Electronic Transactions on Numerical Analysis.
Volume 63, pp. 231–246, 2025.
© 2025, the Author(s), licensed via CC BY 4.0.
ISSN 1068–9613.
DOI: 10.1553/etna_vol63s231

ETNA
Kent State University **and**
Johann Radon Institute (RICAM)

A RIEMANNIAN CONJUGATE GRADIENT METHOD FOR SOLVING THE TENSOR FIXED-RANK LEAST-SQUARES PROBLEM[*]

CHUN-MEI LI[†], YU-YING GU[†], XUE-FENG DUAN[†], AND HUI-YAN PENG[†]

Abstract. In this paper, we consider the tensor fixed-rank least-squares problem arising in image restoration. The Riemannian conjugate gradient method with an exact line search technique is designed to solve this problem. A convergence analysis of this method is given. Numerical experiments with synthetic data and real images demonstrate the feasibility and effectiveness of the new method.

Key words. tensor least-squares problem, fixed-rank constraint, Riemannian conjugate gradient method, exact line search, convergence analysis

AMS subject classifications. 15A69, 58C05, 65F10

1. Introduction.

Throughout this paper, vectors are written in italic lower case letters such as u and v. Matrices are represented by uppercase letters, for example, A and B. Tensors are denoted by calligraphic letters such as $\mathcal{A}$ and $\mathcal{B}$. We define the Frobenius norm of an Nth-order tensor $\mathcal{A} = (\mathcal{A}_{i_1 i_2 \cdots i_N}) \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ by

$$\|\mathcal{A}\| = \left(\sum_{I_1 I_2 \ldots I_N} |\mathcal{A}_{i_1 i_2 \ldots i_N}|^2 \right)^{\frac{1}{2}},$$

which is induced by the inner product

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{I_1 I_2 \cdots I_N} \mathcal{X}_{i_1 i_2 \cdots i_N} \mathcal{Y}_{i_1 i_2 \cdots i_N}.$$

Let $\mathcal{A}$ be the tensor $\mathcal{A} = (\mathcal{A}_{i_1 \ldots i_N j_1 \ldots j_M}) \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ and $\mathcal{B}$ be the tensor $\mathcal{B} = (\mathcal{B}_{j_1 \ldots j_M i_1 \ldots i_N}) \in \mathbb{R}^{J_1 \times \cdots \times J_M \times I_1 \times \cdots \times I_N}$. If $\mathcal{B}_{j_1 \cdots j_M i_1 \cdots i_N} = \mathcal{A}_{i_1 \cdots i_N j_1 \cdots j_M}$, then $\mathcal{B}$ is called the transpose of $\mathcal{A}$ and is denoted by $\mathcal{A}^T$. The Einstein product of the tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times K_1 \times \cdots \times K_N}$ and $\mathcal{B} \in \mathbb{R}^{K_1 \times \cdots \times K_N \times J_1 \times \cdots \times J_M}$ is defined via the operation $*_N$ as

$$(\mathcal{A} *_N \mathcal{B})_{i_1 \cdots i_N j_1 \cdots j_M} = \sum_{k_1 \cdots k_N} \mathcal{A}_{i_1 \cdots i_N k_1 \cdots k_N} \mathcal{B}_{k_1 \cdots k_N j_1 \cdots j_M},$$

where $(\mathcal{A} *_N \mathcal{B}) \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$. Let $\mathcal{B} \in \mathbb{R}^{K_1 \times \cdots \times K_N}$. Then

$$(\mathcal{A} *_N \mathcal{B})_{i_1 \cdots i_N} = \sum_{k_1 \cdots k_N} \mathcal{A}_{i_1 \cdots i_N k_1 \cdots k_N} \mathcal{B}_{k_1 \cdots k_N},$$

[*]Received June 23, 2024. Accepted March 30, 2025. Published online on May 22, 2025. Recommended by Hassane Sadok. Work supported by the Natural Science Foundation of Guangxi Province (Grant No. 2023GXNSFAA026067; 2024GXNSFAA010521) and the National Natural Science Foundation of China (No. 12361079; 12201149; 12261026).

[†]Corresponding author: X.-F. Duan. College of Mathematics and Computational Science, Center for Applied Mathematics of Guangxi (GUET), Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation, Guilin University of Electronic Technology, Guilin 541004, Guangxi, P.R. China (`lichunmei@guet.edu.cn`, `2654409895@qq.com`, `guidian520@126.com`, `2564232142@qq.com`).

where $(\mathcal{A} *_N \mathcal{B}) \in \mathbb{R}^{I_1\times\cdots\times I_N}$. For a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_n}$, the multilinear rank is defined as

$$\operatorname{rank}(\mathcal{X}) = (\operatorname{rank}(\mathcal{X}_1), \dots, \operatorname{rank}(\mathcal{X}_n)),$$

where the $\mathcal{X}_i, i = 1, 2, \dots, n$, are obtained by unfolding the tensor along each mode. For the Tucker decomposition [22] of the tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_N}$ with $\operatorname{rank}(\mathcal{X}) = r = (r_1, \dots, r_n)$, a representation can be given as

$$\mathcal{X} = G \times_1 U_1 \times_2 U_2 \times_3 \cdots \times_N U_N = G \times_{i=1}^{N} U_i,$$

with the core tensor $G \in \mathbb{R}^{r_1\times\cdots\times r_N}$ and the matrices $U_i \in \mathbb{R}^{n_i\times r_i}, i = 1, 2, \dots, N$. The symbol $\times_n$ represents the tensor-matrix product along the nth mode.

In this paper, we consider the following tensor fixed-rank least-squares problem:

PROBLEM 1. *Given tensors $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N}$, $\mathcal{B} \in \mathbb{R}^{I_1\times\cdots\times I_N\times I_1\times\cdots\times I_N}$. Consider the tensor fixed-rank least-squares problem*

$$\begin{aligned}&\min \ \frac{1}{2}\left\|\mathcal{A} *_N \mathcal{X} - \mathcal{B}\right\|_F^2,\\ &\textit{s.t. } \mathcal{X} \in \mathcal{M}_r := \left\{\mathcal{X} \in \mathbb{R}^{J_1\times\cdots\times J_N\times I_1\times\cdots\times I_N} \,\middle|\, \operatorname{rank}(\mathcal{X}) = r\right\}.\end{aligned}$$

When $N = 3$, Problem 1 frequently occurs in image restoration as follows; see Section 5 for more details. The 3D model of color image restoration is formulated by using the tensor expression [8, 12, 18]

$$\mathcal{M} = \mathcal{T} *_3 \mathcal{Q} + \mathcal{N},$$

where $\mathcal{M}$, $\mathcal{T}$, and $\mathcal{N}$ represent the observation image, the Toeplitz tensor (the blurring operator), and the noise, respectively. Moreover, $\mathcal{Q}$ is the restored image to be determined (denoted as $\mathcal{Q}$ to distinguish it from the zero tensor). We note that the primary information of a color image is represented by the top singular values, while the remaining singular values contain noise. In other words, the color image has good low-rank characteristics. Therefore, we add a low-rank constraint to this model and rewrite it as

$$\begin{aligned}&\min \ \frac{1}{2}\left\|\mathcal{T} *_3 \mathcal{Q} + (\mathcal{N} - \mathcal{M})\right\|_F^2,\\ &\text{s.t. } \operatorname{rank}(\mathcal{Q}) = r.\end{aligned}$$

Tensor equations and their least-squares problems have many applications in other areas such as chemometrics [6] and mechanical systems [15]. When $I_2 = \cdots = I_N = 1$ and $J_2 = \cdots = J_N = 1$, Problem 1 reduces to the matrix fixed-rank least-squares problem. Luo, Huang, Li, and Zhang [19] proposed the recursive importance sketching algorithm for solving this problem.

In recent years, there has been a growing interest in developing theories and numerical methods for tensor equations and their least-squares problems. Brazell et al. [4] derived a higher-order biconjugate gradient method for solving tensor equations. Wang and Xu [23] proposed a class of iterative algorithms for solving the tensor equation and the associated least-squares problem. These algorithms utilize tensor computations without involving matricizations. Moreover, in the absence of round-off errors, the solution can be obtained within a finite number of iteration steps. Tang et al. [21] proposed the regular sketch-and-project method for solving tensor equations.

