|\text{grad}F(X_{k+1})\|\\
=&\|P_\gamma^{0\leftarrow t_k}\text{grad}F(X_{k+1})\| = \left\|\text{grad}F(X_k) + \int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[\gamma'(t)]\text{d}t\right\|\\
\le& \left\|\text{grad}F(X_k) + \int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[P_\gamma^{t\leftarrow 0}\xi_k]\text{d}t\right\| + \left\|\int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[P_\gamma^{t\leftarrow 0}\xi_k - \gamma'(t)]\text{d}t\right\|\\
\le&\|\text{grad}F(X_k) + \mathcal{H}_{X_k,\delta_k}[\xi_k]\| + \left\|\int_0^{t_k}\left(\frac{1}{t_k}\mathcal{H}_{X_k,\delta_k} - P_\gamma^{t\leftarrow 0}\text{Hess}F(\gamma(t))P_\gamma^{t\leftarrow 0}\right)[\xi_k]\text{d}t\right\|\\
&+ \left\|\int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[P_\gamma^{t\leftarrow 0}\xi_k - \gamma'(t)]\text{d}t\right\|\\
\le&\|r_k\| + \|(\mathcal{H}_{X_k,\delta_k} - t_k\text{Hess}F(X_k))[\xi_k]\| + \left\|\int_0^{t_k}\left(\text{Hess}F(X_k) - P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))P_\gamma^{t\leftarrow 0}\right)[\xi_k]\text{d}t\right\|\\
&+ \left\|\int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[P_\gamma^{t\leftarrow 0}\xi_k - \gamma'(t)]\text{d}t\right\|\\
\le&\|r_k\| + \|((1-t_k)\text{Hess}F(X_k) - E_k + \delta_k\mathcal{I})[\xi_k]\| + \left\|\int_0^{t_k}(\text{Hess}F(X_k) - P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))P_\gamma^{t\leftarrow 0})[\xi_k]\text{d}t\right\|\\
&+ \left\|\int_0^{t_k} P_\gamma^{0\leftarrow t}\text{Hess}F(\gamma(t))[P_\gamma^{t\leftarrow 0}\xi_k - \gamma'(t)]\text{d}t\right\|\\
\le&\theta_k\|\text{grad}F(X_k)\| + o(\|\xi_k\|) + b_1\|\xi_k\|^2 + b_2\|\xi_k\|^2
\end{aligned}$$

$$=\theta_k\|\text{grad}F(X_k)\| + o(\|\text{grad}F(X_k)\|) \tag{33}$$

for all sufficiently large k, where the final inequality, with two positive constants b_1 and b_2, uses Lemmas 3, 4, and 8 in [15].

Now we consider two cases. If $\theta_k = o(1)$, it follows from (33) that

$$\|\text{grad}F(X_{k+1})\| = o(\|\text{grad}F(X_k)\|). \tag{34}$$

Additionally, by Theorem 7.4.8 in [3], there exist two positive constants c_0 and c_1 such that

$$c_0\text{dist}(X_k, X^*) \leq \|\text{grad}F(X_k)\| \leq c_1\text{dist}(X_k, X^*) \tag{35}$$

for all sufficiently large k. Combining (34) and (35) yields

$$\text{dist}(X_{k+1}, X^*) = o\left(\text{dist}(X_k, X^*)\right).$$

If $\{\theta_k\}_{k\in\mathbb{N}} \subset (0, \overline{\theta}]$ for some $\overline{\theta} \in (0, 1)$, it follows from (33) that

$$\|\text{grad}F(X_{k+1})\| \leq \widetilde{\theta}\|\text{grad}F(X_k)\| \tag{36}$$

for all sufficiently large k, where $\widetilde{\theta} \in (\overline{\theta}, 1)$. Combining (35) and (36), we can find a positive integer K such that for all $k \geq K$, it holds that

$$\begin{aligned}\text{dist}(X_k, X^*) &\leq \frac{1}{c_0}\|\text{grad}F(X_k)\| \leq \frac{1}{c_0}\widetilde{\theta}^{k-K}\|\text{grad}F(X_K)\| \\ &\leq \frac{c_1}{c_0}\widetilde{\theta}^{k-K}\text{dist}(X_K, X^*) \leq C\widetilde{\theta}^k\text{dist}(X_0, X^*),\end{aligned}$$

where $C := \frac{c_1\widetilde{\theta}^{-K}\text{dist}(X_K,X^*)}{c_0\text{dist}(X_0,X^*)}$. The proof is complete. □

5. Numerical experiments

In this section, we report the numerical results of Algorithm 1 and several existing algorithms. All numerical experiments were conducted in MATLAB R2024b on a workstation equipped with an Intel® Core™ i9-14900K processor (24 cores @ 3.19 GHz) and 64 GB RAM, running Windows 11 Professional.

