mathematics

MDPI

Article

An Improved Riemannian Trust-Region Method Combining Line Search Techniques with Applications in Matrix Optimization

Jiahui Yuan and Tao Yan *

School of Mathematics and Statistics, Nanjing University of Science and Technology, Nanjing 210094, China; 124130011646@njust.edu.cn
* Correspondence: tyan@njust.edu.cn

Abstract

We propose a hybrid Riemannian optimization framework that combines trust-region methods with retraction-based backtracking line search to reduce the cost of repeatedly solving subproblems corresponding to rejected trial steps. If a trust-region step is not accepted, the algorithm reuses the descent direction and performs a backtracking line search instead of resolving the subproblem. Under the stated assumptions, we establish $\liminf_{k\to\infty} \|\operatorname{grad} f(x_k)\|_{x_k} = 0$. Numerical experiments on three matrix optimization problems with orthogonality constraints, namely Brockett cost function minimization on the Stiefel manifold, Rayleigh quotient minimization on the sphere manifold, and independent component analysis on the Stiefel manifold, demonstrate that RTR-LS and LRTR-LS are computationally more efficient than classical Riemannian trust-region methods.

Keywords: Riemannian optimization; trust-region method; line search; retraction; global convergence

MSC: 90C30; 65K10; 90C26

1. Introduction

In this paper, we consider the optimization problem on a Riemannian manifold given by

$$\min_{x\in\mathcal{M}} f(x), \tag{1}$$

where $\mathcal{M}$ is a Riemannian manifold and f is a smooth objective function. Riemannian optimization extends classical unconstrained optimization from Euclidean spaces to nonlinear geometric settings and has been successfully applied in numerous domains, including low-rank matrix completion [1,2], phase retrieval [3–5], phase synchronization [6,7], blind deconvolution [8], and dictionary learning [9,10].

Many algorithms have been developed for optimization on manifolds. In addition to the standard Riemannian gradient and Newton methods [11], a number of efficient approaches have been proposed in recent years for solving problem (1), such as Riemannian conjugate gradient methods [11–14], Riemannian quasi-Newton methods [12,15,16], Riemannian trust-region methods [12,15,17,18], Riemannian stochastic algorithms [9,19–21], as well as several other techniques [8,22–25].

Among the various approaches for Riemannian optimization, we focus in particular on trust-region methods on manifolds. At each iteration, a Riemannian trust-region method builds a quadratic model of the objective function on the tangent space at the current iterate, and computes a trial step by approximately minimizing this model within a neighborhood

Check for updates

Academic Editor: Manchun Leung

Received: 27 September 2026
Revised: 2 November 2026
Accepted: 6 November 2026
Published: 8 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

where the model is regarded as reliable. This auxiliary problem is known as the Riemannian trust-region subproblem. The overall efficiency of the method depends strongly on two factors: the accuracy with which the subproblem is solved and the strategy for updating the trust-region radius.

Compared with trust-region methods, Riemannian line-search methods usually have a lower per-iteration cost and are often attractive for large-scale problems. Such methods generate a descent direction in the tangent space and determine an acceptable step size by enforcing sufficient decrease along a retraction curve. Although line-search methods are generally simpler and cheaper to implement, trust-region methods are usually more robust, as they exploit second-order model information and incorporate a principled mechanism for accepting or rejecting trial steps. This complementarity suggests that the two strategies may be combined advantageously. We note that the strategy has been adopted in Euclidean optimization by [26]. Our work differs in that it is developed in the Riemannian setting and extends their approach to this broader geometric framework.

In standard trust-region methods, a common practice is to enlarge or shrink the trust-region radius by a fixed factor based on the quality of the trial step. However, when a trial point is rejected, this mechanism may require resolving the trust-region subproblem several times within a single outer iteration, which can be computationally expensive for certain problems [27]. To alleviate this drawback, adaptive strategies for updating the trust-region radius have been developed. In these approaches, the radius is adjusted more flexibly by exploiting information available at the current iterate, such as the Euclidean gradient and the inverse Hessian [28,29].

Another key component of trust-region methods is the numerical solver for the subproblem [30]. The truncated conjugate gradient method [11] is one of the most widely used choices. However, its computational cost generally increases with problem size, since solving the trust-region subproblem may require many inner iterations. Therefore, it is desirable to reduce the number of repeated subproblem solves, especially when a candidate step has already been computed but is rejected only because it is too aggressive.

This observation motivates the incorporation of Riemannian line-search ideas into the Riemannian trust-region framework. In particular, when the computed trust-region step already provides a sufficient descent direction, it is natural to consider whether one can reuse this direction and perform a cheaper step-size correction, rather than immediately reducing the trust-region radius and resolving the subproblem. Such a strategy is especially appealing on manifolds, where a backtracking procedure can be implemented along a retraction curve.

Motivated by this idea and by the work of [26], we propose a hybrid framework that incorporates a retraction-based backtracking line search into the Riemannian trust-region algorithm. When a subproblem solver such as the Steihaug–Toint truncated conjugate gradient method [31] is employed, the resulting trial step d_k naturally satisfies a sufficient descent condition [17]. This property allows the algorithm to safely reuse the computed direction whenever the original trust-region trial point is rejected. Instead of repeatedly solving new trust-region subproblems, our method performs a backtracking line search along the retraction associated with d_k. This strategy reduces computational overhead while preserving the robustness and convergence guarantees of the trust-region framework. The theoretical contributions of this work are as follows:

1. We establish global convergence in the sense that $\liminf_{k\to\infty} \|\operatorname{grad} f(x_k)\|_{x_k} = 0$, under standard assumptions including Lipschitz continuity of the Riemannian gradient, and the second-order retraction property;
2. We propose a novel mechanism that triggers a line search only when a trust-region trial step is rejected. This approach circumvents the high cost of repeatedly resolving

the manifold subproblem while ensuring a sufficient decrease in the objective function, thereby significantly reducing the total CPU time in large-scale settings;

3. Through problems such as Rayleigh quotient minimization, independent component analysis, and Brockett cost function minimization, we provide empirical evidence that the proposed hybrid RTR-LS and LRTR-LS framework outperforms classical Riemannian trust-region methods in terms of both iteration count and computational efficiency.

2. Preliminaries

We introduce the elements of manifold optimization that will be needed in the study of (1). The concept of retraction is given as follows:

Definition 1 ([11]). *Let $\mathcal{M}$ be a differentiable manifold. A retraction $R : T\mathcal{M} \to \mathcal{M}$ is defined as a smooth map that fulfills two specific criteria. Throughout this paper, we use R_x to represent the restriction of R to the tangent space $T_x\mathcal{M}$:*

1. *$R_x(0_x) = x$, $\forall x \in \mathcal{M}$, where 0_x denotes the zero element of $T_x\mathcal{M}$.*
2. *With the canonical identification $T_{0_x}T_x\mathcal{M} \cong T_x\mathcal{M}$, R_x satisfies*

$$DR_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}, \tag{2}$$

where $DR_x(0_x)$ denotes the differential of R_x at 0_x and $id_{T_x\mathcal{M}}$ denotes the identity map on $T_x\mathcal{M}$.

An important retraction R_x is the exponential mapping, denoted by Exp, satisfying $\mathrm{Exp}_x(\eta_x) = \gamma(1)$, where $\gamma(0) = x$, $\gamma'(0) = \eta_x$, and γ is the geodesic passing through x.

For a smooth real-valued function f defined on a Riemannian manifold, its Riemannian gradient is characterized as the sole tangent vector that fulfills the condition:

$$\langle \mathrm{grad} f(x), \xi \rangle_x = Df(x)[\xi], \quad \forall \xi \in T_x\mathcal{M},$$

where $Df(x)[\xi]$ denotes the directional derivative along the direction ξ [11]. The Riemannian Hessian of f is a linear mapping from $T_x\mathcal{M}$ to $T_x\mathcal{M}$ defined by

$$\mathrm{Hess} f(x)[\xi] = \tilde{\nabla}_\xi \mathrm{grad} f(x), \ \forall \xi \in T_x\mathcal{M},$$

where $\mathrm{Hess} f(x)[\xi]$ denotes the action of $\mathrm{Hess} f(x)$ on a tangent vector $\xi \in T_x\mathcal{M}$, and $\tilde{\nabla}$ is the Riemannian connection [11]. Assuming the objective function $f(x)$ admits a smooth extension to the embedding Euclidean space $\mathbb{R}^{n\times p}$, its Riemannian gradient $\mathrm{grad} f(x)$ and Riemannian Hessian $\mathrm{Hess} f(x)$ can be derived as follows:

$$\begin{aligned} \mathrm{grad}\, f(x) &= P_{T_x\mathcal{M}}(\nabla f(x)), \\ \mathrm{Hess}\, f(x)[\xi] &= P_{T_x\mathcal{M}}(D\,\mathrm{grad}\, f(x)[\xi]), \end{aligned} \tag{3}$$

where D is the Euclidean derivative. If $\mathcal{M}$ is the Euclidean space, we have

$$\begin{aligned} \mathrm{grad}\, f(x) &= \nabla f(x), \\ \mathrm{Hess}\, f(x) &= \nabla^2 f(x). \end{aligned} \tag{4}$$

The objective function f on $\mathcal{M}$ is lifted to an objective function $\hat{f}_x = f \circ R_x$ on $T_x\mathcal{M}$. Note that $\hat{f}_x$ is a real-valued smooth function on a vector space. If $\mathcal{M}$ is endowed with a Riemannian metric, we have

$$\mathrm{grad}\hat{f}_x(0_x) = \mathrm{grad} f(x).$$

As is well known, tangent vectors residing in different tangent spaces cannot be added directly. A mathematically natural way to address the issue is to use the parallel translation operator, denoted $\mathcal{P}$ [32]. However, parallel translation is sometimes computationally expensive, and for certain manifolds—such as the Stiefel manifold—no explicit formula is available [11]. Vector transport is a reasonable approximation of parallel translation [11], and it can be more amenable to computation. Vector transport is defined as follows.