Recently, many scholars derived some results for solving tensor least-squares problems by using the generalized inverse of tensors [2, 7, 14]. Huang and Ma [13] solved the tensor least-squares problem using the conjugate gradient method. Subsequently, Xie et al. [25] generalized the classical iterative method to the preprocess conjugate gradient method for solving Toeplitz tensor equations. Xiao and Zhang [24] employed the proximal-gradient homotopy continuation strategy to solve the sparse least-squares problem. This method ensures that all iterations along the homotopy solution path are sparse. Bentbib et al. [3] proposed the LSQR method for solving the tensor least-squares problem, which uses CP or HOSVD decompositions to obtain a reduced tensor least-squares minimization problem. Soto-Quiros [20] designed a numerical method based on a tensor pseudo-inverse to estimate a solution of the least-squares problem. Thereafter, Paige's algorithm [10] was proposed to solve the problem through an invertible linear transform.

However, as far as we know, there are very few research results on the tensor fixed-rank least-squares Problem 1. Previous numerical algorithms cannot be directly extended to approach Problem 1. The main difficulty in solving Problem 1 is how to characterize the feasible set. In this paper, we overcome this difficulty by using Riemannian manifolds. We first extend the conjugate gradient method from Euclidean space to a Riemannian manifold and then design the Riemannian conjugate gradient method with an exact line search to solve Problem 1. A convergence analysis is also given. Numerical examples are performed to illustrate the feasibility and effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows. In Section 2, we introduce the differential geometry on the Riemannian manifold. In Section 3, we design the Riemannian conjugate gradient algorithm to solve Problem 1. A convergence analysis is given in Section 4. In Section 5, some numerical examples are provided to demonstrate the feasibility and effectiveness of the new algorithm. Section 6 contains concluding remarks.

2. Differential geometry for the low-rank tensor manifold.

In this section, we introduce some differential geometry concepts for the Riemannian manifold.

Kressner, Steinlechner, and Vandereycken showed that the fixed multilinear rank tensor set $\mathcal{M}_r = \{\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_N} : \operatorname{rank}(\mathcal{X}) = r\}$ forms a smooth embedded submanifold of $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$, and its dimension is given by

$$\dim(\mathcal{M}_r) = \prod_{j=1}^{n_N} r_j + \sum_{i=1}^{n_N} r_i n_i - r_i^2.$$

It can be seen that the dimension of Euclidean space $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$ is much larger than $\dim(\mathcal{M}_r)$.

We denote by P_r^{HO} the truncation of a tensor $\mathcal{X} = G \times_{i=1}^N U_i$ to the multilinear rank r using the higher-order singular value decomposition (HOSVD) . The HOSVD procedure can be described as the successive application of the best rank-r_i approximations $P_{r_i}^i$ in each mode $i = 1, \dots, N$:

$$P_r^{HO} : \mathbb{R}^{n_1\times\cdots\times n_d} \to \mathcal{M}_r, \mathcal{X} \mapsto P_{r_N}^{HO} \circ \cdots \circ P_{r_1}^1 \mathcal{X}.$$

Each individual projection can be computed via a truncated SVD as follows. Let $U_\mathcal{X}$ contain the r_i dominant left singular vectors of the ith matricization $\mathcal{X}_{(i)}$ of a given tensor $\mathcal{X}$. Then, the tensor resulting from the projection $\tilde{\mathcal{X}} = P_{r_i}^i \mathcal{X}$ can be expressed in terms of its matricization as $\tilde{\mathcal{X}}_{(i)} = U_\mathcal{X} U_\mathcal{X}^T \mathcal{X}_{(i)}$. In contrast to the matrix case, the HOSVD does not yield the best rank-r approximation in general. Instead, the following quasi-best approximation property [9] holds:

$$\|\mathcal{X} - P_r^{HO}\mathcal{X}\| \le \sqrt{N}\|\mathcal{X} - P_{\mathcal{M}_r}\mathcal{X}\|,$$

where $P_{\mathcal{M}_r}\mathcal{X} \in \mathcal{M}_r$ is the best approximation of $\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_N}$ with respect to the norm $\|\cdot\|$. Let the HOSVD of the tensor $\mathcal{X} \in \mathcal{M}_r$ be $\mathcal{X} = G \times_1 U_1 \times_2 U_2 \times \cdots \times_N U_N$. Then the tangent space of $\mathcal{M}_r$ at $\mathcal{X}$ can be written as

$$T_{\mathcal{X}}\mathcal{M}_r = \left\{ G \times_{i=1}^{n_N} U_i + \sum_{i=1}^{n_N} G \times_i V_i \times_{j\neq i} U_j \;\middle|\; V_i^T U_i = 0 \right\},$$

where $G \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$ and $V_i \in \mathbb{R}^{n_i\times r_i}$ are the free parameters and $\times_{j\neq i}$ means multiplying along every mode except i. Furthermore, the orthogonal projection of a tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$ onto $T_{\mathcal{X}}\mathcal{M}_r$ is given by

$$P_{T_{\mathcal{X}}\mathcal{M}_r} : \mathbb{R}^{n_1\times n_2\times\cdots\times n_N} \to T_{\mathcal{X}}\mathcal{M}_r,$$

$$\mathcal{X} \to (\mathcal{X} \times_{j=1}^{n_N} U_j^T) \times_{i=1}^{n_N} U_i + \sum_{i=1}^{n_N} G \times_i (P_{U_i}^{\perp}[\mathcal{X} \times_{j\neq i} U_j^T]_{(i)} G_{(i)}^{\dagger}) \times_{k\neq i} U_k, \tag{2.1}$$

where $G_{(i)}^{\dagger}$ is the pseudo-inverse of G_i and $P_{U_i}^{\perp} = I - U_i U_i^T$ represents the orthogonal projection.

Next, we introduce the definition of the Riemannian gradient of an objective function on a Riemannian manifold. The Riemannian gradient $\operatorname{grad} f(\mathcal{X})$ at the point $\mathcal{X} \in \mathcal{M}_r$ is defined as the unique tangent vector on $T_{\mathcal{X}}\mathcal{M}_r$ such that

$$\langle \operatorname{grad} f(\mathcal{X}), \xi\rangle = Df(\mathcal{X})[\xi] \qquad \text{for all } \xi \in T_{\mathcal{X}}\mathcal{M}_r.$$

LEMMA 2.1 ([17]). *Let $f : \mathcal{M}_r \to \mathbb{R}$ be a cost function, where $\mathcal{M}_r$ is a certain manifold and f maps tensors in $\mathcal{M}_r$ to real numbers. Let $\nabla f(\mathcal{X})$ denote the Euclidean gradient of f at the point $\mathcal{X} \in \mathcal{M}_r$. Then the Riemannian gradient* $\operatorname{grad} f(\mathcal{X})$ *of f is given by* $\operatorname{grad} f(\mathcal{X}) = P_{T_{\mathcal{X}}\mathcal{M}_r}(\nabla f(\mathcal{X}))$*, where $P_{T_{\mathcal{X}}\mathcal{M}_r}$ is the projection onto the tangent space at $\mathcal{X}$.*

The Euclidean gradient of the objective function $f(\mathcal{X}) = \frac{1}{2}\|\mathcal{A} *_N \mathcal{X} - \mathcal{B}\|_F^2$ is given by

$$\nabla f(\mathcal{X}) = (\mathcal{A}^T *_N \mathcal{A}) *_N \mathcal{X} - \mathcal{A}^T *_N \mathcal{B},$$

so according to Lemma 2.1 we get the Riemannian gradient

$$\operatorname{grad} f(\mathcal{X}_k) = P_{T_{\mathcal{X}_k}\mathcal{M}_r}(\nabla f(\mathcal{X}_k)) = C \times_{i=1}^{n_N} U_i + \sum_{i=1}^{n_N} G \times_i W_i \times_{k\neq i} U_k,$$

where $C = \nabla f(\mathcal{X}_k) \times_{j=1}^{n_N} U_j^T$ and $W_i = P_{U_i}^{\perp}[\nabla f(\mathcal{X}_k) \times_{j\neq i} U_j^T]_{(i)} G_{(i)}^{\dagger}$.