5.1. Experimental details

We compare Algorithm 1, denoted as RRGN, against several Riemannian optimization algorithms, including:

- RRN: a Riemannian regularized Newton method;
- RCG: a Riemannian conjugate gradient method [33];
- RGD: a Riemannian gradient method;
- ARNT*: the adaptively regularized Newton method in [14].

*https://github.com/optsuite/ARNT

RRN, RCG, and RGD were programmed by us. RRN replaces the approximate Riemannian Hessian in RRGN with the true Riemannian Hessian. RCG is an implementation of the algorithm in [33] without the C language code `part_XY.c` for partial matrix multiplication (to ensure a fair comparison in a pure MATLAB environment). RGD is just an ordinary Riemannian gradient descent method. ARNT is the original code in [14]. In RRGN and RRN, Algorithm 3 is invoked as the CG solver for the Newton-type equations. In RRN and ARNT, the true Riemannian Hessian is used.

In our experiments, we performed the above algorithms on synthetic random instances. Given m, n, and l, the tested matrices A were generated as $A = LR^\top$, where $L \in \mathbb{R}^{m\times l}$ and $R \in \mathbb{R}^{n\times l}$ were random matrices with entries sampled independently from the standard normal distribution. The observation set Ω was generated by uniform random sampling, with the sampling density determined by an oversampling factor $\mathsf{OS} = 3$, where

$$\mathsf{OS} = \frac{|\Omega|}{l(2n-l)}.$$

The implementation leverages MATLAB's `sprandn` function to generate a sparse random matrix. The initial point $X_0 \in \mathcal{M}_l$ was set as $X_0 = U_0V_0^\top$, where U_0 and V_0 were matrices with orthonormal columns obtained randomly via `orth(randn(·))`.

The stopping criterion for the first four algorithms is

$$\|\mathrm{grad}F(X_k)\| < 10^{-11}. \tag{37}$$

ARNT was implemented with its default stopping criterion, which is that one of (37) and other conditions is satisfied. The parameters in RRGN were set to $\mu = 10^{-4}$, $\tau = 1$, $\theta_k = 0.1$, $\sigma = 10^{-8}$, $\rho = 0.2$, $\underline{t} = 10^{-10}$, and $\sigma = 10^{-8}$. In RRN, we set $\tau = 0.3$ and replaced μ with $\mu_k = 10^{-6} + 300 \times (0.6)^k$ since we found it hard to fix μ while getting good numerical results. The other parameters of RRN were the same as those of RRGN.

5.2. Numerical results on synthetic problems

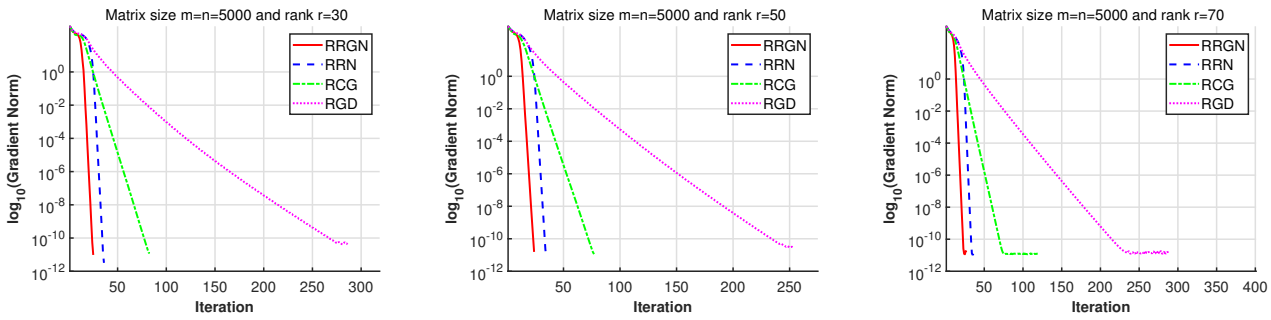
In this subsection, we evaluate the performance of the above algorithms through randomly generated synthetic problems. To rigorously assess the performance of the algorithms, the matrices were randomly generated with various sizes and ranks, and each experimental setting was tested on 10 random instances. In Tables 1–3, average results of all 10 random trials of the following items are reported. **Time(s)** represents the average CPU time (in seconds) that the algorithms spent to reach the stopping criterion. **NormGrad** represents the average norm of the Riemannian gradient at the final iterate. **Recovery** represents the average absolute error of the final iterate X_k, which is defined as recovery $= \|X_k - A\|$. **Iteration** represents the average number of outer iterations. The numbers in parentheses in RRGN, RRN, and ARNT are the total numbers of inner iterations for solving subproblems throughout the whole process. Note that the numbers of iterations of ARNT are very small because the iteration couting of ARNT is quite different from those of the other four algorithms

Table 1 shows the results of matrix size 5000×5000, Table 2 shows the results of matrix size 7500×7500, and Table 3 shows the results of matrix size 10000×10000. The ranks in these tables were chosen as 30, 50, and 70. In addition, Figures 1–3 (corresponding to Tables 1–3, respectively) show the plots of $\log_{10}(\|\mathrm{grad}F(X_k)\|)$ versus iterations of the first four algorithms. We did not plot such a curve for ARNT because its iteration counting is quite different.