Definition 2 ([11]). *The vector transport $\mathcal{T}$ on a manifold $\mathcal{M}$ is a smooth map*

$$T\mathcal{M} \oplus T\mathcal{M} \to T\mathcal{M} : (\eta, \xi) \mapsto \mathcal{T}_\eta(\xi) \in T\mathcal{M},$$

which satisfies the following properties for all $x \in \mathcal{M}$, where $\oplus$ denotes the Whitney sum $T\mathcal{M} \oplus T\mathcal{M} = \{(\xi_x, \eta_x) : \xi_x, \eta_x \in T_x\mathcal{M}, x \in \mathcal{M}\}$.

1. *There exists an associated retraction R such that $\mathcal{T}_{\eta_x}(\xi_x) \in T_{R_x(\eta_x)}\mathcal{M}$ for all $\eta_x, \xi_x \in T_x\mathcal{M}$,*
2. *$\mathcal{T}_{0_x}(\xi_x) = \xi_x$ for all $\xi_x \in T_x\mathcal{M}$,*
3. *$\mathcal{T}_{\eta_x}(a\xi_x + b\zeta_x) = a\mathcal{T}_{\eta_x}(\xi_x) + b\mathcal{T}_{\eta_x}(\zeta_x)$ for all $a, b \in \mathbb{R}$ and $\eta_x, \xi_x, \zeta_x \in T_x\mathcal{M}$.*

It is worth noting that if $\mathcal{M}$ is an embedded submanifold of a Euclidean space and $\mathcal{M}$ is endowed with a retraction R, then a particular choice of vector transport is given by

$$\mathcal{T}_{\eta_x}(\xi_x) := DR_x(\eta_x)[\xi_x],$$

where P_x denotes the orthogonal projector onto $T_x\mathcal{M}$. In [18], an isometric vector transport $\mathcal{T}_S$ is proposed; it is a special vector transport that satisfies the following condition

$$\langle \mathcal{T}_{S_{\eta_x}} \xi_x, \mathcal{T}_{S_{\eta_x}} \zeta_x \rangle_x = \langle \xi_x, \zeta_x \rangle_x.$$

In computing the step size α_k on $\mathbb{R}^n$, the strong Wolfe conditions are often used on $\mathcal{M}$ [33], which require α_k to satisfy

$$f\big(R_{x_k}(\alpha_k \eta_k)\big) \leq f(x_k) + c_1 \alpha_k \langle \operatorname{grad} f(x_k), \eta_k \rangle_{x_k}, \tag{5}$$

$$|\langle \operatorname{grad} f(R_{x_k}(\alpha_k \eta_k)), \mathcal{T}_{\alpha_k \eta_k} \eta_k \rangle_{R_{x_k}(\alpha_k \eta_k)}| \leq c_2 |\langle \operatorname{grad} f(x_k), \eta_k \rangle_{x_k}|. \tag{6}$$

where $0 < c_1 < c_2 < 1$, and η_k is the search direction. We call the conditions (5) and (6) the strong Wolfe conditions.

Proposition 1. *Let M be a Riemannian manifold with a retraction R. If a smooth objective function f on M is bounded below on $\{R_{x_k}(\alpha \eta_k) \mid \alpha > 0\}$ for $x_k \in M$ and for a descent direction $\eta_k \in T_{x_k}M$, and if constants c_1 and c_2 satisfy $0 < c_1 < c_2 < 1$, then there exists a step size α_k which satisfies the strong Wolfe conditions (5) and (6).*

We note that the strong Wolfe conditions (5) and (6) together with the existence of a step size satisfying them are also discussed in [12].

3. Riemannian Trust-Region Method with Backtracking Line Search

In the Riemannian trust-region method, the trial step $d \in T_{x_k}\mathcal{M}$ is obtained by solving the trust-region subproblem [11].

$$\begin{aligned} \min_{d \in T_{x_k}\mathcal{M}} \ & \phi_k(d) = \langle \mathrm{grad} f(x_k),\, d \rangle_{x_k} + \frac{1}{2} \langle \mathcal{H}_k d,\, d \rangle_{x_k} \\ \text{s.t.} \ & \|d\|_{x_k} \le \Delta_k, \end{aligned} \tag{7}$$

where $\Delta_k > 0$ is the trust-region radius and $\mathrm{grad} f(x_k) \in T_{x_k}\mathcal{M}$ denotes the Riemannian gradient of f at the current iterate x_k on the manifold $\mathcal{M}$. The operator $\mathcal{H}_k : T_{x_k}\mathcal{M} \to T_{x_k}\mathcal{M}$ is a self-adjoint approximation to the Riemannian Hessian.After obtaining a trial step d_k from (7), trust-region methods compute the ratio ρ_k between the actual reduction in the function value, $f(x_k) - f(R_{x_k}(d_k))$, and the predicted reduction, $\phi_k(0) - \phi_k(d_k)$:

$$\rho_k = \frac{f(x_k) - f(R_{x_k}(d_k))}{\phi_k(0) - \phi_k(d_k)}, \tag{8}$$

the trust-region radius Δ_k is updated according to the value of ρ_k.

Line search strategies are often effective in improving global descent behavior by enforcing a sufficient decrease condition along a given search direction. Motivated by these complementary advantages of the two approaches, we combine the trust-region mechanism with a line-search procedure to enhance both stability and practical efficiency. More specifically, the trust-region subproblem is first used to generate a candidate direction in the tangent space. If the step d_k is not successful, that is, if $f(R_{x_k}(d_k)) > f(x_k)$, then a line search step is performed along this direction to determine a suitable step length. This hybrid strategy preserves the geometric structure of the Riemannian setting while improving the adaptability of the algorithm in challenging optimization scenarios.

We now describe the proposed trust-region algorithm with a backtracking line search. At each iteration, the algorithm approximately solves a quadratic trust-region subproblem in the current tangent space to obtain a trial step. If the full step decreases the objective function, it is accepted, and the ratio of actual to predicted reduction determines the subsequent radius update. Otherwise, the computed direction is reused in a backtracking line search along the retraction curve, rather than immediately resolving the subproblem. The accepted tangent step is then used to construct the SR1 pair, and the model information is transported to the new tangent space. The framework is summarized in Algorithm 1, followed by the subproblem solver and the SR1 update.

For the sequences generated by Algorithm 1 above, we have the following properties.

Lemma 1 ([11]). *A vector $d^* \in T_x\mathcal{M}$ is a solution of the subproblem (7) if there exists $\lambda^* \ge 0$ such that*

$$(\mathcal{H} + \lambda^* \,\mathrm{id}_{T_x\mathcal{M}}) d^* = -\,\mathrm{grad} f(x), \tag{9}$$

$$\left\langle v,\, (\mathcal{H} + \lambda^* \,\mathrm{id}_{T_x\mathcal{M}}) v \right\rangle_x \ge 0, \forall v \in T_x\mathcal{M}, \tag{10}$$

which means $(\mathcal{H} + \lambda^* \,\mathrm{id}_{T_x\mathcal{M}})$ is positive semidefinite on $T_x\mathcal{M}$. Furthermore,

$$\lambda^*(\Delta - \|d^*\|_x) = 0, \tag{11}$$

and the solution of the trust-region problem is given by

$$d^* = -(\mathcal{H} + \lambda^* \,\mathrm{id}_{T_x\mathcal{M}})^{-1} \,\mathrm{grad} f(x) + v, \qquad v \in \ker(\mathcal{H} + \lambda^* I), \tag{12}$$

where $(H + \lambda^* \,\mathrm{id}_{T_x\mathcal{M}})^{-1}$ denotes the pseudoinverse of the operator $\mathcal{H} + \lambda^* \,\mathrm{id}_{T_x\mathcal{M}}$ on $T_x\mathcal{M}$, and v is chosen so that $\|d^*\|_x \le \Delta$ and $\lambda^*(\Delta - \|d^*\|_x) = 0$.

Algorithm 1 Riemannian trust-region framework with backtracking line search (RTR-LS)

Given a Riemannian manifold $\mathcal{M}$, $x_0 \in \mathcal{M}$, an initial Hessian approximation operator $\mathcal{B}_0$, and an initial trust-region radius $\Delta_0 = 1$. Set the trust-region parameters $0 < c_2 < c_3 < 1 < c_1$, $0 < \theta < 1$. Initialize the iteration counter as $k := 1$;
loop
  Solve the trust-region subproblem (7)
  Compute $f(R_{x_k}(d_k))$;
  if $f(R_{x_k}(d_k)) < f(x_k)$ **then**
    Compute

$$x_{k+1} = R_{x_k}(d_k); \tag{13}$$

    Compute the ratio:

$$\rho_k = \frac{f(x_k) - f(x_{k+1})}{-\langle \operatorname{grad} f(x_k), d_k \rangle_{x_k} - \frac{1}{2}\langle d_k, \mathcal{H}_k d_k \rangle_{x_k}}, \tag{14}$$

    Update the trust-region radius:

$$\Delta_{k+1} = \begin{cases} \Delta_k, & \text{if } \rho_k \geq \theta \text{ and } \|d_k\| < \Delta_k \\ [c_2\|d_k\|, c_3\Delta_k], & \text{if } \rho_k < \theta \\ [\Delta_k, c_1\Delta_k], & \text{if } \rho_k \geq \theta \text{ and } \|d_k\| = \Delta_k \end{cases} \tag{15}$$

  else
    Find the minimum positive integer i_k such that

$$f(R_{x_k}(d_k^{(i_k)})) < f(x_k), \tag{16}$$

    where

$$\|d_k^{(i+1)}\|_{x_k} \in [\alpha_1, \alpha_2]\|d_k^{(i)}\|_{x_k}, 0 < \alpha_1 < \alpha_2 < 1, i = 1, 2, ..., i_k \tag{17}$$

    Compute

$$x_{k+1} = R_{x_k}(d_k^{(i_k)}); \tag{18}$$

    Set

$$\Delta_{k+1} \in \{\|d_k^{(i_k)}\|_{x_k}, c_3\Delta_k\}; \tag{19}$$

  end if
  Compute $\operatorname{grad} f(x_{k+1})$ and update $\mathcal{H}_{k+1}$;
  if a stopping criterion is satisfied **then**
    return x_{k+1};
  end if
  Set $k := k + 1$;
end loop

Lemma 2 ([11]). *The global convergence theory of trust-region algorithms only requires that the computed trial step d_k satisfies*

$$\phi_k(0) - \phi_k(d_k) \geq \beta\|\operatorname{grad} f(x_k)\|_{x_k} \min\left\{\Delta_k, \frac{\|\operatorname{grad} f(x_k)\|_{x_k}}{\|\mathcal{H}_k\|_{x_k}}\right\}, \tag{20}$$

for all k, where β is a positive constant. In Algorithm 1, the truncated conjugate gradient method is used to obtain the returned value d_k, where $\beta = \frac{1}{2}$ is taken. Inequality (20) is clearly satisfied if d_k is the exact solution of (7).