The retraction map is a mapping from the tangent space at $\mathcal{X} \in \mathcal{M}_r$, denoted as $T_{\mathcal{X}}\mathcal{M}_r$, to the manifold $\mathcal{M}_r$. A popular theoretical choice is the so-called exponential map. However, computing this map is usually computationally expensive. For the purposes of optimization algorithms, it is often sufficient to approximate this exponential map to first order. More specifically, a retraction satisfies the following properties:

DEFINITION 2.2 ([1]). *Let $\mathcal{M}_r$ be a smooth submanifold of $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$. Denote the zero element of the tangent space $T_{\mathcal{X}}\mathcal{M}_r$ as $0_{\mathcal{X}}$. A mapping R from the tangent bundle $T\mathcal{M}_r$ to $\mathcal{M}_r$ is said to be a retraction on $\mathcal{M}_r$ around $\mathcal{X} \in \mathcal{M}_r$ if there exists a neighborhood U of $(\mathcal{X}, 0_{\mathcal{X}})$ in $T\mathcal{M}_r$ such that the following properties hold:*

(a) We have $U \subseteq \operatorname{dom}(R)$ and the restriction $R : U \to \mathcal{M}_r$ is smooth.

(b) $R(\mathcal{Y}, 0_{\mathcal{Y}}) = \mathcal{Y}$ for all $(\mathcal{Y}, 0_{\mathcal{Y}}) \in U$.

(c) With the canonical identification $T_{0_{\mathcal{X}}} T\mathcal{M}_r \simeq T\mathcal{M}_r$, R satisfies the local rigidity condition

$$DR(\mathcal{X}, \cdot)(0_{\mathcal{X}}) = id_{T_{\mathcal{X}}\mathcal{M}_r} \qquad \textit{for all } (\mathcal{X}, 0_{\mathcal{X}}) \in U,$$

where $\mathrm{id}_{T_\mathcal{X}\mathcal{M}_r}$ *denotes the identity mapping on* $T_\mathcal{X}\mathcal{M}_r$.

In fact, if the Riemannian manifold $\mathcal{M}_r$ is an embedded submanifold of $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$, then the orthogonal projection from $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$ onto the tangent space of $\mathcal{M}_r$ is

$$P_{\mathcal{M}_r}(\mathcal{X}+\xi)=\underset{y\in\mathcal{M}_r}{\arg\min}\,\|\mathcal{X}+\xi-y\|,$$

which induces the projective retraction

$$R_\mathcal{X}:\mathcal{U}_\mathcal{X}\to\mathcal{M}_r,(\mathcal{X},\eta)\to P_{\mathcal{M}_r}(\mathcal{X}+\eta).$$

Here, $\mathcal{U}_\mathcal{X}\subseteq T_\mathcal{X}\mathcal{M}_r$ represents a suitable neighborhood around zero, and $P_{\mathcal{M}_r}$ denotes the orthogonal projection onto $\mathcal{M}_r$. However, we find that the HOSVD procedure cannot yield a projection retraction that satisfies Definition 2.2 (see [1] for more details). Fortunately, this problem can be solved by using the truncated HOSVD.

LEMMA 2.3 ([16]). *Let* $\mathcal{X}$ *be an element of* $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$ *with multirank* $\operatorname{rank}(\mathcal{X})=r=(r_1,r_2,\cdots,r_d)$. *Then a retraction on* $\mathcal{M}_r$ *around* $\mathcal{X}$ *is given by*

$$\begin{aligned}&R_\mathcal{X}:T_\mathcal{X}\mathcal{M}_r\to\mathcal{M}_r,\\&\quad(\mathcal{X},\eta)\to P_{T_\mathcal{X}\mathcal{M}_r}(\mathcal{X}+\eta):=\bar{G}\times_1 V_1\times_2 V_2\times\cdots\times_d V_d,\end{aligned}$$

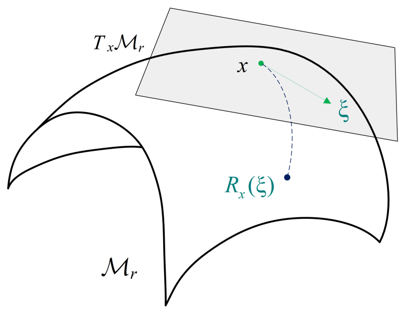
where $\bar{G}\in\mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$, $V_i\in\mathbb{R}^{2n_i\times n_i}$, *for* $1\le i\le d$.

A graphical depiction of this concept is shown in Figure 2.1(a).

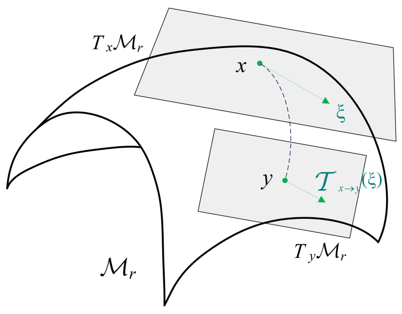
Vector transport was first introduced as a means of transporting vectors from one tangent space to another; see [1] for more details. Since $\mathcal{M}_r$ is an embedded submanifold of $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$, orthogonally projecting the translated tangent vector in $\mathbb{R}^{n_1\times n_2\times\cdots\times n_N}$ onto the new tangent space constitutes a vector transport. Then we can define

$$\mathcal{T}_{\mathcal{X}\to\mathcal{Y}}:\mathcal{T}_\mathcal{X}\mathcal{M}_r\to\mathcal{T}_\mathcal{Y}\mathcal{M}_r,\quad \xi\to P_{\mathcal{T}_\mathcal{Y}\mathcal{M}_r}(\xi),$$

where $P_{\mathcal{T}_\mathcal{Y}\mathcal{M}_r}$ is defined as in (2.1). A graphical depiction of vector transport is shown in Figure 2.1(b).



(a) Retraction



(b) Vector transport

FIG. 2.1. *Retraction and vector transport on a Riemannian manifold.*

3. A Riemannian conjugate gradient method for solving Problem 1.

In this section, we first introduce the conjugate gradient method in the Euclidean space and then extend this method from the Euclidean space to the Riemannian manifold. Finally, we employ it to solve Problem 1.

The conjugate gradient method is an excellent method for solving the large-scale optimization problem

$$\min_{x\in\mathbb{R}^n} f(x),$$

and its iterative formula is given as follows:

$$x_{k+1} = x_k + \alpha_k d_k, \qquad k = 0, 1, 2, \ldots, \tag{3.1}$$

where d_k represents the search direction at the point x_k, which is determined by

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_k d_{k-1}, & k \geq 1. \end{cases} \tag{3.2}$$

Here, the coefficient β_k is defined as [11]

$$\beta_k = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}. \tag{3.3}$$

The step length α_k is determined by an exact line search, that is, we find α_k such that

$$\alpha_k = \operatorname*{argmin}_{\alpha} f(x_k + \alpha d_k). \tag{3.4}$$

Next, we construct the Riemannian conjugate gradient (RCG) method for solving the optimization problem

$$\min_{\mathcal{X}\in\mathcal{M}_r} f(\mathcal{X}),$$

where $\mathcal{M}_r$ is a Riemannian manifold. The expressions for the iterative formula (3.1), the search direction (3.2), the iterative coefficient (3.3), and the step length (3.4) must be adjusted, which can be done as follows. We can move x_k along the direction $\alpha_k d_k$ to update (3.1) in $\mathbb{R}^n$. However, it cannot be guaranteed that we can do the same on a Riemannian manifold. Fortunately, we can use a retraction to overcome this difficulty, and the iterative formula can be written as

$$\mathcal{X}_{k+1} = R(\mathcal{X}_k,\ \alpha_k d_k), \qquad k = 0, 1, \ldots$$

We consider the search direction d_{k+1}. Unlike in the Euclidean space, on the Riemannian manifold, the Riemannian gradient $\operatorname{grad} f(\mathcal{X}_k) \in T_{\mathcal{X}_k}\mathcal{M}_r$ and the search direction $\eta_{k-1} \in T_{\mathcal{X}_{k-1}}\mathcal{M}_r$ belong to different tangent spaces. Consequently, they cannot be directly combined. Therefore, (3.2) and (3.3) also cannot be generalized to the Riemannian manifold. This difficulty can be solved by a vector transport that transports d_{k-1} and g_k to the tangent space of $\mathcal{X}_k$. Using vector transport, we obtain the new search direction

$$\eta_k = \begin{cases} -\operatorname{grad} f(\mathcal{X}_k), & k = 0, \\ -\operatorname{grad} f(\mathcal{X}_k) + \beta_k \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1}), & k \geq 1. \end{cases} \tag{3.5}$$