We highlight the top performer for each setting in Tables 1–3 using bold text. It can be observed that RRGN performed best in all our cases. RRGN reached the stopping criterion within the least CPU time for all cases. The numbers of outer iterations of RRGN ranged from 23 to 29. RRN and ARNT are Newton-type algorithms as well, but use the true Riemannian Hessian. They performed slightly worse than RRGN. There might be two major reasons. One is that the true Riemannian Hessian has higher computational cost than that of the approximate Riemannian Hessian. The other is that it is more difficult to choose a good regularization parameter in the two algorithms because the true Riemannian Hessian is possibly non-positive semidefinite according to Proposition 3.3. RGD was obviously the worst one among all the five algorithms. RCG performed similarly to RRGN for ranks 30 and 50 and unsatisfactorily for rank 70. According to case (c) in Figures 1–3, the norm of the Riemannian gradient of RCG fluctuated and hardly made progress when it was close to 10^{-11}. The reason might be that RCG only utilizes first-order information like RGD.

Table 1. Numerical comparison for matrix size $m = n = 5000$.

Rank	Algorithm	Time(s)	NormGrad	Recovery	Iteration
30	**RRGN**	**23.6**	**5.01e-12**	**3.36e-10**	**23.7(108.4)**
	RRN	27.2	5.65e-12	2.95e-10	34.5(96.6)
	RCG	24.2	8.17e-12	5.58e-10	81.7
	RGD	70.7	9.43e-12	1.38e-09	285.0
	ARNT	29.1	6.39e-12	4.42e-10	2.8(12.0)
50	**RRGN**	**23.6**	**6.95e-12**	**1.69e-10**	**23.1(96.6)**
	RRN	28.1	6.47e-12	1.26e-10	34.0(91.2)
	RCG	26.2	8.95e-12	3.13e-10	76.2
	RGD	69.6	9.62e-12	5.94e-10	251.7
	ARNT	29.2	7.48e-12	2.67e-10	3.2(11.4)
70	**RRGN**	**30.6**	**9.56e-12**	**1.15e-10**	**25.6(101.0)**
	RRN	35.7	9.60e-12	1.12e-10	35.4(97.6)
	RCG	56.3	9.74e-12	1.34e-10	118.3
	RGD	99.0	9.78e-12	1.66e-10	286.7
	ARNT	32.7	9.94e-12	1.79e-10	4.8(15.4)



(a) rank r=30 **(b)** rank r=50 **(c)** rank r=70

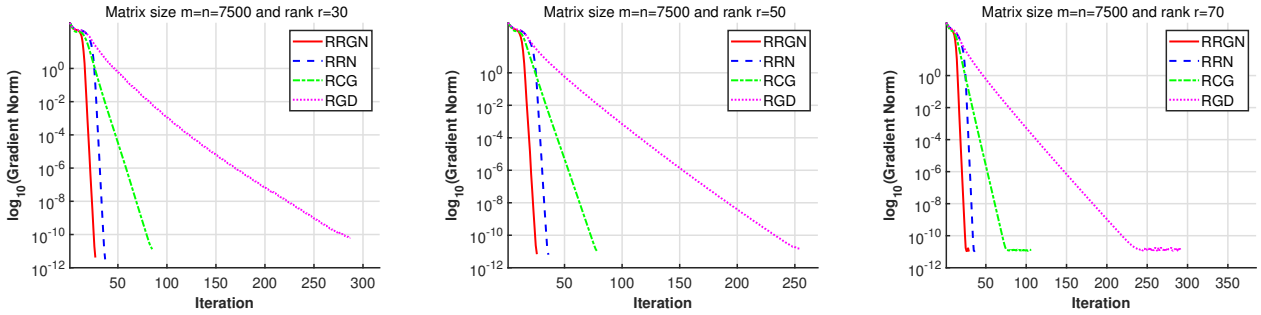
Figure 1. Gradient norm convergence for matrix size $m = n = 5000$.

Table 2. Numerical comparison for matrix size $m = n = 7500$.