Theorem 1. *If d^* is a solution of (7) and satisfies $\|d^*\|_x \leq \Delta$, and $\operatorname{grad} f(x) \neq 0$, then*

$$\langle d^*, \operatorname{grad} f(x)\rangle_x \leq -\beta\|\operatorname{grad} f(x)\|_x \min\left\{\Delta, \frac{\|\operatorname{grad} f(x)\|_x}{\|\mathcal{H}\|_x}\right\}. \tag{21}$$

Here, $\Delta > 0$ denotes the trust-region radius and $\beta \leq 1/4$.

Proof. By the optimality conditions, there exists $\lambda^* \geq 0$ such that

$$A := \mathcal{H} + \lambda^* I \succeq 0, \qquad Ad^* = -\mathrm{grad} f(x), \qquad \lambda^*(\Delta - \|d^*\|_x) = 0.$$

Since $\mathrm{grad} f(x) \neq 0$, we have $\|A\|_x > 0$. Moreover,

$$\begin{aligned} \|\mathrm{grad} f(x)\|_x^2 &= \|Ad^*\|_x^2 \\ &\leq \|A\|_x \langle d^*, Ad^* \rangle_x \\ &= \|A\|_x [-\langle \mathrm{grad} f(x), d^* \rangle_x], \end{aligned}$$

hence

$$-\langle \mathrm{grad} f(x), d^* \rangle_x \geq \frac{\|\mathrm{grad} f(x)\|_x^2}{\|A\|_x}. \tag{22}$$

If $\|\mathcal{H}\|_x = 0$, the subproblem is linear, and its minimizer is $d^* = -\Delta \mathrm{grad} f(x) / \|\mathrm{grad} f(x)\|_x$. The result follows immediately. We therefore assume $\|\mathcal{H}\|_x > 0$.

Case I: $\|d^*\|_x < \Delta$. Complementary slackness gives $\lambda^* = 0$. According to (20), we have

$$\begin{aligned} \phi(0) - \phi(d^*) &= -\langle d^*, \mathrm{grad} f(x) \rangle_x - \frac{1}{2} \langle \mathcal{H} d^*, d^* \rangle_x \\ &\geq \beta \| \mathrm{grad} f(x) \|_x \min\left\{ \Delta, \frac{\| \mathrm{grad} f(x) \|_x}{\|\mathcal{H}\|_x} \right\}. \end{aligned}$$

Case II: $\|d^*\|_x = \Delta$. From $\lambda^* d^* = -\mathrm{grad} f(x) - \mathcal{H} d^*$, we obtain

$$\lambda^* \Delta \leq \|\mathrm{grad} f(x)\|_x + \|\mathcal{H}\|_x \Delta,$$

therefore,

$$\|A\|_x \leq \|\mathcal{H}\|_x + \lambda^* \leq 2\|\mathcal{H}\|_x + \frac{\|\mathrm{grad} f(x)\|_x}{\Delta},$$

using (22), we have

$$\begin{aligned} -\langle \mathrm{grad} f(x), d^* \rangle_x &\geq \frac{\|\mathrm{grad} f(x)\|_x^2}{2\|\mathcal{H}\|_x + \|\mathrm{grad} f(x)\|_x / \Delta} \\ &\geq \frac{1}{2} \|\mathrm{grad} f(x)\|_x \min\left\{ \Delta, \frac{\|\mathrm{grad} f(x)\|_x}{2\|\mathcal{H}\|_x} \right\} \\ &\geq \frac{1}{4} \|\mathrm{grad} f(x)\|_x \min\left\{ \Delta, \frac{\|\mathrm{grad} f(x)\|_x}{\|\mathcal{H}\|_x} \right\}, \end{aligned}$$

thus, there always exists a positive number β such that (21) holds in both cases. □

We now show that the angle between d and $-\mathrm{grad} f(x)$ is a monotonic function of Δ. To establish this result, we assume that in the hard case, we always choose a solution at the boundary of the trust region.

Theorem 2. *Assume that* $\mathrm{grad} f(x) \neq 0$. *Let* $\Delta^{(2)} > \Delta^{(1)} > 0$, *and define* $d_{\Delta^{(1)}}$ *and* $d_{\Delta^{(2)}}$ *to be the solutions of the Riemannian trust-region subproblem (1) when* $\Delta = \Delta^{(1)}$ *and* $\Delta = \Delta^{(2)}$, *respectively. Then*

$$\cos \angle (d_{\Delta^{(1)}}, -\mathrm{grad} f(x)) \geq \cos \angle (d_{\Delta^{(2)}}, -\mathrm{grad} f(x)), \tag{23}$$

where

$$\cos \angle (u, v) = \frac{\langle u, v \rangle_x}{\|u\|_x \, \|v\|_x}. \tag{24}$$

Proof. From Lemma 1, there exist $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ such that

$$(\mathcal{H} + \lambda_1 \mathcal{I}) d_{\Delta^{(1)}} = -\operatorname{grad} f(x), \quad \lambda_1(\Delta^{(1)} - \|d_{\Delta^{(1)}}\|_x) = 0, \tag{25}$$

$$(\mathcal{H} + \lambda_2 \mathcal{I}) d_{\Delta^{(2)}} = -\operatorname{grad} f(x), \quad \lambda_2(\Delta^{(2)} - \|d_{\Delta^{(2)}}\|_x) = 0. \tag{26}$$

We first show that $\lambda_1 \geq \lambda_2$. Suppose by contradiction that $\lambda_2 > \lambda_1$, so we have $\lambda_2 > 0$, and it yields $\|d_{\Delta^{(2)}}\|_x = \Delta^{(2)}$. Because $\mathcal{H} + \lambda_1 \mathcal{I}$ is positive semidefinite on $T_x\mathcal{M}$, i.e.,

$$\langle d_{\Delta^{(1)}} - d_{\Delta^{(2)}}, (\mathcal{H} + \lambda_1 \mathcal{I})(d_{\Delta^{(1)}} - d_{\Delta^{(2)}}) \rangle_x \geq 0, \tag{27}$$

using the condition (26), (27) becomes

$$\begin{aligned} &\langle d_{\Delta^{(1)}} - d_{\Delta^{(2)}}, (\mathcal{H} + \lambda_2 \mathcal{I}) d_{\Delta^{(2)}} - (\mathcal{H} + \lambda_1 \mathcal{I}) d_{\Delta^{(2)}} \rangle_x \\ =&\langle d_{\Delta^{(1)}} - d_{\Delta^{(2)}}, (\lambda_2 - \lambda_1) d_{\Delta^{(2)}} \rangle_x \\ =&(\lambda_2 - \lambda_1)\big(\langle d_{\Delta^{(1)}}, d_{\Delta^{(2)}} \rangle_x - \|d_{\Delta^{(2)}}\|_x^2\big), \end{aligned} \tag{28}$$

since $\|d_{\Delta^{(1)}}\|_x \leq \Delta^{(1)}$ and $\|d_{\Delta^{(2)}}\|_x = \Delta^{(2)}$,

$$\langle d_{\Delta^{(1)}}, d_{\Delta^{(2)}} \rangle_x \leq \|d_{\Delta^{(1)}}\|_x \|d_{\Delta^{(2)}}\|_x \leq \Delta^{(1)} \Delta^{(2)},$$

hence

$$(\lambda_2 - \lambda_1)(\Delta^{(1)} \Delta^{(2)} - {\Delta^{(2)}}^2) < 0,$$

which contradicts (27). Thus $\lambda_1 \geq \lambda_2$. We prove the result (23) considering three cases.

Case I: If $\mathcal{H} + \lambda_2 \mathcal{I}$ is positive definite, then

$$d_{\Delta^{(i)}} = -(\mathcal{H} + \lambda_i \mathcal{I})^{-1} \operatorname{grad} f(x),$$

define

$$\cos(\langle d_\Delta, -\operatorname{grad} f(x) \rangle) = \theta(\lambda) = \frac{\langle \operatorname{grad} f(x), (\mathcal{H} + \lambda \mathcal{I})^{-1} \operatorname{grad} f(x) \rangle_x}{\|(\mathcal{H} + \lambda \mathcal{I})^{-1} \operatorname{grad} f(x)\|_x \| \operatorname{grad} f(x)\|_x},$$

by spectral decomposition in $T_x\mathcal{M}$, $\theta(\lambda)$ is monotonically increasing for $\lambda \geq \max\{0, -\sigma_n(\mathcal{H})\}$, where $-\sigma_n(\mathcal{H})$ is the smallest eigenvalue on $T_x\mathcal{M}$.