The coefficient β_k is adjusted as

$$\beta_k = \frac{\langle \operatorname{grad} f(\mathcal{X}_k), \operatorname{grad} f(\mathcal{X}_k)\rangle}{\langle \operatorname{grad} f(\mathcal{X}_{k-1}), \operatorname{grad} f(\mathcal{X}_{k-1})\rangle}. \tag{3.6}$$

On the other hand, we extend the exact line search from the Euclidean space to the manifold $\mathcal{M}_r$, that is, we find α_k satisfying

$$\alpha_k = \min_{\alpha} f(R(\mathcal{X},\ \alpha\eta_k)).$$

Noting that

$$0 = \frac{\mathrm{d}}{\mathrm{d}\alpha_k}\left[R(\mathcal{X}_k,\ \alpha_k\eta_k)\right] = Df(\mathcal{X}_{k+1})\left[\frac{\mathrm{d}}{\mathrm{d}\alpha_k}R(\mathcal{X}_k,\ \alpha_k\eta_k)\right],$$

we have

$$\alpha_k = \frac{\langle \mathcal{A} *_N \eta_k, \mathcal{B} - \mathcal{A} *_N \mathcal{X}_k\rangle}{\langle \mathcal{A} *_N \eta_k, \mathcal{A} *_N \eta_k\rangle}. \tag{3.7}$$

Therefore, the Riemannian conjugate gradient method (RCG) for solving Problem 1 is summarized in Algorithm 1.

Algorithm 1 RCG method for solving Problem 1.

```
Require: Initial values $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N}$, $\mathcal{B} \in \mathbb{R}^{I_1\times\cdots\times I_N\times K_1\times\cdots\times K_N}$,
  $\mathcal{X}_0 \in \mathbb{R}^{J_1\times\cdots\times J_N\times K_1\times\cdots\times K_N} \in \mathcal{M}_r$.
  The fixed-rank constraint is $r = (r_1, r_2, \ldots, r_d)$. The tolerance is $\varepsilon$.
  1. $\eta_0 = -\operatorname{grad} f(\mathcal{X}_0)$
  2. $\alpha_0 = \operatorname{argmin}_\alpha f(\mathcal{X}_0 + \alpha\eta_0)$
  3. $\mathcal{X}_1 = R(\mathcal{X}_0,\ \alpha_0\eta_0)$
  for $k = 1, 2, \ldots$ do
      4. Compute the Riemannian gradient
             $\operatorname{grad} f(\mathcal{X}_k) = P_{T_{\mathcal{X}_k}\mathcal{M}_r}(\bigtriangledown f(\mathcal{X}_k))$.
      5. Compute the Riemannian conjugate coefficient
             $\beta_k = \frac{\|\operatorname{grad} f(\mathcal{X}_k)\|^2}{\|\operatorname{grad} f(\mathcal{X}_{k-1})\|^2}$.
      6. Compute the Riemannian search direction
             $\eta_k = -\operatorname{grad} f(\mathcal{X}_k) + \beta_k \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})$.
      7. Compute the step size $\alpha_k$ satisfying the exact line search condition (3.7).
      8. Obtain the next iteration by retraction
             $\mathcal{X}_{k+1} = R(\mathcal{X}_k,\ \alpha_k\eta_k)$.
      if $\|\operatorname{grad} f(\mathcal{X}_k)\| < \varepsilon$ then
          break
      end if
  end for
```

4. Convergence analysis for Algorithm 1.

In this section, we provide a convergence analysis for Algorithm 1 and begin with an assumption.

ASSUMPTION 1.

(1) The function $f : \mathcal{M}_r \to \mathbb{R}$ is differentiable and bounded below in the level set $\Omega := \{\mathcal{X}_* \in \mathcal{M}_r \mid f(\mathcal{X}_*) < f(\mathcal{X}_0)\}$.

(2) The gradient $\operatorname{grad} f(\mathcal{X}) : T_{\mathcal{X}}\mathcal{M}_r \to T_{\mathcal{X}}\mathcal{M}_r$ is Lipschitz continuous, that is, there exists a constant $L > 0$ such that

$$\| \operatorname{grad} f(\xi_{\mathcal{X}}) - \operatorname{grad} f(0_{\mathcal{X}})\| \leq L\|\xi_{\mathcal{X}}\|$$

for all $\mathcal{X} \in \Omega$, and $\xi_{\mathcal{X}} \in \mathcal{T}_{\mathcal{X}}\mathcal{M}_r$ with $\|\xi_{\mathcal{X}}\| \leq s$.

(3) The vector transport $\mathcal{T}$ satisfies the inequality

$$\|\mathcal{T}_{\eta_x}\xi_{\mathcal{X}}\| \leq \|\xi_{\mathcal{X}}\|, \qquad \forall \xi_{\mathcal{X}}, \eta_{\mathcal{X}} \in \mathcal{T}_{\mathcal{X}}\mathcal{M}_r, \mathcal{X} \in \mathcal{M}_r. \tag{4.1}$$

LEMMA 4.1. *The search direction η_k of Algorithm 1 satisfies the sufficient descent condition*

$$\langle \operatorname{grad} f(\mathcal{X}_k), \eta_k\rangle \leq -C\| \operatorname{grad} f(\mathcal{X}_k)\|^2 \quad \textit{for } k \geq 0, \textit{ where } C > 0. \tag{4.2}$$

Proof. If $k = 0$, then

$$\langle \eta_k, \operatorname{grad} f(\mathcal{X}_k)\rangle = -\langle \operatorname{grad} f(\mathcal{X}_0), \operatorname{grad} f(\mathcal{X}_0)\rangle = -\| \operatorname{grad} f(\mathcal{X}_0)\|^2.$$

When $k \geq 1$, we have

$$\begin{aligned}
&\langle \operatorname{grad} f(\mathcal{X}_k), \eta_k\rangle \\
&= \langle - \operatorname{grad} f(\mathcal{X}_k), \operatorname{grad} f(\mathcal{X}_k) + \beta^{FR}\mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\rangle \\
&= -\| \operatorname{grad} f(\mathcal{X}_k)\|^2 - \beta^{FR}\langle \operatorname{grad} f(\mathcal{X}_k), \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\rangle \\
&= -\| \operatorname{grad} f(\mathcal{X}_k)\|^2 - \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^2}{\| \operatorname{grad} f(\mathcal{X}_{k-1})\|^2} \langle \operatorname{grad} f(\mathcal{X}_k), \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\rangle.
\end{aligned}$$

Assuming that $\mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})$ is collinear with $\frac{\mathrm{d}}{\mathrm{d}t}R(\mathcal{X}_k,\ t\eta_k)|_{t=\alpha_k}$ leads to

$$\langle \operatorname{grad} f(\mathcal{X}_k), \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\rangle = 0. \tag{4.3}$$

Thus, we have

$$\langle \operatorname{grad} f(\mathcal{X}_k), \eta_k\rangle = -\| \operatorname{grad} f(\mathcal{X}_k)\|^2.$$

Therefore, the search direction η_k is a descent direction. □

LEMMA 4.2. *Suppose that Assumption 1 is satisfied. Then the Zoutendijk condition*

$$\sum_{k=0}^{\infty} \left(\frac{\langle \operatorname{grad} f(\mathcal{X}_k), \eta_k\rangle}{\|\eta_k\|}\right)^2 = \sum_{k=0}^{\infty} \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^4}{\|\eta_k\|^2} < \infty \tag{4.4}$$

holds.