Rank	Algorithm	Time(s)	NormGrad	Recovery	Iteration
30	**RRGN**	**48.9**	**4.98e-12**	**3.75e-10**	**25.5(109.3)**
	RRN	56.8	5.19e-12	4.30e-10	35.6(101.8)
	RCG	49.5	8.24e-12	8.33e-10	83.6
	RGD	143.8	9.60e-12	2.15e-09	285.6
	ARNT	63.1	8.23e-12	8.79e-10	3.0(14.0)
50	**RRGN**	**49.0**	**7.22e-12**	**2.77e-10**	**24.5(95.7)**
	RRN	58.4	6.99e-12	2.61e-10	34.7(92.8)
	RCG	52.0	8.72e-12	4.52e-10	77.2
	RGD	142.7	9.48e-12	8.54e-10	253.8
	ARNT	588	8.53e-12	4.96e-10	3.1(11.8)
70	**RRGN**	**64.2**	**9.66e-12**	**1.54e-10**	**27.9(108.7)**
	RRN	69.7	9.55e-12	1.62e-10	36.1(96.8)
	RCG	91.1	9.87e-12	2.01e-10	105.0
	RGD	196.6	9.83e-12	2.35e-10	289.5
	ARNT	68.4	1.16e-11	2.53e-10	5.6(15.9)

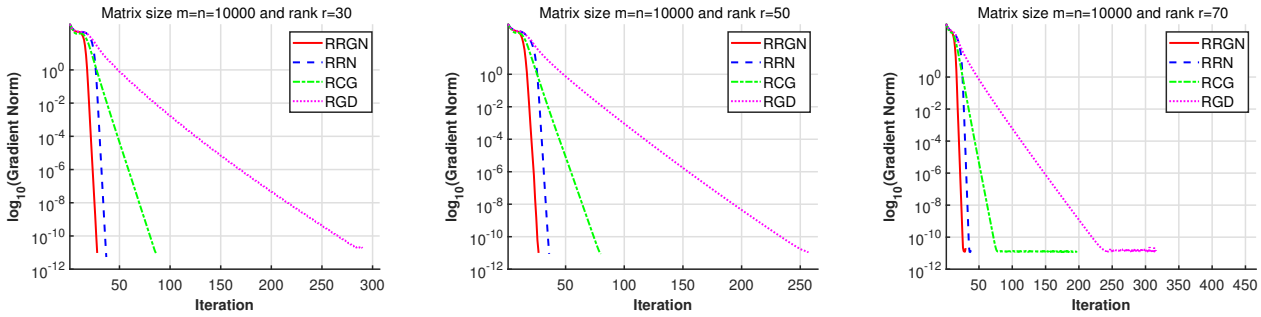
Table 3. Numerical comparison for matrix size $m = n = 10000$.

Rank	Algorithm	Time(s)	NormGrad	Recovery	Iteration
30	**RRGN**	**81.7**	**5.16e-12**	**5.64e-10**	**27.2(106.9)**
	RRN	91.5	5.41e-12	6.09e-10	36.0(100.0)
	RCG	83.4	8.26e-12	1.10e-09	84.9
	RGD	241.9	9.60e-12	2.68e-09	288.8
	ARNT	123.5	8.38e-12	1.17e-09	2.7(12.0)
50	**RRGN**	**84.3**	**6.56e-12**	**2.48e-10**	**26.0(99.2)**
	RRN	95.7	7.91e-12	4.05e-10	35.1(93.2)
	RCG	87.2	8.79e-12	5.94e-10	78.7
	RGD	240.8	9.63e-12	1.12e-09	256.2
	ARNT	102.7	8.22e-12	7.12e-10	3.4(12.6)
70	**RRGN**	**105.0**	**9.60e-12**	**2.15e-10**	**28.8(104.5)**
	RRN	124.5	9.68e-12	2.20e-10	38.9(102.5)
	RCG	311.9	9.87e-12	2.59e-10	196.0
	RGD	358.3	9.88e-12	3.13e-10	314.8
	ARNT	115.8	1.13e-11	3.47e-10	5.5(15.1)



(a) rank r=30 **(b)** rank r=50 **(c)** rank r=70

Figure 2. Gradient norm convergence for matrix size $m = n = 7500$.



(a) rank r=30 **(b)** rank r=50 **(c)** rank r=70

Figure 3. Gradient norm convergence for matrix size $m = n = 10000$.

5.3. Numerical results on image recovery

In this subsection, we evaluate the performance of the above algorithms through an image recovery task. To adapt the image to our experiments, we performed scaling and rank truncation on the original image, constructing three matrix recovery problems of different scales. Specifically, we considered three scenarios: matrix sizes $m = n = 2500$ with rank $r = 50$, matrix size $m = n = 5000$ with rank $r = 70$, and matrix size $m = n = 7500$ with rank $r = 100$. In each experiment, we randomly sampled 20% of the data for image recovery.

Figure 4 presents the visual recovery results of all five algorithms alongside the degraded image in one trial of random sampling with $m = n = 5000$ and $r = 70$. All methods successfully recovered the original image from severely incomplete data (80% missing entries), demonstrating their effectiveness for practical image recovery tasks. The visual recovery results are similar in the other two settings, so we do not display them.

Tables 4–6 show average results of 5 trials of random sampling in each case. These results demonstrate the superiority of the proposed method. Compared to RCG and RGD, RRGN achieved significantly faster convergence, which can be attributed to the advantage of utilizing second-order information when facing high-precision stopping criteria. When compared to the other second-order algorithms, RRN and ARNT, RRGN shows better performance in terms of computational efficiency. This improvement may stem from the reduced computational cost of the Gauss–Newton approximation of Hessian. Moreover, the positive semidefiniteness of the approximate Hessian makes

it easier to choose the regularization parameter. The gradient norm convergence curves in Figure 5 further verify these observations.