Case II: If $\mathcal{H} + \lambda_1 \mathcal{I}$ is singular, then $\lambda_1 = -\sigma_n(\mathcal{H})$, and we have the condition $\lambda_1 \geq \lambda_2$ and $\mathcal{H} + \lambda_2 \mathcal{I}$ is positive semidefinite, so that $\lambda_1 = \lambda_2 = -\sigma_n(\mathcal{H})$. In this case, the minimizers may not be unique. By the selection rule in the theorem, we choose boundary minimizers satisfying $\|d_{\Delta^{(i)}}\|_x = \Delta^{(i)}$, $i = 1, 2$.

$$\begin{aligned} \cos(\langle d_{\Delta^{(1)}}, -\operatorname{grad} f(x) \rangle) &= \frac{\langle \operatorname{grad} f(x), (\mathcal{H} - \sigma_n(\mathcal{H}) \mathcal{I})^{-1} \operatorname{grad} f(x) \rangle_x}{\Delta^{(1)} \| \operatorname{grad} f(x)\|_x} \\ &\geq \frac{\langle \operatorname{grad} f(x), (\mathcal{H} - \sigma_n(\mathcal{H}) \mathcal{I})^{-1} \operatorname{grad} f(x) \rangle_x}{\Delta^{(2)} \| \operatorname{grad} f(x)\|_x} \\ &= \cos(\langle d_{\Delta^{(2)}}, -\operatorname{grad} f(x) \rangle). \end{aligned} \tag{29}$$

Case III: If $\mathcal{H} + \lambda_1 \mathcal{I}$ is positive definite and $\lambda_2 = -\sigma_n(\mathcal{H})$, then by the monotonic increase of $\theta(\lambda)$ as $\lambda \to -\sigma_n(\mathcal{H})$, we obtain

$$\begin{aligned} \cos(\langle d_{\Delta^{(1)}}, -\operatorname{grad} f(x) \rangle) &= \frac{\langle \operatorname{grad} f(x), (\mathcal{H} + \lambda_1 \mathcal{I})^{-1} \operatorname{grad} f(x) \rangle_x}{\|(\mathcal{H} + \lambda_1 \mathcal{I})^{-1} \operatorname{grad} f(x)\|_x \| \operatorname{grad} f(x)\|_x} \\ &\geq \lim_{\lambda \to -\sigma_n(\mathcal{H})} \frac{\langle \operatorname{grad} f(x), (\mathcal{H} + \lambda \mathcal{I})^{-1} \operatorname{grad} f(x) \rangle_x}{\|(\mathcal{H} + \lambda \mathcal{I})^{-1} \operatorname{grad} f(x)\|_x \| \operatorname{grad} f(x)\|_x} \\ &= \frac{\langle \operatorname{grad} f(x), (\mathcal{H} - \sigma_n(\mathcal{H}) I)^{-1} \operatorname{grad} f(x) \rangle_x}{\|(\mathcal{H} - \sigma_n(\mathcal{H}) I)^{-1} \operatorname{grad} f(x)\|_x \| \operatorname{grad} f(x)\|_x} \\ &\geq \frac{\langle \operatorname{grad} f(x), (\mathcal{H} - \sigma_n(\mathcal{H}) I)^{-1} \operatorname{grad} f(x) \rangle_x}{\|d_{\Delta^{(2)}}\|_x \| \operatorname{grad} f(x)\|_x} \\ &= \cos \angle\big(d_{\Delta^{(2)}}, -\operatorname{grad} f(x)\big), \end{aligned} \tag{30}$$

this completes the proof. □

The preceding results concern exact global minimizers of the Riemannian trust-region subproblem. Since computing them can be costly for large-scale problems, we use iterative methods such as truncated conjugate gradient (TCG) to approximately solve the subproblem [17,34].

As detailed in Algorithm 2, the inner CG iterations are systematically truncated whenever the iterates encounter a direction of negative curvature or breach the trust-region boundary Δ_k, thereby ensuring both global convergence and computational efficiency. Note that we use superscripts to denote the evolution of d within the inner iteration, while subscripts are used in the outer iteration. Another component that needs to be addressed in Algorithm 1 is the update strategy for the Riemannian Hessian approximation. Several strategies have been proposed in the literature. In this paper, we adopt the symmetric rank-one update method. The Riemannian trust-region method based on the symmetric rank-one update can be interpreted as a Riemannian generalization of the classical Euclidean SR1 trust-region method [33]. Several aspects discussed in [11] remain applicable to Algorithm 3. In this paper, we follow the scheme proposed in [18].

Algorithm 2 Truncated Conjugate Gradient method for Trust-Region Subproblem [11]

set $d^0 = 0$, $r_0 = g_k$, $\delta_0 = -r_0$, $j = 0$;
loop
  if $\langle \delta_j, \mathcal{H}_k \delta_j \rangle_{x_k} \leq 0$ **then**
    Compute $\tau \geq 0$ such that $\|d^j + \tau \delta_j\|_{x_k} = \Delta_k$;
    return $d_k := d^j + \tau \delta_j$;
  end if
  Set step size: $\alpha_j = \frac{\langle r_j, r_j \rangle_{x_k}}{\langle \delta_j, \mathcal{H}_k \delta_j \rangle_{x_k}}$;
  Update step: $d^{j+1} = d^j + \alpha_j \delta_j$;
  if $\|d^{j+1}\|_{x_k} \geq \Delta_k$ **then**
    Compute $\tau \geq 0$ such that $\|d^j + \tau \delta_j\|_{x_k} = \Delta_k$;
    return $d_k := d^j + \tau \delta_j$;
  end if
  Update residual: $r_{j+1} = r_j + \alpha_j \mathcal{H}_k \delta_j$; Orthogonal projection: $r_{j+1} = P_{T_{x_k}\mathcal{M}}(r_{j+1})$;
  if a stopping criterion is satisfied **then**
    return $d_k := d^{j+1}$;
  end if
  Update conjugate direction parameter: $\beta_{j+1} = \frac{\langle r_{j+1}, r_{j+1} \rangle_{x_k}}{\langle r_j, r_j \rangle_{x_k}}$;
  Update new direction: $\delta_{j+1} = -r_{j+1} + \beta_{j+1} \delta_j$;
  Update $j = j + 1$;
end loop

Algorithm 3 Symmetric Rank-one Update on Riemannian Manifolds

Let $s_k = d_k$, where d_k denotes the final accepted step, and compute $y_k = \mathcal{T}_{S_{s_k}}^{-1} \operatorname{grad} f(R_{x_k}(s_k)) - \operatorname{grad} f(x_k)$;
if $|\langle s_k, y_k - \mathcal{H}_k s_k \rangle_{x_k}| \geq 10^{-6} \|s_k\|_{x_k} \|y_k - \mathcal{H}_k s_k\|_{x_k}$. **then**
  Update the Hessian approximation using the SR1 formula:

$$\tilde{\mathcal{H}}_{k+1} = \mathcal{H}_k + \frac{(y_k - \mathcal{H}_k s_k) \otimes (y_k - \mathcal{H}_k s_k)}{\langle s_k,\ y_k - \mathcal{H}_k s_k \rangle_{x_k}}, \tag{31}$$

else
  $\tilde{\mathcal{H}}_{k+1} \leftarrow \mathcal{H}_k$;
end if
Compute

$$\mathcal{H}_{k+1} \leftarrow \mathcal{T}_{S_{s_k}} \circ \tilde{\mathcal{H}}_{k+1} \circ \mathcal{T}_{S_{s_k}}^{-1}$$

The Analysis of Convergence

The trial step d_k computed by Algorithm 2 at each iteration is such that there exists a positive constant β that satisfies (20) and (21). In Algorithm 1, we can see that if d_k is not accepted, that is, if $f(R_{x_k}(d_k)) > f(x_k)$, we can perform a backtracking line search along the direction d_k. In the backtracking line search, we determine the minimum positive integer i_k such that

$$f(R_{x_k}(\alpha^{i_k} d_k)) < f(x_k), \tag{32}$$

where $\alpha \in (0,1)$ is a positive constant. The sequence of trial points $d_k^{(i)}$ satisfies (17):

$$\|d_k^{(i+1)}\|_{x_k} \in [\alpha_1, \alpha_2]\|d_k^{(i)}\|_{x_k}, 0 < \alpha_1 < \alpha_2 < 1,$$

and from Theorem 2, we have

$$\cos \angle(d_k^{(i)}, -\operatorname{grad} f(x_k)) \geq \cos \angle(d_k, -\operatorname{grad} f(x_k)). \tag{33}$$

For $i = 0, 1, 2, \ldots\ldots, d_k^{(0)} = d_k$.

To analyze the new algorithm, we need to make the following assumptions:

Assumption 1. *The sequence $\{x_k\}$ generated by Algorithm 1 is contained in a compact set $\mathcal{K} \subseteq \mathcal{M}$.*

Assumption 2. *For the pullbacks $\hat{f}_x = f \circ R_x$, there exists a constant $M > 0$, independent of x, such that*

$$\|\nabla \hat{f}_x(\eta) - \nabla \hat{f}_x(\zeta)\|_x \leq M\|\eta - \zeta\|_x,$$

for all $x \in \mathcal{K}$ and $\|\eta\|_x, \|\zeta\|_x \leq \Delta_{max}$, where $0 < \Delta_k \leq \Delta_{\max} < \infty$ for all k. Since R is a second-order retraction,

$$\nabla^2 \hat{f}_x(0) = \operatorname{Hess} f(x),$$

$$\|\nabla^2 \hat{f}_x(0)\|_x = \|\operatorname{Hess} f(x)\|_x \leq M, \forall x \in \mathcal{K}.$$

Based on the above assumptions, we will prove that the sequence $\{x_k\}$ generated by Algorithm 1 is globally convergent and satisfies:

$$\liminf_{k \to \infty} \|\operatorname{grad} f(x_k)\|_{x_k} = 0, \tag{34}$$

we proceed by contradiction. If (34) were not true, then there would exist a constant $0 < \delta < 1$ such that

$$\|\operatorname{grad} f(x_k)\|_{x_k} \geq \delta > 0, \tag{35}$$

for all k. There exists a positive constant μ such that

$$\phi_k(0) - \phi_k(d_k) \geq \mu \min\{\Delta_k, 1/\|\mathcal{H}_k\|_{x_k}\}, \tag{36}$$

$$\langle d_k, \operatorname{grad} f(x_k)\rangle_{x_k} \leq -\mu \min\{\Delta_k, 1/\|\mathcal{H}_k\|_{x_k}\}, \tag{37}$$

where $\mu > 0$ is a positive constant independent of k, whose existence follows from (20) and (21).