Proof. By the step length (3.7), (4.2), and Assumption 1, we can deduce that the sequence $\{f(\mathcal{X}_k)\}$ is decreasing and has a lower bound, and therefore it is convergent. Thus,

$$\sum_{k=0}^{\infty} f(R(\mathcal{X}_k,\ \alpha_k\eta_k)) - f(\mathcal{X}_k) < \infty. \tag{4.5}$$

From (4.5) and (4.2), it follows that

$$\sum_{k=0}^{\infty} \alpha_k^2 \|\eta_k\|^2 < \infty$$

and

$$\sum_{k=0}^{\infty} \alpha_k \| \operatorname{grad} f(\mathcal{X}_k)\|^2 < -C \sum_{k=0}^{\infty} \alpha_k \langle \operatorname{grad} f(\mathcal{X}_k), \eta_k\rangle < \infty.$$

Consequently,

$$\lim_{k\to\infty} \alpha_k \|\eta_k\| = 0, \qquad \lim_{k\to\infty} \alpha_k \| \operatorname{grad} f(\mathcal{X}_k)\|^2 = 0.$$

Therefore,

$$\sum_{k=0}^{\infty} \left(\frac{\langle \operatorname{grad} f(\mathcal{X}_k), \eta_k\rangle}{\|\eta_k\|} \right)^2 = \sum_{k=0}^{\infty} \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^4}{\|\eta_k\|^2} < \infty. \qquad \square$$

Now, we present the convergence theorem for Algorithm 1.

THEOREM 4.3. *Under Assumption 1, we obtain that*

$$\liminf_{k\to\infty} \| \operatorname{grad} f(\mathcal{X}_k)\| = 0.$$

Proof. We assume the existence of a constant $\varepsilon > 0$ such that

$$\| \operatorname{grad} f(\mathcal{X}_k)\| \geq \varepsilon, \quad \forall k \geq 0.$$

From (3.5) and (3.6), for $k \geq 1$, we have

$$\begin{aligned}
\|\eta_k\|^2 &= \| - \operatorname{grad} f(\mathcal{X}_k) + \beta_k^{FR} \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\|^2 \\
&= \| \operatorname{grad} f(\mathcal{X}_k)\|^2 - 2\beta_k^{FR} \langle \operatorname{grad} f(\mathcal{X}_k), \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\rangle \\
&\qquad + (\beta_k^{FR})^2 \|\mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\|^2 \\
&= \| \operatorname{grad} f(\mathcal{X}_k)\|^2 - 2\frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^2}{\| \operatorname{grad} f(\mathcal{X}_{k-1})\|^2} \langle \operatorname{grad} f(\mathcal{X}_k), \mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\rangle \\
&\qquad + \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^4}{\| \operatorname{grad} f(\mathcal{X}_{k-1})\|^4} \|\mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\|^2.
\end{aligned}$$

By (4.3) we have

$$\|\eta_k\|^2 = \| \operatorname{grad} f(\mathcal{X}_k)\|^2 + \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^4}{\| \operatorname{grad} f(\mathcal{X}_{k-1})\|^4} \|\mathcal{T}_{\mathcal{X}_{k-1}\to\mathcal{X}_k}(\eta_{k-1})\|^2. \tag{4.6}$$

Combining (4.1) and (4.6), we obtain that

$$\|\eta_k\|^2 \leq \| \operatorname{grad} f(\mathcal{X}_k)\|^2 + \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^4}{\| \operatorname{grad} f(\mathcal{X}_{k-1})\|^4} \|\eta_{k-1}\|^2. \tag{4.7}$$

Dividing both sides of (4.7) by $\| \operatorname{grad} f(\mathcal{X}_k)\|^4$, we get

$$\begin{aligned}\frac{\|\eta_k\|^2}{\| \operatorname{grad} f(\mathcal{X}_k)\|^4} &\leq \frac{1}{\| \operatorname{grad} f(\mathcal{X}_k)\|^2} + \cdots + \frac{1}{\| \operatorname{grad} f(\mathcal{X}_1)\|^2} + \frac{\|\eta_0\|^2}{\| \operatorname{grad} f(\mathcal{X}_0)\|^4} \\ &= \sum_{i=1}^{k} \frac{1}{\| \operatorname{grad} f(\mathcal{X}_i)\|^2} + \frac{1}{\| \operatorname{grad} f(\mathcal{X}_0)\|^2} \\ &\leq \frac{k}{\varepsilon^2} + 1,\end{aligned}$$

which implies that

$$\sum_{k=0}^{\infty} \frac{\| \operatorname{grad} f(\mathcal{X}_k)\|^4}{\|\eta_k\|^2} \geq \sum_{i=0}^{k} \frac{\varepsilon^2}{\varepsilon^2 + k} = \infty.$$

This is in contradiction with the Riemannian Zoutendijk condition (4.4). Thus we have

$$\liminf_{k\to\infty} \| \operatorname{grad} f(\mathcal{X}_k)\| = 0. \qquad \square$$

5. Numerical experiments. In this section, we first employ a simple example to demonstrate that Algorithm 1 is feasible for solving Problem 1. Subsequently, we compare it with the Riemannian steepest descent (RSD) algorithm (Algorithm 2) and the nonmonotone spectral projected gradient method (denoted by "NSPG") [5], both of which can also be utilized to solve Problem 1. All the tests were implemented in Matlab (R2018b) on a personal computer with a 1.80 GHz central processing unit (Intel(R) Core(TM) i5-8265U), 8 GB memory, and the Windows 11 operating system.

Algorithm 2 RSD method for solving Problem 1.

```
Require: Initial values $\mathcal{A} \in \mathbb{R}^{I_1\times\cdots\times I_N\times J_1\times\cdots\times J_N}$, $\mathcal{B} \in \mathbb{R}^{I_1\times\cdots\times I_N\times K_1\times\cdots\times K_N}$,
    $\mathcal{X}_0 \in \mathbb{R}^{J_1\times\cdots\times J_N\times K_1\times\cdots\times K_N} \in \mathcal{M}_r$.
  The fixed-rank constraint is $r = (r_1, r_2, \ldots, r_d)$. The tolerance is $\varepsilon$.
  for $k = 0, 1, \ldots$ do
      1. $\operatorname{grad} f(\mathcal{X}_k) = P_{\mathcal{T}_{\mathcal{X}_k}\mathcal{M}_r}(\bigtriangledown f(\mathcal{X}_k))$
      2. $\eta_k = -\operatorname{grad} f(\mathcal{X}_k)$
      3. $\alpha_k = \operatorname{argmin}_\alpha f(\mathcal{X}_k + \alpha\eta_k)$
      4. $\mathcal{X}_{k+1} = R(\mathcal{X}_k, \alpha_k\eta_k)$
      if $\| \operatorname{grad} f(\mathcal{X}_k)\| < \varepsilon$ then
          break
      end if
  end for
```

We use either the norm of the Riemannian gradient

$$\| \operatorname{grad} f(\mathcal{X}_k)\| \leq 10^{-4}$$

or the maximum number of iterations

$$k \leq 5000$$

as the stopping criterion.

EXAMPLE 5.1. Consider Problem 1 with

$$\mathcal{A}(:,:,1,1) = \begin{bmatrix} 0.8282 & 0.7158 & 0.9007 \\ 0.9303 & 0.7285 & 0.2425 \\ 0.6372 & 0.2940 & 0.4178 \end{bmatrix}, \quad \mathcal{A}(:,:,2,1) = \begin{bmatrix} 0.3744 & 0.9261 & 0.4707 \\ 0.4870 & 0.6615 & 0.5774 \\ 0.4749 & 0.9177 & 0.2081 \end{bmatrix},$$
$$\mathcal{A}(:,:,3,1) = \begin{bmatrix} 0.3744 & 0.9261 & 0.4707 \\ 0.4870 & 0.6615 & 0.5774 \\ 0.4749 & 0.9177 & 0.2081 \end{bmatrix}, \quad \mathcal{A}(:,:,1,2) = \begin{bmatrix} 0.8703 & 0.9272 & 0.5196 \\ 0.1707 & 0.9840 & 0.5508 \\ 0.1153 & 0.4996 & 0.7451 \end{bmatrix},$$
$$\mathcal{A}(:,:,2,2) = \begin{bmatrix} 0.6603 & 0.9750 & 0.9542 \\ 0.6790 & 0.8510 & 0.2445 \\ 0.3121 & 0.8926 & 0.2323 \end{bmatrix}, \quad \mathcal{A}(:,:,3,2) = \begin{bmatrix} 0.1189 & 0.7551 & 0.0961 \\ 0.2239 & 0.7063 & 0.1043 \\ 0.5493 & 0.0219 & 0.3799 \end{bmatrix},$$
$$\mathcal{A}(:,:,1,3) = \begin{bmatrix} 0.1066 & 0.2142 & 0.8517 \\ 0.2022 & 0.2551 & 0.9627 \\ 0.6694 & 0.8460 & 0.1468 \end{bmatrix}, \quad \mathcal{A}(:,:,2,3) = \begin{bmatrix} 0.9916 & 0.5635 & 0.7265 \\ 0.9298 & 0.3691 & 0.7303 \\ 0.7496 & 0.4069 & 0.5483 \end{bmatrix},$$
$$\mathcal{A}(:,:,3,3) = \begin{bmatrix} 0.9493 & 0.6050 & 0.2580 \\ 0.7107 & 0.8135 & 0.6820 \\ 0.4686 & 0.5102 & 0.9946 \end{bmatrix},$$