(a) Degraded image **(b)** RRGN **(c)** RRN

(d) RCG **(e)** RGD **(f)** ARNT

Figure 4. Visual comparison of recovery results by different algorithms for $m = n = 5000$ and $r = 70$.

Table 4. Numerical comparison for matrix size $m = n = 2500$ and rank $r = 50$.

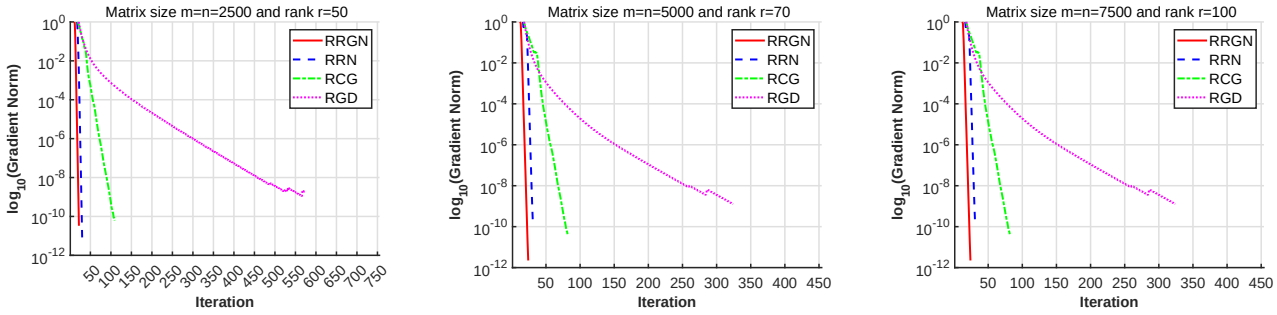
Algorithm	Time(s)	NormGrad	Recovery	Iteration
RRGN	**10.1**	**6.57e-12**	**1.57e-10**	**21.4(140.4)**
RRN	13.6	4.43e-12	1.54e-10	29.2(131.6)
RCG	13.0	8.72e-12	2.12e-10	107.8
RGD	53.9	9.63e-12	7.67e-10	571.8
ARNT	12.4	7.55e-12	2.62e-10	3.0(18.6)

Table 5. Numerical comparison for matrix size $m = n = 5000$ and rank $r = 70$.

Algorithm	Time(s)	NormGrad	Recovery	Iteration
RRGN	**43.8**	**1.98e-12**	**4.72e-11**	**22.4(107.4)**
RRN	49.4	4.36e-12	1.03e-10	28.0(97.2)
RCG	45.1	8.41e-12	1.38e-10	77.2
RGD	146.1	9.35e-12	4.26e-10	304.4
ARNT	45.5	7.93e-12	2.10e-10	2.8(13.6)

Table 6. Numerical comparison for matrix size $m = n = 7500$ and rank $r = 100$.

Algorithm	Time(s)	NormGrad	Recovery	Iteration
RRGN	**99.1**	**5.00e-12**	**7.05e-11**	**22.6(107.8)**
RRN	135.2	7.58e-12	9.32e-11	29.8(114.6)
RCG	115.6	7.48e-12	1.45e-10	81.4
RGD	358.3	9.40e-12	4.89e-10	322.6
ARNT	115.9	6.82e-12	1.39e-10	2.6(11.0)



(a) matrix size 2500×2500 **(b)** matrix size 5000×5000 **(c)** matrix size 7500×7500.

Figure 5. Gradient norm convergence.

6. Conclusions

We proposed and analyzed a Riemannian regularized Gauss–Newton method for low-rank matrix completion. Inspired by Gauss–Newton methods, the true Riemannian Hessian in Newton's method was replaced with a simple symmetric positive semidefinite operator. Adding a regularization term, we solved the corresponding Riemannian Gauss–Newton equation by the linear CG method. The approximate Riemannian Hessian has several advantages. First, it has lower computational cost than the true Riemannian Hessian. Second, it approximates the true Riemannian Hessian when the observed entries of the variable matrix are close to those of the data matrix. Third, the positive semidefiniteness of the approximate Riemannian Hessian makes it easy to choose an appropriate regularization parameter. We have also established global and local convergence of our method under mild assumptions. Numerical results on synthetic and image recovery problems demonstrate the efficiency of our method.

Author contributions

Xiaojing Zhu: Conceptualization, Methodology, Formal analysis, Writing-original draft, Writing-review & editing. Fengyi Yuan: Software, Investigation, Validation, Writing-original draft, Writing-review & editing. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare that they have used Artificial Intelligence (AI) tools in the creation of this article. AI tools used: Code for creating tables and figures was written with the assistance of generative AI tools (DeepSeek and ChatGPT).