Lemma 3. *Suppose that Assumption 1 holds, and define the set $\mathcal{I}$ as the index set of iterations in Algorithm 1 for which $\rho_k \geq \theta$. Then*

$$\lim_{k \to \infty,\, k \in \mathcal{I}} \min\{\Delta_k, 1/M_k\} = 0, \tag{38}$$

where

$$M_k := 1 + \max_{1 \leq i \leq k} \|\mathcal{H}_i\|_{x_i}, \tag{39}$$

Proof. Algorithm 1 ensures that the sequence $\{f(x_k)\}$ is monotonically decreasing. Therefore, we can obtain

$$\begin{aligned}\sum_{k=1}^{\infty}[f(x_k)-f(x_{k+1})] &\geq \sum_{k\in\mathcal{J}}[f(x_k)-f(x_{k+1})]\\ &\geq \theta\sum_{k\in\mathcal{J}}[\phi_k(0)-\phi_k(d_k)]\\ &\geq \theta\mu\sum_{k\in\mathcal{J}}\min\{\Delta_k, 1/\|\mathcal{H}_k\|_{x_k}\},\end{aligned} \tag{40}$$

here, the second inequality follows from (8) and the definition $\rho_k \geq \theta$. Since Assumption 1 and Assumption 2 imply that the sequence $\{f(x_k)\}$ is bounded below, it follows from (39) and (40) that

$$\sum_{k\in\mathcal{J}}\min\{\Delta_k,\ 1/M_k\} < \infty, \tag{41}$$

so that

$$\lim_{k\to\infty,\, k\in\mathcal{J}}\min\{\Delta_k,\ 1/M_k\} = 0.$$

□

For the above problem, we also have the following result. This limit also holds for $k \notin \mathcal{J}$. If $\mathcal{J}$ is a finite set, it follows from Equations (15) and (19) that $\Delta_{k+1} \leq c_3\Delta_k$ for all sufficiently large k, so that $\{\Delta_k\} \to 0$, which gives

$$\lim_{k\to\infty}\min\{\Delta_k,\ 1/M_k\} = 0, \tag{42}$$

if $\mathcal{J}$ is infinite, we consider an index $j \notin \mathcal{J}$, and let $\hat{j}$ be the greatest integer less than j such that $\hat{j} \in \mathcal{J}$. Since the sequence $\{M_k\}$ is monotonically increasing and $\Delta_j \leq c_1\Delta_{\hat{j}}$, we obtain

$$\min\{\Delta_j,\ 1/M_j\} \leq c_1\min\{\Delta_{\hat{j}},\ 1/M_{\hat{j}}\}. \tag{43}$$

Note that the limit (42) was established under the assumption that $\|\operatorname{grad} f(x_k)\|_{x_k}$ is bounded away from zero. It states that if the algorithm fails then either $\{\Delta_k\}$ converges to zero or $\|\mathcal{H}_k\|_{x_k}$ is unbounded. We will use this limit to arrive at a contradiction.

The next two lemmas consider the two cases under which the trust region is reduced by Algorithm 1.

Lemma 4. *If inequality (35) and Assumptions 1 and 2 are satisfied, then*

$$\Delta_k \geq \|d_k\|_{x_k} \geq \min\{\mu(1-\theta), 1\}/M_k, \tag{44}$$

for all sufficiently large k for which x_{k+1} is defined by (13) of Algorithm 1, and for which $\rho_k < \theta$.

Proof. The algorithm ensures that $\|d_k\|_{x_k} < \Delta_k$, so the first inequality holds.

If the second inequality is false, there exist infinitely many $k_i (i = 1, 2, \ldots)$ such that

$$\rho_{k_i} < \theta,$$

and

$$\|d_{k_i}\|_{x_{k_i}} < \min\{\mu(1-\theta), 1\}/M_{k_i} \tag{45}$$

combining the first inequality in (44), (42) and (45), we can have

$$\lim_{i\to\infty}\|d_{k_i}\|_{x_{k_i}} = 0. \tag{46}$$

By the Cauchy–Schwarz inequality and the definition of the operator norm [11], Assumption 2, (39) and (45), we obtain

$$\begin{aligned}\rho_{k_i} &= 1 + \frac{1}{2}\frac{\langle d_{k_i}, \mathcal{H}_{k_i} d_{k_i}\rangle_{x_{k_i}} - \langle d_{k_i}, \mathrm{Hess} f(\xi_{k_i})[d_{k_i}]\rangle_{x_{k_i}}}{\phi_{k_i}(0) - \phi_{k_i}(d_{k_i})} \\ &> 1 - \frac{1}{2}\frac{\|d_{k_i}\|_{x_{k_i}}\mu(1-\theta) + M\|d_{k_i}\|^2_{x_{k_i}}}{\phi_{k_i}(0) - \phi_{k_i}(d_{k_i})},\end{aligned} \tag{47}$$

the limit (46) implies that for sufficiently large i

$$M\|d_{k_i}\|^2_{x_{k_i}} \;\le\; \frac{1}{2}\|d_{k_i}\|_{x_{k_i}}\mu(1-\theta). \tag{48}$$

Moreover, (45) implies that $\|d_{k_i}\|_{x_{k_i}} < 1/M_{k_i}$, and since we also know that $\|d_{k_i}\|_{x_{k_i}} \le \Delta_{k_i}$, we see that

$$\frac{\|d_{k_i}\|_{x_{k_i}}}{\min\{\Delta_{k_i},\, 1/M_{k_i}\}} \;\le\; 1, \tag{49}$$

using (36), (48) and (49), for sufficiently large i, we obtain the following from (47):

$$\begin{aligned}\rho_{k_i} &> 1 - \frac{3}{4}\frac{\|d_{k_i}\|_{x_k}\mu(1-\theta)}{\mu\,\min\{\Delta_{k_i},\, 1/M_{k_i}\}} \\ &\ge 1 - \frac{3}{4}(1-\theta) \;>\; \theta,\end{aligned} \tag{50}$$

since $\theta < 1$. This contradicts (3). □

Lemma 5. *If inequality (35) and Assumptions 1 and 2 are satisfied, there is a constant $0 < \bar{\mu} \le 1$ such that*

$$\Delta_k \;>\; \|d_k^{(i_k)}\|_{x_k} \;\ge\; \bar{\mu}\,\min\{\mu(1-\theta),\, 1\}/M_k \tag{51}$$

for all sufficiently large k for which x_{k+1} is given by $x_{k+1} = R_{x_k}(d_k^{(i_k)})$ in (18) of Algorithm 1.

Proof. The first inequality follows from (17).

Assume that the second inequality does not hold. Then there exist infinitely many k such that $x_{k+1} = R_{x_k}(d_k^{(i_k)})$ and

$$\|d_k^{(i_k)}\|_{x_k} = o(1/M_k), \tag{52}$$

hence, we have from Assumption 2, (17), (33), (37), and [31] that

$$\begin{aligned}&f(x_k) - f\big(R_{x_k}(d_k^{(i_k-1)})\big) \\ &\quad\ge \|d_k^{(i_k-1)}\|_{x_k}\,\|\operatorname{grad} f(x_k)\|_{x_k}\cos\big(\angle(d_k^{(i_k-1)}, -\operatorname{grad} f(x_k))\big) \\ &\qquad - M\,\|d_k^{(i_k-1)}\|^2_{x_k}/2 \\ &\quad> \|d_k^{(i_k)}\|_{x_k}\,\|\operatorname{grad} f(x_k)\|_{x_k} \\ &\quad\cos(\angle(d_k, -\operatorname{grad} f(x_k))) - M\|d_k^{(i_k)}\|^2_{x_k}/(2\alpha_1^2) \\ &\quad\ge \|d_k^{(i_k)}\|_{x_k}\,\mu\,\min\{\Delta_k/\|d_k\|_{x_k},\, 1/\|\mathcal{H}_k\|_{x_k}\|d_k\|_{x_k}\} \\ &\qquad - M\|d_k^{(i_k)}\|^2_{x_k}/(2\alpha_1^2) \\ &\quad\ge \|d_k^{(i_k)}\|_{x_k}\,\mu\,\min\{1,\, 1/(M_k\Delta_k)\} - M\|d_k^{(i_k)}\|^2_{x_k}/(2\alpha_1^2) \\ &\quad> \frac{1}{M_k}\|d_k^{(i_k)}\|_{x_k}\,[\mu\min\{M_k,\, 1/\Delta_k\} - M\|d_k^{(i_k)}\|_{x_k}M_k/(2\alpha_1^2)].\end{aligned} \tag{53}$$

The radius update ensures that $\Delta_k \le \Delta_{\max}$ for all k. Using this condition and (52), we see that the right-hand side of (53) is positive for large k, which contradicts our definition of i_k. □

A global convergence result can now be obtained based on the above lemmas. The only assumption needed is that the growth of $\|\mathcal{H}_k\|_{x_k}$ is not excessively rapid.

Theorem 3. *Suppose that Assumptions 1 and 2 hold. If*

$$\sum_{k=1}^{\infty} \frac{1}{M_k} = \infty, \tag{54}$$

holds as an additional growth assumption on the model operators, then the sequence $\{x_k\}$ generated by Algorithm 1 satisfies

$$\liminf_{k\to\infty} \|\operatorname{grad} f(x_k)\|_{x_k} = 0. \tag{55}$$

Proof. We proceed by contradiction. Assume that there exists a positive constant $\delta > 0$ such that

$$\|\operatorname{grad} f(x_k)\|_{x_k} \geq \delta, \qquad \forall k. \tag{56}$$

If there are only finitely many k such that $\Delta_{k+1} < \Delta_k$, there exists a positive constant $\bar{\delta} > 0$ such that $\Delta_k > \bar{\delta}$ for all k. Due to (42), we have $\lim_{k\to\infty} 1/M_k = 0$, and hence (33) holds for all large k.

Now assume that there are infinitely many k such that $\Delta_{k+1} < \Delta_k$. From Lemma 4 and Lemma 5, there exists an integer $\bar{k}$ such that if $k \geq \bar{k}$ and $\Delta_{k+1} < \Delta_k$, then

$$\Delta_k \geq \bar{\mu} \min\{\mu(1-\theta),\ 1\}/M_k, \tag{57}$$

let $\hat{k}$ be an integer such that $\hat{k} \geq \bar{k}$ and $\Delta_{\hat{k}+1} < \Delta_{\hat{k}}$. We now show that

$$\Delta_k \geq \bar{\mu} c_2 \min\{\mu(1-\theta),\ 1\}/M_k \tag{58}$$

holds for any $k \geq \hat{k}$.

Case I: If $\Delta_{k+1} < \Delta_k$, then (57) holds, which implies (58) since $c_2 < 1$.