and

$$\mathcal{B}(:,:,1,1) = \begin{bmatrix} 2.6269 & 3.1528 & 2.8715 \\ 2.6044 & 3.0048 & 2.2484 \\ 2.1588 & 2.8888 & 1.7384 \end{bmatrix}, \quad \mathcal{B}(:,:,2,1) = \begin{bmatrix} 2.4238 & 2.7627 & 2.7196 \\ 2.5001 & 2.6989 & 2.0084 \\ 1.8746 & 2.8184 & 1.4862 \end{bmatrix},$$
$$\mathcal{B}(:,:,3,1) = \begin{bmatrix} 3.4676 & 3.5632 & 3.3165 \\ 3.0304 & 3.7055 & 2.8426 \\ 2.4777 & 3.2230 & 2.5623 \end{bmatrix}, \quad \mathcal{B}(:,:,1,2) = \begin{bmatrix} 2.8713 & 2.6893 & 2.5327 \\ 2.3565 & 3.0512 & 2.8868 \\ 1.8461 & 3.2353 & 2.2592 \end{bmatrix},$$
$$\mathcal{B}(:,:,2,2) = \begin{bmatrix} 1.7029 & 2.3161 & 1.2142 \\ 1.5768 & 2.2643 & 1.2301 \\ 1.4342 & 1.4129 & 1.5922 \end{bmatrix}, \quad \mathcal{B}(:,:,3,2) = \begin{bmatrix} 2.4897 & 2.8340 & 2.3325 \\ 2.1881 & 2.7225 & 2.4539 \\ 1.9714 & 2.7054 & 1.9011 \end{bmatrix},$$
$$\mathcal{B}(:,:,1,3) = \begin{bmatrix} 4.2534 & 4.5300 & 3.9948 \\ 3.8104 & 4.6087 & 3.3532 \\ 3.1201 & 3.8296 & 3.1548 \end{bmatrix}, \quad \mathcal{B}(:,:,2,3) = \begin{bmatrix} 4.3930 & 4.8080 & 4.4708 \\ 4.0471 & 4.7148 & 3.3010 \\ 3.1531 & 4.0900 & 2.8217 \end{bmatrix},$$
$$\mathcal{B}(:,:,3,3) = \begin{bmatrix} 3.6551 & 3.0675 & 3.0901 \\ 2.9177 & 3.2714 & 2.9909 \\ 2.2886 & 2.8759 & 2.6592 \end{bmatrix}.$$

Set $r = (1, 2, 1, 1)$. We use Algorithm 1 to solve it with the initial value

$$\mathcal{X}_0(:,:,1,1) = \begin{bmatrix} 0.5041 & 0.5174 & 0.5206 \\ 0.5953 & 0.6110 & 0.6148 \\ 0.5249 & 0.5388 & 0.5422 \end{bmatrix}, \quad \mathcal{X}_0(:,:,2,1) = \begin{bmatrix} 0.4672 & 0.4795 & 0.4825 \\ 0.5517 & 0.5662 & 0.5698 \\ 0.4865 & 0.4993 & 0.5025 \end{bmatrix},$$
$$\mathcal{X}_0(:,:,3,1) = \begin{bmatrix} 0.4514 & 0.4633 & 0.4662 \\ 0.5330 & 0.5471 & 0.5505 \\ 0.4700 & 0.4824 & 0.4854 \end{bmatrix}, \quad \mathcal{X}_0(:,:,1,2) = \begin{bmatrix} 0.4309 & 0.4423 & 0.4451 \\ 0.5089 & 0.5223 & 0.5256 \\ 0.4487 & 0.4606 & 0.4635 \end{bmatrix},$$

$$\mathcal{X}_0(:,:,2,2) = \begin{bmatrix} 0.3994 & 0.4099 & 0.4125 \\ 0.4716 & 0.4840 & 0.4871 \\ 0.4159 & 0.4269 & 0.4295 \end{bmatrix}, \quad \mathcal{X}_0(:,:,3,2) = \begin{bmatrix} 0.3858 & 0.3960 & 0.3985 \\ 0.4556 & 0.4677 & 0.4706 \\ 0.4018 & 0.4124 & 0.4150 \end{bmatrix},$$

$$\mathcal{X}_0(:,:,1,3) = \begin{bmatrix} 0.6725 & 0.6902 & 0.6945 \\ 0.7941 & 0.8151 & 0.8202 \\ 0.7003 & 0.7188 & 0.7233 \end{bmatrix}, \quad \mathcal{X}_0(:,:,2,3) = \begin{bmatrix} 0.6232 & 0.6397 & 0.6437 \\ 0.7359 & 0.7554 & 0.7601 \\ 0.6490 & 0.6661 & 0.6703 \end{bmatrix},$$

$$\mathcal{X}_0(:,:,3,3) = \begin{bmatrix} 0.6021 & 0.6180 & 0.6219 \\ 0.7110 & 0.7298 & 0.7343 \\ 0.6270 & 0.6436 & 0.6476 \end{bmatrix}.$$

After 58 iterations, we obtain a solution $\mathcal{X}_*$ as

$$\mathcal{X}_*(:,:,1,1) = \begin{bmatrix} 0.5325 & 0.5641 & 0.5658 \\ 0.5974 & 0.6329 & 0.6348 \\ 0.4606 & 0.4879 & 0.4894 \end{bmatrix}, \quad \mathcal{X}_*(:,:,2,1) = \begin{bmatrix} 0.4874 & 0.5163 & 0.5179 \\ 0.5468 & 0.5793 & 0.5810 \\ 0.4216 & 0.4466 & 0.4479 \end{bmatrix},$$

$$\mathcal{X}_*(:,:,3,1) = \begin{bmatrix} 0.4872 & 0.5161 & 0.5177 \\ 0.5466 & 0.5790 & 0.5808 \\ 0.4214 & 0.4464 & 0.4478 \end{bmatrix}, \quad \mathcal{X}_*(:,:,1,2) = \begin{bmatrix} 0.4405 & 0.4666 & 0.4680 \\ 0.4941 & 0.5235 & 0.5251 \\ 0.3810 & 0.4036 & 0.4048 \end{bmatrix},$$

$$\mathcal{X}_*(:,:,2,2) = \begin{bmatrix} 0.4031 & 0.4271 & 0.4284 \\ 0.4523 & 0.4791 & 0.4806 \\ 0.3487 & 0.3694 & 0.3705 \end{bmatrix}, \quad \mathcal{X}_*(:,:,3,2) = \begin{bmatrix} 0.4030 & 0.4269 & 0.4282 \\ 0.4521 & 0.4789 & 0.4804 \\ 0.3485 & 0.3692 & 0.3703 \end{bmatrix},$$

$$\mathcal{X}_*(:,:,1,3) = \begin{bmatrix} 0.7130 & 0.7553 & 0.7576 \\ 0.7999 & 0.8473 & 0.8499 \\ 0.6166 & 0.6532 & 0.6552 \end{bmatrix}, \quad \mathcal{X}_*(:,:,2,3) = \begin{bmatrix} 0.6526 & 0.6913 & 0.6934 \\ 0.7321 & 0.7756 & 0.7779 \\ 0.5644 & 0.5979 & 0.5997 \end{bmatrix},$$

$$\mathcal{X}_*(:,:,3,3) = \begin{bmatrix} 0.6523 & 0.6910 & 0.6931 \\ 0.7318 & 0.7752 & 0.7776 \\ 0.5642 & 0.5977 & 0.5995 \end{bmatrix}$$

and the norm of the Riemannian gradient

$$\| \operatorname{grad} f(\mathcal{X}_*)\| = 9.9648 \times 10^{-5}.$$

The convergence curve of the norm of the Riemannian gradient is presented in Figure 5.1. As demonstrated in Example 5.1, Algorithm 1 is feasible for solving Problem 1.