Acknowledgments

This research was financially supported by the National Natural Science Foundation of China (Grant Nos. 12271342 and 11601317) and Shanghai Oriental Talents Program (Grant No. QNJY2024199).

The authors are very grateful to the three anonymous referees for their constructive and insightful comments that significantly improved the quality of this paper.

Code availability

The matlab code used for the numerical experiments in this paper is available at https://github.com/xjzhu2013/RRGNLRMC.

Conflict of interest

The authors declare no conflicts of interest.

References

1. P. A. Absil, L. Amodei, G. Meyer, Two Newton methods on the manifold of fixed-rank matrices endowed with Riemannian quotient geometries, *Comput. Stat.*, **29** (2014), 569–590. https://doi.org/10.1007/s00180-013-0441-6
2. P. A. Absil, C. G. Baker, K. A. Gallivan, Trust-region methods on Riemannian manifolds, *Found. Comput. Math.*, **7** (2007), 303–330. https://doi.org/10.1007/s10208-005-0179-9
3. P. A. Absil, R. Mahony, R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton: Princeton University Press, 2008.
4. P. A. Absil, J. Malick, Projection-like retractions on matrix manifolds, *SIAM J. Optim.*, **22** (2012), 135–158. https://doi.org/10.1137/100802529
5. P. A. Absil, I. V. Oseledets, Low-rank retractions: a survey and new results, *Comput. Optim. Appl.*, **62** (2015), 5–29. https://doi.org/10.1007/s10589-014-9714-4
6. N. Boumal, *An Introduction to Optimization on Smooth Manifolds*, Cambridge: Cambridge University Press, 2023.
7. N. Boumal, P. A. Absil, Low-rank matrix completion via preconditioned optimization on the Grassmann manifold, *Linear Algebra Appl.*, **475** (2015), 200–239. https://doi.org/10.1016/j.laa.2015.02.027
8. J. F. Cai, E. J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, *SIAM J. Optim.*, **20** (2010), 1956–1982. https://doi.org/10.1137/080738970

9. J. F. Cai, W. Huang, H. Wang, K. Wei, Tensor completion via tensor train based low-rank quotient geometry under a preconditioned metric, preprint paper, 2022. https://doi.org/10.48550/arXiv.2209.04786
10. W. Dai, E. Kerman, O. Milenkovic, A geometric approach to low-rank matrix completion, *IEEE Trans. Inform. Theory*, **58** (2012), 237–247. https://doi.org/10.1109/TIT.2011.2171521
11. Y. Deng, T. Mu, Faster Riemannian Newton-type optimization by subsampling and cubic regularization, *Mach. Learn.*, **112** (2023), 3527–3589. https://doi.org/10.1007/s10994-023-06321-0
12. B. Gao, P. A. Absil, A Riemannian rank-adaptive method for low-rank matrix completion, *Comput. Optim. Appl.*, **81** (2022), 67–90. https://doi.org/10.1007/s10589-021-00328-w
13. B. Gao, R. Peng, Y. Yuan, Low-rank optimization on Tucker tensor varieties, *Math. Program.*, **214** (2025), 357–407. https://doi.org/10.1007/s10107-024-02186-w
14. J. Hu, A. Milzarek, Z. Wen, Y. Yuan, Adaptive quadratically regularized Newton method for Riemannian optimization, *SIAM J. Optim.*, **39** (2018), 1181–1207. https://doi.org/10.1137/17M1142478
15. W. Huang, P. A. Absil, K. A. Gallivan, A Riemannian symmetric rank-one trust-region method, *Math. Program.*, **150** (2015), 179–216. https://doi.org/10.1007/s10107-014-0765-1
16. P. Jain, P. Netrapalli, S. Sanghavi, Low-rank matrix completion using alternating minimization, In: *Proceedings of the 45th annual ACM symposium on Theory of Computing*, 2013, 665–674. https://doi.org/10.1145/2488608.2488693
17. O. Koch, C. Lubich, Dynamical low-rank approximation, *SIAM J. Matrix Anal. Appl.*, **29** (2007), 434–454. https://doi.org/10.1137/050639703
18. D. Kressner, M. Steinlechner, B. Vandereycken, Low-rank tensor completion by Riemannian optimization, *BIT Numer. Math.*, **54** (2014), 447–468. https://doi.org/10.1007/s10543-013-0455-z
19. D. Kressner, M. Steinlechner, B. Vandereycken, Preconditioned low-rank Riemannian optimization for linear systems with tensor product structure, *SIAM J. Sci. Comput.*, **38** (2016), A2018–A2044. https://doi.org/10.1137/15M1032909
20. J. M. Lee, *Introducton to Riemannian Manifolds*, 2 Eds., Berlin: Springer, 2018.
21. S. Ma, D. Goldfarb, L. Chen, Fixed point and Bregman iterative methods for matrix rank minimization, *Math. Program.*, **128** (2011), 321–353. https://doi.org/10.1007/s10107-009-0306-5
22. S. Mao, L. Xiong, L. Jiao, T. Feng, S. K. Yeung, A novel Riemannian metric based on Riemannian structure and scaling information for fixed low-rank matrix completion, *IEEE Trans. Cybern.*, **47** (2017), 1299–1312. https://doi.org/10.1109/TCYB.2016.2587825
23. B. Mishra, G. Meyer, S. Bonnabel, R. Sepulchre, Fixed-rank matrix factorizations and Riemannian low-rank optimization, *Comput. Stat.*, **29** (2014), 591–621. https://doi.org/10.1007/s00180-013-0464-z
24. L. T. Nguyen, J. Kim, B. Shim, Low-rank matrix Completion: A contemporary survey, *IEEE Access*, **7** (2019), 94215–94237. https://doi.org/10.1109/ACCESS.2019.2928130