Case II: If $\Delta_{k+1} \geq \Delta_k$, let k' be the largest integer less than k such that $\Delta_{k'+1} < \Delta_{k'}$. We see from (19), Assumption 2, (44) and (51) that

$$\begin{aligned}
\Delta_k &\geq \Delta_{k'+1} \\
&\geq c_2 \|d_{k'}\|_{x_k} \\
&\geq c_2 \bar{\mu} \min\{\mu(1-\theta),\ 1\}/M_{k'} \\
&\geq c_2 \bar{\mu} \min\{\mu(1-\theta),\ 1\}/M_k,
\end{aligned}$$

for all sufficiently large k. Then, we have a constant $\nu > 0$ such that

$$\Delta_k \geq \nu/M_k \tag{59}$$

for all sufficiently large k.

Let $\mathcal{J}$ denote the set of successful iterations, which means $\rho_k \geq \theta$. Then, from (15)–(19) and the definition (39), for $k \notin \mathcal{J}$, either the full step is accepted with $\rho_k < \theta$, or backtracking is performed. In the former case, (15) gives $\Delta_{k+1} \leq c_3 \Delta_k$. In the latter case, (15) and (17) give

$$\|d_k^{(i_k)}\|_{x_k} \leq \alpha_2 \|d_k\|_{x_k} \leq \alpha_2 \Delta_k.$$

Thus, either choice in (19) satisfies

$$\Delta_{k+1} \leq q\Delta_k, \qquad q := \max\{c_3, \alpha_2\} < 1,$$

and we have

$$\Delta_{k+1} \leq c_1 \Delta_k, \qquad i \in \mathcal{J} \tag{60}$$

$$\Delta_{k+1} \leq q \Delta_k, \qquad i \notin \mathcal{J} \tag{61}$$

$$M_{k+1} \geq M_k, \qquad \text{for all } k \tag{62}$$

and from (59), we have

$$\min\{\Delta_k, 1/M_k\} \geq \frac{\min\{\nu, 1\}}{M_k}$$

with (41), we obtain

$$\sum_{k\in\mathcal{J}} \frac{1}{M_k} \leq \frac{1}{\min\{\nu,1\}} \sum_{k\in\mathcal{J}} \min\{\Delta_k, 1/M_k\} < \infty. \tag{63}$$

Because $c_1 > 1, q < 1$ are positive constants, let the indices of the failed steps be k_1, k_2, k_3, ..., i.e.,$k_j \notin \mathcal{J}$, which means Δ_k. Since each failed step leads to (61), we have $\Delta_{k_j} \leq q^{j-1}\Delta_{k_1}$. The sum of a geometric series is finite:

$$\sum_{j=1}^{\infty} \Delta_{k_j} \leq \Delta_{k_1} \sum_{j=1}^{\infty} q^{j-1} = \frac{\Delta_{k_1}}{1-q} < \infty, \tag{64}$$

this proves that

$$\sum_{k\notin\mathcal{J}} \Delta_k < \infty, \tag{65}$$

then with (63)–(65), we obtain

$$\sum_{k=1}^{\infty} 1/M_k < \infty, \tag{66}$$

(66) contradicts (54). □

4. Numerical Experiments

4.1. Experimental Settings

In this section, we verify the effectiveness of the proposed algorithms through a series of test problems. All the test problems in this section are matrix optimization problems with orthogonality or normalization constraints. The Brockett cost function minimization and the independent component analysis are formulated on the Stiefel manifold $St(n,p)$, while the Rayleigh quotient optimization is formulated on the sphere manifold S^{n-1}. These problems serve as representative benchmarks for Riemannian optimization algorithms. All experiments were conducted on a Lenovo laptop using MATLAB R2024a, equipped with an Intel Core i7 processor (4.9 GHz) and 24.00 GB of memory.

The performance of the Riemannian Trust-Region method with Symmetric Rank-One update and Truncated Conjugate Gradient (RTR-SR1-TCG), Riemannian Trust-Region method with Symmetric Rank-One update and Truncated Three-Term Conjugate Gradient (RTR-SR1-TTCG), Limited Riemannian Trust-Region method with Symmetric Rank-One update and Truncated Conjugate Gradient(LRTR-SR1-TCG), and Limited Riemannian Trust-Region method with Symmetric Rank-One update and Truncated Three-Term Conjugate Gradient (LRTR-SR1-TTCG) algorithms, as well as their variants combined with line search techniques, is demonstrated in solving trust-region subproblems (7).

The manifold operations are implemented through separate interfaces for the Riemannian metric, tangent-space projection, retraction, and vector transport. Tangent-space projection is used to control numerical drift in the inner iterations. In the implementation, the exponential mapping is chosen as the retraction operator, and parallel transport is used, ensuring that the updated points remain on the manifold.

In our algorithm, the parameters c_1, c_2, c_3 are set to 2, 0.25, and 0.6. The initial radius and the maximum radius are set to 1 and 200, and θ is chosen as 0.6; the error tolerance is set to 1×10^{-6} and the maximum number of iterations max_t to 8000, so as to achieve a good balance between computational efficiency and solution accuracy.

In Steps 4 and 10 of Algorithm 2, the boundary parameter τ is determined by solving for the positive root of the quadratic equation:

$$\tau^2\langle\delta_j,\delta_j\rangle_{x_k} + 2\tau\langle d^j,\delta_j\rangle_{x_k} + \langle d^j,d^j\rangle_{x_k} = \Delta_k^2.$$

A key computational advantage of the TCG scheme is its inverse-free property; it interacts with the Hessian $\mathcal{H}_k$ solely through matrix–vector products of the form $\mathcal{H}_k\delta_j$, thereby avoiding the prohibitive cost of explicit matrix inversion or the construction of the full Hessian operator.

To ensure numerical robustness in large-scale applications, practical implementations often incorporate several refinements. For instance, the residual norm $\langle r_{j+1}, r_{j+1}\rangle_{x_k}$ is cached to avoid redundant inner product computations in subsequent iterations. Additionally, we apply an orthogonal projection $P_{x_k}(\cdot)$ to the iterates d, r, and δ, which is crucial for mitigating numerical drift and ensuring that all search directions remain strictly within the relevant tangent space. The authors of [26] conducted similar experiments combining trust-region and line search techniques in the Euclidean setting, showing that performing a backtracking line search from a failed trust-region step reduces the overall computational cost. Motivated by this idea, we extend it to the Riemannian setting. And the SR1 formula is used to update $\mathcal{H}_k$. In our Riemannian trust-region algorithm, the trial step is computed by Algorithm 2, and $\mathcal{H}_k$ is updated by the symmetric rank-one formula in Algorithm 3. Let the step and gradient difference be defined as

$$s_k = d_k, \tag{67}$$

$$y_k = \mathcal{T}_{S_{s_k}}^{-1}(\mathrm{grad} f(x_{k+1})) - \mathrm{grad} f(x_k), \tag{68}$$

where $\mathcal{T}_{S_{s_k}}^{-1}(\cdot)$ denotes the vector transport operator along the retraction curve.

Backtracking is performed along the direction of the failed trial step and is based on truncated quadratic interpolation. If $f(R_{x_k}(d_k)) \geq f(x_k)$, we compute

$$\alpha_k = \max\{0.1,\ 0.5/[1 + (f(x_k) - f(R_{x_k}(d_k)))/(\langle \mathrm{grad} f(x_k), d_k\rangle_{x_k})]\}, \tag{69}$$

set $d_k := \alpha_k d_k$, and repeat this process until a lower function value is obtained.

To avoid numerical instability, the update is performed only when the denominator satisfies the regularity condition

$$|\langle s_k,\ y_k - \mathcal{H}_k[s_k]\rangle_{x_k}| \geq 10^{-6}\,\|s_k\|_{x_k}\,\|y_k - \mathcal{H}_k[s_k]\|_{x_k}, \tag{70}$$

otherwise, the SR1 update is skipped and $\mathcal{H}_k$ remains unchanged.

To reduce the storage and computational cost of the full SR1 update in Algorithm 3, we employ a limited-memory Riemannian SR1 scheme [35]. Instead of storing the entire Hessian approximation $\mathcal{H}_k \in \mathcal{L}(T_{x_k}\mathcal{M})$, we only keep the most recent m pairs of step and gradient-difference vectors:

$$\{(s_i,\ y_i)\}_{i=k-m+1}^{k}, \tag{71}$$

all stored vectors are transported to the current tangent space by vector transport at each accepted iteration. Given these pairs, the action of the approximate Hessian on any tangent vector $v \in T_{x_k}\mathcal{M}$ is computed implicitly as

$$\mathcal{H}_k v = \gamma_k v + \sum_{i=1}^{m_k} \alpha_i\,(y_i - \gamma_k s_i), \tag{72}$$

where the coefficients $\alpha = (\alpha_1, \ldots, \alpha_{m_k})$ are determined from the small linear system

$$(L_k + D_k + L_k^\top - \gamma_k S_k^\top S_k)\,\alpha = \left(\langle y_i - \gamma_k s_i,\ v\rangle_{x_k}\right)_{i=1}^{m_k}, \tag{73}$$

here, the matrix $((L_k + D_k + L_k^\top)$ represents the symmetrized version of $S_k^\top Y_k$, where L_k and D_k are the strict lower triangular and diagonal parts of the matrix with elements $\langle s_i, y_j \rangle_{x_k}$, and $S_k^\top S_k$ denotes the $m_k \times m_k$ matrix whose (i, j)-th element is the Riemannian inner product $\langle s_i, s_j \rangle_{x_k}$. γ_k is a scalar scaling parameter, typically chosen as

$$\gamma_k = \frac{\langle y_k, y_k \rangle_{x_k}}{\langle s_k, y_k \rangle_{x_k}}.$$

Consequently, the storage requirement is $O(mn)$ instead of $O(n^2)$ to reduce consumption. To avoid numerical instability, the memory update is performed only when the denominator satisfies the following regularity condition [33]

$$|\langle s_k, y_k - \mathcal{H}_k s_k \rangle_{x_k}| \geq 10^{-8} \|s_k\|_{x_k} \|y_k - \mathcal{H}_k s_k\|_{x_k}, \tag{74}$$

otherwise, the SR1 update is skipped and $\mathcal{H}_k$ remains unchanged.