EXAMPLE 5.2. Consider Problem 1 with

$$\mathcal{A} = \mathrm{tenrand}(I_1, I_2, J_1, J_2), \qquad \mathcal{B} = \mathrm{tenrand}(I_1, I_2, K_1, K_2).$$

(Here tenrand generates a tensor with uniformly distributed values in $[0, 1]$.) With the same initial value and stopping criteria, we employ Algorithm 1 (denoted by "RCG"), Algorithm 2 (denoted by "RSD"), and the nonmonotone spectral projected gradient method (denoted by "NSPG") to solve the problem. The experimental results are presented in Table 5.1, which include the number of iterations (denoted by "IT"), the CPU time (denoted by "CPU (s)"), and the norm of the Riemannian gradient $\| \operatorname{grad} f(\mathcal{X})\|$. When $I_1 = 6$, $I_2 = 5$, $J_1 = 4$, $J_2 = 4$, $K_1 = 3$, $K_2 = 2$, and $r = (3, 2, 3, 2)$, the convergence curves are depicted in Figure 5.2. From Figure 5.2 and Table 5.1, it can be seen that Algorithm 1 converges more rapidly than Algorithm 2 and the NSPG method. Here, the symbol '—' means that the iteration step k has reached the upper limit 5000, but it did not deliver a solution.

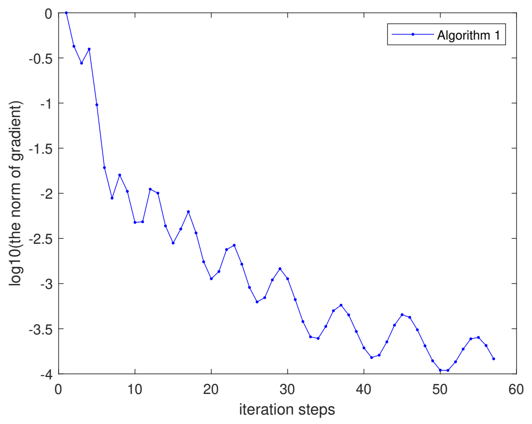


FIG. 5.1. *The convergence curve of the norm of the Riemannian gradient.*

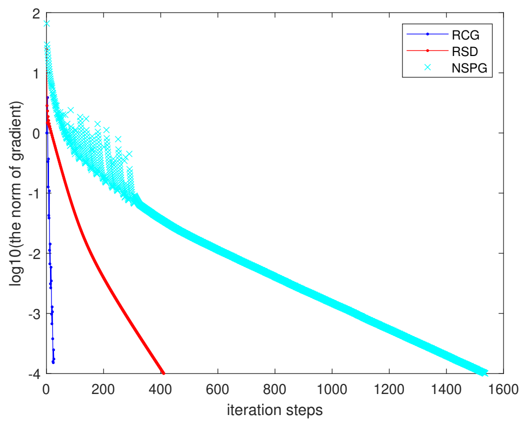


FIG. 5.2. *The convergence curves of Algorithms 1, 2, and the NSPG method.*

TABLE 5.1
Comparative results for Example 5.2.

$I_1, I_2, J_1, J_2, K_1, K_2$	Rank	Algorithm	IT	CPU(s)	$\| \operatorname{grad} f(\mathcal{X}_k)\|$
4,4,4,4,4,4	(1, 2, 1, 2)	RCG	**46**	**1.6712**	$\mathbf{8.2324 \times 10^{-5}}$
		RSD	395	8.3373	9.8042×10^{-5}
		NSPG	956	25.9390	9.9455×10^{-5}
6,5,4,4,3,2	(1, 2, 1, 2)	RCG	**27**	**1.0160**	$\mathbf{8.0931 \times 10^{-5}}$
		RSD	412	8.6599	9.7172×10^{-5}
		NSPG	1536	31.6997	9.9866×10^{-5}
6,5,4,4,3,2	(3, 2, 3, 2)	RCG	**112**	**3.7249**	$\mathbf{9.8653 \times 10^{-5}}$
		RSD	2255	44.1385	9.9713×10^{-5}
		NSPG	2443	51.6834	9.9958×10^{-5}
10, 10,10, 10, 10, 10	(2, 3, 5, 4)	RCG	**768**	**769.0**	$\mathbf{9.9821 \times 10^{-5}}$
		RSD	3255	1044.1385	9.9713×10^{-5}
		NSPG	—	—	—

EXAMPLE 5.3 (Application to image restoration). The modeling of colored 3D models for image restoration can be expressed as the tensor expression [8, 18]

$$\mathcal{M} = \mathcal{T} *_3 \mathcal{Q} + \mathcal{N},$$

where $\mathcal{M}, \mathcal{Q}, \mathcal{N} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ are third-order tensors, which represent the image, the object, and the noise, respectively. Additionally, $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_1 \times I_2 \times I_3}$ is a Toeplitz tensor derived from a third-order tensor $\mathcal{S} \in \mathbb{R}^{(I_1-1)\times(I_2-1)\times(I_3-1)}$, where $\mathcal{S}$ is a three-dimensional Gaussian function

$$\mathcal{S}(i,j,k) = \frac{1}{(\sqrt{2\pi}\sigma)^3} \exp\left(-\frac{i^2+j^2+k^2}{2\sigma^2}\right),$$

and $\mathcal{T}$ is obtained from

$$\mathcal{T}(i_1,i_2,i_3,j_1,j_2,j_3) = \mathcal{S}(j_1-i_1, j_2-i_2, j_3-i_3), \qquad 1 \le i_k, j_k \le I_k, k = 1,2,3.$$

Let $\mathcal{A} := \mathcal{T}$, $\mathcal{B} := \mathcal{M} - \mathcal{N}$, and considering the low-rank constraint, the above model can be rewritten as

$$\begin{aligned} &\min f(\mathcal{X}) = \frac{1}{2}\left\|\mathcal{A} *_3 \mathcal{X} - \mathcal{B}\right\|_F^2, \\ &\text{s.t. } \mathcal{X} \in \mathcal{M}_r := \left\{\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times I_3} \;\middle|\; \operatorname{rank}(\mathcal{X}) = r\right\}. \end{aligned}$$

In this experiment, we carefully selected four color images, namely "house", "pepper", "tree", and "scenery". The size of the first two pictures is $50 \times 50 \times 3$, while that of the latter two is $100 \times 100 \times 3$. Considering the low-rank nature of the images themselves and based on past experience, for the first two pictures, we set the rank as $(25, 25, 3)$. For the latter two images, the chosen rank is $(50, 50, 3)$. We use a 3D Gaussian function $(\sigma = 1)$ with random noise N. The noise level of both the "house" and "pepper" pictures is set to 0.01. The noise level of the "tree" image is set to 0.0001 and that of the "scenery" image is also set to 0.01. Accordingly, we use $N_1 = 0.001 * \text{tenrand}(50, 50, 3)$, $N_2 = 0.001 * \text{tenrand}(50, 50, 3)$, $N_3 = 0.0001 * \text{tenrand}(100, 100, 3)$, and $N_4 = 0.01 * \text{tenrand}(100, 100, 3)$ to represent the noise corresponding to the "house", "pepper", "tree", and "scenery" images, respectively.

The relative error (RE) and the peak signal-to-noise ratio (PSNR) are employed as evaluation indicators, that is,

$$RE = \frac{\|\mathcal{X} - \mathcal{Q}\|_F}{\|\mathcal{Q}\|_F}, \qquad PSNR = 10\log_{10}\left(\frac{\mathcal{Q}_{\max}^2 I_1 I_2 I_3}{\|\mathcal{X} - \mathcal{Q}\|_F^2}\right),$$

where $\mathcal{Q}$ represents the ground-truth tensor, $\mathcal{X}$ is the recovered tensor, and I_1, I_2, I_3 are the dimensions of the tensor $\mathcal{Q}$. The maximum pixel value of the ground-truth tensor $\mathcal{Q}$ is denoted by $\mathcal{Q}_{\max}$. The numerical results are presented in Figure 5.3 and Table 5.2.