25. G. Olikier, A. Uschmajew, B. Vandereycken, Gauss-Southwell type descent methods for low-rank matrix optimization, *J. Optim. Theory Appl.*, **206** (2025), 6. https://doi.org/10.1007/s10957-025-02682-9

26. M. V. Rakhuba, I. V. Oseledets, Jacobi-Davidson method on low-rank matrix manifolds, *SIAM J. Sci. Comput.*, **40** (2018), A1149–A1170. https://doi.org/10.1137/17M1123080

27. B. Recht, M. Fazel, P. A. Parrilo, Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, *SIAM Rev.*, **52** (2010), 471–501. https://doi.org/10.1137/070697835

28. H. Sato, *Riemannian Optimization and Its Applications*, Switzerland: Springer, 2021.

29. A. Séguin, D. Kressner, Continuation methods for Riemannian optimization, *SIAM J. Optim.*, **32** (2022), 1069–1093. https://doi.org/10.1137/21M1428650

30. G. J. Song, M. K. Ng, Nonnegative low rank matrix completion by Riemannian optimalization methods, *Ann. Appl. Math.*, **39** (2023), 181–205. https://doi.org/10.4208/aam.OA-2023-0010

31. M. Steinlechner, Riemannian optimization for high-dimensional tensor completion, *SIAM J. Sci. Comput.*, **38** (2016), S461–S484. https://doi.org/10.1137/15M1010506

32. J. Tanner, K. Wei, Low rank matrix completion by alternating steepest descent methods, *Appl. Comput. Harmon. Anal.*, **40** (2016), 417–429. https://doi.org/10.1016/j.acha.2015.08.003

33. B. Vandereycken, Low-rank matrix completion by Riemannian optimization, *SIAM J. Optim.*, **23** (2013), 1214–1236. https://doi.org/10.1137/110845768

34. B. Vandereycken, S. Vandewalle, A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations, *SIAM J. Matrix Anal. Appl.*, **31** (2010), 2553–2679. https://doi.org/10.1137/090764566

35. Z. Wen, W. Yin, Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, *Math. Program. Comput.*, **4** (2012), 333–361. https://doi.org/10.1007/s12532-012-0044-1

36. S. Zheng, W. Huang, B. Vandereycken, X. Zhang, Riemannian optimization using three differentmetrics for Hermitian PSD fixed-rank constraints, *Comput. Optim. Appl.*, **91** (2025), 1135–1184. https://doi.org/10.1007/s10589-025-00687-8

37. J. Zhou, K. Deng, H. Wang, Z. Peng, Inexact Riemannian gradient descent method for nonconvex optimization with strong convergence, *J. Sci. Comput.*, **103** (2025), 96. https://doi.org/10.1007/s10915-025-02913-1

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

A. Proof of Proposition 2.1

Proof. From (5)–(7), we have

$$\mathrm{Hess}F(X)[\xi] = \mathcal{P}_{T_X\mathcal{M}_l}\mathrm{Dgrad}F(X)[\xi] = \mathcal{P}_{T_X\mathcal{M}_l}\mathrm{D}\mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_{\Omega}(X-A)[\xi]. \tag{A.1}$$

The defining formula (4) of $\mathcal{P}_{T_X\mathcal{M}_l}$ reads

$$\mathcal{P}_{T_X\mathcal{M}_l}(Z) = P_U Z P_V + P_U^\perp Z P_V + P_U Z P_V^\perp. \tag{A.2}$$

In what follows, we denote the directional derivative of a variable along ξ by a dot notation for convenience, e.g., $\mathrm{D}U[\xi] = \dot{U}$.