We use TCG and TTCG to solve the trust-region subproblem. Compared with the standard two-term tCG method, the TTCG scheme introduces an additional correction direction, which improves the stability of the inner iterations under nonconvex or ill-conditioned curvature. The search direction is updated as

$$\delta_j = -r_j + \beta_j \delta_{j-1} + \gamma_j \delta_t, \tag{75}$$

where the additional term δ_t is a previously stored direction that helps correct loss of conjugacy. The coefficient γ_j is chosen adaptively to maintain numerical stability

$$\gamma_j = \frac{\langle r_j, r_j - r_t \rangle_{x_k}}{\langle \delta_t, r_j - r_t \rangle_{x_k}}.$$

In practice, this additional correction term is activated only when needed, and it effectively prevents the search direction from degenerating when the SR1 model is indefinite.

To avoid numerical instability in the SR1 update, we perform the update only if

$$|\langle s_k, y_k - \mathcal{H}_k[s_k] \rangle_{x_k}| \geq 10^{-6} \|s_k\|_{x_k} \|y_k - \mathcal{H}_k[s_k]\|_{x_k}, \tag{76}$$

otherwise, the SR1 update is skipped and $\mathcal{H}_k$ is kept unchanged.

4.2. Tests and Results

4.2.1. Brockett Cost Function Minimization on the Stiefel Manifold

In this experiment, we study a Brockett cost function minimization problem [36].

Consider a symmetric matrix $A \in \mathbb{R}^{n \times n}$, which is generated by a random orthogonal matrix Q such that $A = Q \Lambda Q^\top$, where Λ is a diagonal matrix, and $N = \text{diag}(1{:}p)$.

Our goal is to find a rank-r subspace on the Stiefel manifold that minimizes the following objective function:

$$\begin{aligned} \min_{x \in \mathbb{R}^{n \times p}} \quad & \text{trace}(x^\top A x N), \\ \text{s.t.} \quad & x^\top x = I_p, \end{aligned} \tag{77}$$

the constraint $x^\top x = I_p$ indicates that the column vectors of x are orthonormal on the Riemannian manifold, i.e., $x \in \text{St}(n, p)$. This model is commonly used in low-rank subspace learning and feature subspace optimization, which corresponds to finding an optimal r-dimensional subspace according to the spectral structure of matrix A.

We conducted numerical experiments on the aforementioned problem. Using Algorithm 2 and Algorithm TTCG as the trust-region subproblem solvers, we compared the Riemannian trust-region SR1 method (denoted as RTR-SR1) with its line-search-enhanced variant (denoted as RTR-SR1-LS), as well as the limited-memory SR1 method with and without line

search (denoted as LRTR-SR1-LS and LRTR-SR1, respectively). The results are summarized in Tables 1 and 2. In these tables, 'iter' denotes the number of iterations, 'time' represents the computational time (in seconds), and 'f' denotes the final objective function value.

Table 1. Performance comparison of RTR-SR1 solvers with and without line search for the Brockett problem.

n	p	RTR-SR1-TCG			RTR-SR1-TCG-LS			RTR-SR1-TTCG			RTR-SR1-TTCG-LS		
		Iter	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f
100	5	1767	90.13	35.01	631	34.54	35.98	2008	101.91	35.01	1675	87.19	35.12
	8	1269	111.14	120.10	1401	125.86	120.04	1928	166.32	120.01	1692	152.11	120.02
	10	1331	160.18	220.58	824	97.09	221.59	1945	242.79	220.09	1224	171.45	220.67
200	5	1815	310.00	35.15	1081	190.49	35.00	1857	308.35	35.01	419	67.79	35.82
	8	4525	2008.64	120.07	1123	494.39	121.96	3645	1575.62	120.05	2104	945.02	120.89
	10	2623	1854.98	220.10	1666	1206.01	220.25	1864	1289.51	220.49	988	716.14	221.48
300	5	2021	942.27	35.24	1587	753.59	35.34	2187	1032.06	35.07	1937	925.90	35.09
	8	1125	1414.36	122.84	611	795.54	125.68	3588	4545.39	120.03	3259	4327.24	120.32
	10	1766	3762.53	221.726	502	989.25	223.09	2727	7020.38	221.30	976	2743.43	223.77

Table 2. Performance comparison of LRTR-SR1 solvers with and without line search for the Brockett problem.

n	p	LRTR-SR1-TCG			LRTR-SR1-TCG-LS			LRTR-SR1-TTCG			LRTR-SR1-TTCG-LS		
		Iter	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f
100	5	682	10.30	35.00	571	8.88	35.00	717	10.60	35.00	457	7.34	35.00
	8	802	26.65	120.00	863	28.38	120.00	854	28.23	120.00	845	27.94	120.00
	10	1468	63.89	220.00	1031	44.85	220.00	1565	70.23	220.00	1269	61.69	220.01
200	5	929	53.57	35.01	915	54.02	35.00	891	51.24	35.01	945	55.15	35.00
	8	1956	273.64	120.04	1811	250.40	120.00	1674	233.40	120.01	1973	273.27	120.01
	10	1560	330.85	220.01	1407	292.70	220.01	1413	302.64	220.02	1247	267.76	220.03
300	5	1238	180.69	35.01	1234	174.65	35.02	1201	170.63	35.24	1125	159.35	35.03
	8	2544	966.50	120.18	3410	1323.71	120.02	2407	901.78	120.03	2313	904.04	120.03
	10	2243	2022.16	220.06	2381	2029.54	220.02	2928	2332.56	220.03	1962	1659.83	220.12

Across the tested Brockett problems, incorporating line search generally reduces the computational cost, particularly within the RTR-SR1 framework. For example, at $(n,p) = (100,5)$, RTR-SR1-TCG-LS reduces the iteration count from 1767 to 631 and the CPU time from 90.13 s to 34.54 s. Substantial reductions are also observed for TTCG: at $(n,p) = (200,5)$, adding line search reduces the iterations from 1857 to 419 and the CPU time from 308.35 s to 67.79 s. The limited-memory variants also benefit in several cases. At $(n,p) = (100,10)$, LRTR-SR1-TCG-LS reduces the CPU time from 63.89 s to 44.85 s, with both methods reporting an objective value of 220.00. At $(n,p) = (200,10)$, the corresponding times decrease from 330.85 s to 292.70 s, while both reported objective values are 220.01.

The improvements are not uniform. For example, LRTR-SR1-TCG-LS increases the CPU time from 966.50 s to 1323.71 s at $(n,p) = (300,8)$. The additional line-search evaluations may offset computational savings in some instances. Furthermore, where final objective values differ, timing reductions require confirmation at matched accuracy. Overall, the

tables indicate substantial benefits in many cases, alongside a dependence on the problem instance and subproblem solver.

4.2.2. Rayleigh Quotient Optimization Problem on the Sphere Manifold

We study a Rayleigh quotient optimization problem on the sphere manifold, which is a well-known problem in spectral analysis and feature extraction. A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is generated as $A = Q\Lambda Q^\top$, where $Q \in \mathbb{R}^{n\times n}$ is a random orthogonal matrix and $\Lambda = \mathrm{diag}(1{:}n)$ is a diagonal matrix. The spectral structure of A is determined by its eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_n$.

Our goal is to find the eigenvector corresponding to the smallest eigenvalue of A by solving the following Rayleigh quotient optimization problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & R(x) = x^\top Ax, \\ \text{s.t.} \quad & x^\top x = 1, \end{aligned}$$

the constraint $x^\top x = 1$ ensures that x lies on the sphere manifold $\mathcal{S}^{n-1}$, where all feasible solutions are normalized. Note that the sphere manifold S^{n-1} is a special case of the Stiefel manifold $St(n, 1)$, and hence this problem can also be regarded as a matrix optimization problem with a single orthonormal column. The minimum objective value is $\lambda_{\min}(A) = 1$, attained at a unit eigenvector associated with the smallest eigenvalue of A.

In the implementation, the unit normalization is chosen as the retraction operator to ensure the updated points remain on the sphere manifold during optimization. Furthermore, parallel transport is used to update the tangent vectors while preserving the geometric structure of the manifold, which facilitates the use of Riemannian optimization methods. This experiment leverages manifold-based optimization techniques, focusing on dynamically adjusting the trust-region radius and utilizing a geometric framework to achieve rapid convergence while maintaining the spherical constraints. Based on the above problem, numerical experiments are conducted using eight algorithms in Tables 3 and 4.

Table 3. Performance comparison of RTR-SR1 solvers with and without line search for Rayleigh quotient optimization.

n	RTR-SR1-TCG			RTR-SR1-TCG-LS			RTR-SR1-TTCG			RTR-SR1-TTCG-LS		
	Iter	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f
500	270	0.3949	1.00	259	0.3704	1.00	217	0.3392	1.00	252	0.3694	1.00
800	282	0.8897	1.00	263	0.8568	1.00	227	0.7248	1.00	275	0.8584	1.00
1000	384	1.8321	1.00	355	1.6692	1.00	311	1.4709	1.00	351	1.6605	1.00

Table 4. Performance comparison of LRTR-SR1 solvers with and without line search for Rayleigh quotient optimization.

n	LRTR-SR1-TCG			LRTR-SR1-TCG-LS			LRTR-SR1-TTCG			LRTR-SR1-TTCG-LS		
	Iter	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f
500	646	0.0569	1.00	493	0.0583	1.00	601	0.0571	1.00	531	0.0564	1.00
800	762	0.0971	1.00	676	0.1065	1.00	790	0.1144	1.00	681	0.1147	1.00
1000	1020	0.1815	1.00	824	0.1732	1.00	1113	0.2025	1.00	910	0.1724	1.00

Tables 3 and 4 report objective values of 1.00 at the displayed precision, consistent with the known minimum $\lambda_{\min}(A) = 1$. Adding line search reduces the iteration count of the TCG variants at every tested dimension. Within the RTR-SR1 framework, these reductions

are accompanied by lower CPU times in all three cases. For example, at $n = 1000$, the iterations decrease from 384 to 355, while the CPU time decreases from 1.8321 s to 1.6692 s. For LRTR-SR1-TCG at the same dimension, line search reduces the iteration count from 1020 to 824 and the CPU time from 0.1815 s to 0.1732 s. The limited-memory TTCG variant also benefits: at $n = 1000$, adding line search reduces the iterations from 1113 to 910 and the CPU time from 0.2025 s to 0.1724 s.