From Figure 5.3, it can be clearly seen that the restored images obtained using the RCG method exhibit clear contour and texture features, which are very similar to those of the original images. On the other hand, as shown in Table 5.2, the RCG method achieves a higher peak signal-to-noise ratio (PSNR), a lower relative error (RE), and a shorter CPU time. The higher the PSNR value and the lower the RE value, the better the recovery performance. Therefore, the image restoration quality of the RCG method is superior to that of the RSD method and the NSPG method.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

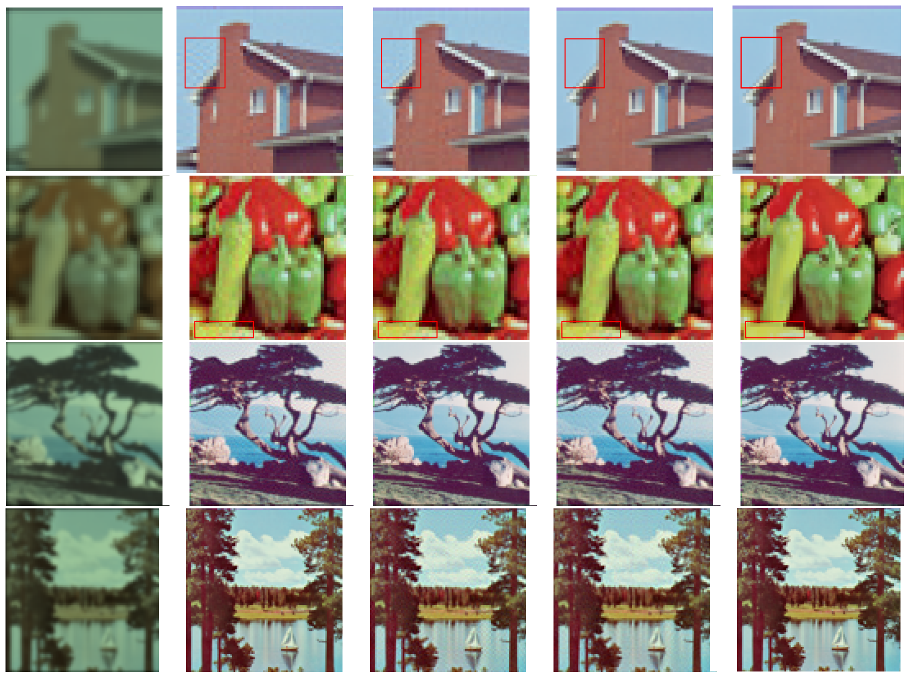


FIG. 5.3. *The restored images after 5000 iterations for three algorithms are shown. The first, second, third, and fourth images are the house images, the pepper images, the tree images, and the scenery images, respectively. ©Copyrighted content. Creative commons license does not apply.*

TABLE 5.2
The experiment results of Algorithms 1, 2, and the NSPG method.

Image		RE	PSNR	CPU(s)
house	RCG	**0.0108**	**43.9814**	**0.0302**
	NSPG	0.0187	39.2085	546.5812
	RSD	0.0256	36.4912	188.2715
peppers	RCG	**0.0218**	**39.2606**	**0.0313**
	NSPG	0.0375	34.5702	528.2247
	RSD	0.0521	31.7103	187.4417
tree	RCG	**0.0141**	**35.6968**	**0.2743**
	NSPG	0.0205	32.4635	7.2848e+03
	RSD	0.0295	29.3063	2.5523e+03
scenery	RCG	**0.0153**	**35.5203**	**0.2671**
	NSPG	0.0245	31.4505	7.3378e+03
	RSD	0.0356	28.1877	2.6123e+03

6. Conclusion.

In this paper, we consider a class of tensor least-squares problems that arise in images restoration. The Riemannian conjugate gradient method with exact line search are proposed to solve this problem. Additionally, a convergence analysis is also given. Numerical experiments demonstrate that the new algorithm is feasible and effective for solving

Problem 1. In particular, the simulation experiments for image restoration show that the new method has excellent performance.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, Princeton, 2008.

[2] R. Behera and D. Mishra, *Further results on generalized inverses of tensors via the Einstein product*, Linear Multilinear Algebra, 65 (2017), pp. 1662–1682.

[3] A. H. Bentbib, A. Khouia, and H. Sadok, *The LSQR method for solving tensor least-squares problems*, Electron. Trans. Numer. Anal., 55 (2022), pp. 92–111. https://etna.ricam.oeaw.ac.at/vol.55.2022/pp92-111.dir/pp92-111.pdf

[4] M. Brazell, N. Li, C. Navasca, and C. Tamon, *Solving multilinear systems via tensor inversion*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 542–570.

[5] E. G. Birgin, J.M. Martinez, and M. Raydan, *Nonmonotone spectral projected gradient methods on convex sets*, SIAM J. Optim., 10 (2000), pp. 1196–1211.

[6] R. Bro and S. De Jong, *A fast non-negativity-constrained least squares algorithm*, J. Chemometrics, 11 (1997), pp. 393–401.

[7] Z. Cao and P. Xie, *Perturbation analysis for t-product-based tensor inverse, Moore-Penrose inverse and tensor system*, Commun. Appl. Math. Comput., 4 (2022), pp. 1441–1456.

[8] L.-B. Cui, C. Chen, W. Li, and M. K. Ng, *An eigenvalue problem for even order tensors with its applications*, Linear Multilinear Algebra, 64 (2016), pp. 602–621.

[9] L. De Lathauwer, B. De Moor and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253-1278.

[10] X.-F. Duan, Y.-S. Zhang, Q.-W. Wang, and C.-M. Li, *Paige's algorithm for solving a class of tensor least squares problem*, BIT Numer. Math., 63 (2023), Paper No. 48, 26 pages.

[11] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, Comput. J., 7 (1964), pp. 149–154.

[12] M. Hanke, J. G. Nagy, and C. Vogel, *Quasi-Newton approach to nonnegative image restorations*, Linear Algebra Appl., 316 (2000), pp. 223–236.

[13] B. Huang and C. Ma, *An iterative algorithm to solve the generalized Sylvester tensor equations*, Linear Multilinear Algebra, 68 (2020), pp. 1175–1200.

[14] E. Kheirandish and A. Salemi, *Generalized bilateral inverses of tensors via Einstein product with applications to singular tensor equations*, Comput. Appl. Math., 42 (2023), Paper No. 343, 19 pages.

[15] A. Klarbring, *Quadratic programs in frictionless contact problems*, Internat. J. Engrg. Sci., 24 (1986), pp. 1207–1217.

[16] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT Numer. Math., 54 (2014), pp. 447–468.

[17] J. Landsberg, *Tensors: Geometry and Applications*, American Mathematical Society, Providence, 2012.

[18] X. Liu, L. Wang, J. Wang, and H. Meng, *A three-dimensional point spread function for phase retrieval and deconvolution*, Opt. Express, 20 (2012), pp. 15392–15405.

[19] Y. Luo, W. Huang, X. Li, and A. Zhang, *Recursive importance sketching for rank constrained least squares: algorithms and high-order convergence*, Oper. Res., 72 (2024), pp. 237–256.

[20] P. Soto-Quiros, *A least-squares problem of a linear tensor equation of third-order for audio and color image processing*, in 45th International Conference on Telecommunications and Signal Processing (TSP), Prague, IEEE Conference Proceedings, Los Alamitos, 2022, pp. 59–65.

[21] L. Tang, Y. Zhang, and H. Li, *On sketch-and-project methods for solving tensor equations*, Preprint on arXiv, 2022. https://arxiv.org/abs/2210.08241

[22] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[23] Q.-W. Wang and X. Xu, *Iterative algorithms for solving some tensor equations*, Linear Multilinear Algebra, 67 (2019), pp. 1325–1349.

[24] L. Xiao and T. Zhang, *A proximal-gradient homotopy method for the sparse least-squares problem*, SIAM J. Optim., 23 (2013), pp. 1062–1091.

[25] Z.-J. Xie, X.-Q. Jin, and V.-K. Sin, *An optimal preconditioner for tensor equations involving Einstein product*, Linear Multilinear Algebra, 68 (2020), pp. 886–902.