Using (A.2), we have

$$\begin{aligned}
\mathrm{D}\mathcal{P}_{T_X\mathcal{M}_l}\mathcal{P}_\Omega(X-A)[\xi] &= \dot{P}_U\mathcal{P}_\Omega(X-A)P_V + P_U\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(X-A)\dot{P}_V \\
&\quad - \dot{P}_U\mathcal{P}_\Omega(X-A)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(X-A)\dot{P}_V \\
&\quad + \dot{P}_U\mathcal{P}_\Omega(X-A)P_V^\perp + P_U\mathcal{P}_\Omega(\xi)P_V^\perp - P_U\mathcal{P}_\Omega(X-A)\dot{P}_V \\
&= P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp \\
&\quad + P_U^\perp\mathcal{P}_\Omega(X-A)\dot{P}_V + \dot{P}_U\mathcal{P}_\Omega(X-A)P_V^\perp \\
&= P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp \\
&\quad + P_U^\perp\mathcal{P}_\Omega(X-A)(\dot{V}V^\top + V\dot{V}^\top) + (\dot{U}U^\top + U\dot{U}^\top)\mathcal{P}_\Omega(X-A)P_V^\perp.
\end{aligned}$$

Combining this with (A.1), $U_p = \dot{U}\Sigma$, and $V_p = \dot{V}\Sigma$, we have

$$\begin{aligned}
\mathrm{Hess}F(X)[\xi] &= \mathcal{P}_{T_X\mathcal{M}}\mathrm{D}\mathcal{P}_{T_X\mathcal{M}}\mathcal{P}_\Omega(X-A)[\xi] \\
&= P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp \\
&\quad + P_U^\perp\mathcal{P}_\Omega(X-A)(\dot{V}V^\top + V\dot{V}^\top)P_V + P_U(\dot{U}U^\top + U\dot{U}^\top)\mathcal{P}_\Omega(X-A)P_V^\perp \\
&= P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp \\
&\quad + P_U^\perp\mathcal{P}_\Omega(X-A)\dot{V}V^\top + U\dot{U}^\top\mathcal{P}_\Omega(X-A)P_V^\perp \\
&= P_U\mathcal{P}_\Omega(\xi)P_V + P_U^\perp\mathcal{P}_\Omega(\xi)P_V + P_U\mathcal{P}_\Omega(\xi)P_V^\perp \\
&\quad + P_U^\perp\mathcal{P}_\Omega(X-A)V_p\Sigma^{-1}V^\top + U\Sigma^{-1}U_p^\top\mathcal{P}_\Omega(X-A)P_V^\perp,
\end{aligned}$$

which is the desired formula. □

B. Proof of Lemma 4.7

Proof. Define $G(t) := P_\gamma^{0\leftarrow t}\mathrm{grad}F(\gamma(t)) \in T_{X_k}\mathcal{M}_l$ for $t \in \mathbb{R}$. It follows that $G(0) = \mathrm{grad}F(X_k)$ and $G(t_k) = P_\gamma^{0\leftarrow t_k}\mathrm{grad}F(X_{k+1})$. Then, we have from the fundamental theorem of calculus in vector spaces that

$$P_\gamma^{0\leftarrow t_k}\mathrm{grad}F(X_{k+1}) - \mathrm{grad}F(X_k) = G(t_k) - G(0) = \int_0^{t_k} G'(t)\mathrm{d}t.$$

The proof is completed by finding the derivative of $G(t)$ with respect to t. In fact,

$$\begin{aligned}
G'(t) &= \lim_{\varepsilon\to 0}\frac{G(t+\varepsilon) - G(t)}{\varepsilon} \\
&= \lim_{\varepsilon\to 0}\frac{P_\gamma^{0\leftarrow t+\varepsilon}\mathrm{grad}F(\gamma(t+\varepsilon)) - P_\gamma^{0\leftarrow t}\mathrm{grad}F(\gamma(t))}{\varepsilon} \\
&= \lim_{\varepsilon\to 0}\frac{P_\gamma^{0\leftarrow t}(P_\gamma^{t\leftarrow t+\varepsilon}\mathrm{grad}F(\gamma(t+\varepsilon)) - P_\gamma^{0\leftarrow t}\mathrm{grad}F(\gamma(t)))}{\varepsilon}
\end{aligned}$$

$$
\begin{aligned}
&= P_{\gamma}^{0\leftarrow t} \lim_{\varepsilon\to 0} \frac{P_{\gamma}^{t\leftarrow t+\varepsilon}\mathrm{grad}F(\gamma(t+\varepsilon)) - \mathrm{grad}F(\gamma(t))}{\varepsilon} \\
&= P_{\gamma}^{0\leftarrow t} \frac{\mathrm{d}}{\mathrm{d}\varepsilon}\bigg|_{\varepsilon=0} P_{\gamma}^{t\leftarrow t+\varepsilon}\mathrm{grad}F(\gamma(t+\varepsilon)) \\
&= P_{\gamma}^{0\leftarrow t}\nabla_{\gamma'(t)}\mathrm{grad}F(\gamma(t)) \\
&= P_{\gamma}^{0\leftarrow t}\mathrm{Hess}F(\gamma(t))[\gamma'(t)],
\end{aligned}
$$

where the 6th equality uses Theorem 4.34 in [20] and the final equality uses (7). □