The timing improvements are not uniform across all variants. For example, at $n = 800$, RTR-SR1-TTCG-LS takes 0.8584 s compared with 0.7248 s without line search. For LRTR-SR1-TCG at this dimension, fewer iterations are accompanied by a slight increase in CPU time. Additional line-search evaluations may offset the savings from fewer iterations. Overall, the results show consistent iteration reductions for the TCG variants, while the computational benefit depends on the solver and problem dimension.

4.2.3. Independent Component Analysis on the Stiefel Manifold

ICA is a technique for blind source separation that seeks statistically independent components from multivariate observations. We consider the following ICA-related optimization model on the Stiefel manifold:

$$\begin{aligned} \min_{x \in \mathbb{R}^{n \times p}} \quad & f(x) = -\sum_{i=1}^{N} \left\| \mathrm{diag}(x^\top C_i x) \right\|_2^2 \\ \text{s.t.} \quad & x^\top x = I_p, \end{aligned} \tag{78}$$

here, $p \leq n$, $C_i \in \mathbb{R}^{n \times n}$, $i = 1, \ldots, N$, are symmetric matrices, and $\mathrm{diag}(M)$ denotes the vector of diagonal entries of M. The negative sign converts maximization of the summed squared diagonal entries into minimization. In ICA applications, the matrices may represent cumulants or time-lagged covariances. The optimization is conducted over the compact Stiefel manifold $St(n,p) = \{x \in \mathbb{R}^{n \times p} : x^\top x = I_p\}$ [18,37]. In our experiments, we set $N = 100$.

Tables 5 and 6 show that adding line search reduces CPU time for the TCG variants in all reported cases. For RTR-SR1-TCG at $(n,p) = (200,10)$, the iteration count decreases from 7817 to 3147, while the CPU time decreases from 1074.56 s to 532.80 s. The improvement is particularly consistent for LRTR-SR1-TCG-LS, which reduces both iteration counts and CPU times in every tested case. For example, at $(n,p) = (300,8)$, the iterations decrease from 2248 to 1226 and the CPU time from 197.49 s to 90.07 s. The TTCG variants also achieve lower CPU times in most cases. At $(n,p) = (100,8)$, RTR-SR1-TTCG-LS reduces the iterations from 4279 to 2074 and the CPU time from 157.91 s to 79.63 s. The final objective values are close relative to their magnitudes, although this alone does not establish matched accuracy.

The improvements are not uniform across all performance measures. At $(n,p) = (200,5)$, LRTR-SR1-TTCG-LS reduces the iterations from 1560 to 1496 but increases CPU time from 110.49 s to 113.94 s. Conversely, at $(n,p) = (300,8)$, RTR-SR1-TCG-LS requires more iterations but less CPU time than its non-LS counterpart. These cases show that iteration counts and runtime may not change together.

Table 5. Performance comparison of RTR-SR1 variants for ICA on the Stiefel manifold.

n	p	RTR-SR1-TCG			RTR-SR1-TCG-LS			RTR-SR1-TTCG			RTR-SR1-TTCG-LS		
		Iter	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f
100	5	2714	48.43	−4,802,636.99	2085	42.89	−4,802,643.84	2073	41.58	−4,802,642.97	1847	38.06	−4,802,653.17
	8	2529	92.39	−7,454,450.42	2266	85.33	−7,454,463.09	4279	157.91	−7,454,448.59	2074	79.63	−7,454,436.54
	10	4183	195.87	−9,128,972.77	2537	144.43	−9,128,971.91	2895	152.69	−9,128,973.03	2223	123.39	−9,128,971.22
200	5	1602	120.92	−19,602,270.07	1119	108.83	−19,602,279.54	2641	206.46	−19,602,264.65	2024	168.65	−19,602,268.52
	8	3186	478.88	−30,896,240.07	2838	468.60	−30,896,374.35	2284	304.41	−30,896,224.06	2108	301.57	−30,896,357.12
	10	7817	1074.56	−38,230,942.54	3147	532.80	−38,230,966.96	3732	616.46	−38,230,955.54	3492	581.38	−38,230,966.23
300	5	167	22.21	−44,402,684.55	144	19.82	−44,402,720.72	229	30.75	−44,402,699.73	161	21.84	−44,402,712.28
	8	2163	329.84	−70,334,767.66	2488	316.75	−70,334,780.88	1803	311.71	−70,334,748.27	2051	269.50	−70,334,734.55
	10	1983	323.39	−87,330,328.37	1723	289.10	−87,330,377.52	2091	378.74	−87,330,348.57	1752	358.22	−87,330,287.44

Table 6. Performance comparison of LRTR-SR1 variants for ICA on the Stiefel manifold.

n	p	LRTR-SR1-TCG			LRTR-SR1-TCG-LS			LRTR-SR1-TTCG			LRTR-SR1-TTCG-LS		
		Iter	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f	**Iter**	**Time(s)**	f
100	5	2316	42.94	−4,802,627.96	1527	28.18	−4,802,647.74	2167	39.21	−4,802,611.19	2092	37.90	−4,802,642.87
	8	2070	66.68	−7,454,443.47	1216	40.15	−7,454,473.51	2331	75.67	−7,454,462.48	2127	71.49	−7,454,453.67
	10	2147	98.04	−9,128,946.99	1653	66.89	−9,129,013.25	2212	107.66	−9,128,958.82	1884	81.76	−9,128,940.74
200	5	1278	85.99	−19,602,269.23	893	71.34	−19,602,324.37	1560	110.49	−19,602,285.85	1496	113.94	−19,602,280.57
	8	3346	266.27	−30,896,394.23	1812	220.45	−30,896,481.34	3208	488.64	−30,896,357.04	3137	388.29	−30,896,407.13
	10	3258	388.18	−38,230,875.98	1676	196.34	−38,231,042.54	3469	405.92	−38,230,908.51	3029	368.58	−38,230,814.24
300	5	303	35.43	−44,402,670.67	185	22.64	−44,402,713.50	292	34.21	−44,402,703.49	267	33.40	−44,402,705.32
	8	2248	197.49	−70,334,799.50	1226	90.07	−70,334,931.64	2231	216.76	−70,334,751.03	2140	169.32	−70,334,725.65
	10	2373	259.18	−87,330,389.29	1252	131.87	−87,330,608.78	2395	254.64	−87,330,383.50	2230	243.21	−87,330,310.58

5. Conclusions

In this paper, we proposed Riemannian trust-region and limited-memory Riemannian trust-region algorithms equipped with symmetric rank-one updates and an inexact line search strategy for unconstrained optimization problems on Riemannian manifolds. We established $\liminf_{k\to\infty} \|\operatorname{grad} f(x_k)\|_{x_k} = 0$ and revealed the computational advantages of combining the standard TCG solver with the line search mechanism. We also applied our method to instances of Brockett cost function minimization on the Stiefel manifold, Rayleigh quotient optimization on the sphere manifold, and independent component analysis on the Stiefel manifold. In future work, we will use Dolan–Moré performance profiles to compare the algorithms, provide a separate accounting of the cost of the backtracking procedure relative to the savings from avoiding repeated trust-region subproblem solves, analyze theoretically whether the radius update may lead to unnecessarily aggressive contractions, and explore the adaptability of this framework to other complex quotient manifolds, as well as apply the proposed algorithms to large-scale machine learning applications involving orthogonal constraints.

Author Contributions: Conceptualization, J.Y. and T.Y.; methodology, T.Y.; validation, J.Y.; formal analysis, J.Y.; investigation, J.Y.; data curation, J.Y.; writing—original draft preparation, J.Y.; writing—review and editing, T.Y.; supervision, T.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Grant No. 11671205) and the Stability Support Program for Basic Research (Grant No. WZJC202601003).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: This work was supported by the National Natural Science Foundation of China (Grant No. 11671205) and the Stability Support Program for Basic Research (Grant No. WZJC202601003).

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

Abbreviations

The following abbreviations are used in this manuscript:

RTR-SR1-TCG	Riemannian Trust-Region with Symmetric Rank-One and Truncated Conjugate Gradient
RTR-SR1-TCG-LS	Riemannian Trust-Region with Symmetric Rank-One, Truncated Conjugate Gradient and Line Search
RTR-SR1-TTCG	Riemannian Trust-Region with Symmetric Rank-One and Truncated Three-Term Conjugate Gradient
RTR-SR1-TTCG-LS	Riemannian Trust-Region with Symmetric Rank-One, Truncated Three-Term Conjugate Gradient and Line Search
LRTR-SR1-TCG	Limited-memory Riemannian Trust-Region with Symmetric Rank-One and Truncated Conjugate Gradient
LRTR-SR1-TCG-LS	Limited-memory Riemannian Trust-Region with Symmetric Rank-One, Truncated Conjugate Gradient and Line Search
LRTR-SR1-TTCG	Limited-memory Riemannian Trust-Region with Symmetric Rank-One and Truncated Three-Term Conjugate Gradient
LRTR-SR1-TTCG-LS	Limited-memory Riemannian Trust-Region with Symmetric Rank-One, Truncated Three-Term Conjugate Gradient and Line Search

References

1. Boumal, N.; Absil, P.-A. RTRMC: A Riemannian trust-region method for low-rank matrix completion. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 406–414.
2. Vandereycken, B. Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.* **2013**, *23*, 1214–1236. [CrossRef]

3. Bendory, T.; Eldar, Y.C.; Boumal, N. Non-convex phase retrieval from STFT measurements. *IEEE Trans. Inf. Theory* **2017**, *64*, 467–484. [CrossRef]
4. Sun, J.; Qu, Q.; Wright, J. A geometric analysis of phase retrieval. *Found. Comput. Math.* **2018**, *18*, 1131–1198. [CrossRef]
5. Cai, J.-F.; Liu, H.; Wang, Y. Fast rank-one alternating minimization algorithm for phase retrieval. *J. Sci. Comput.* **2019**, *79*, 128–147. [CrossRef]
6. Boumal, N. Nonconvex phase synchronization. *SIAM J. Optim.* **2016**, *26*, 2355–2377. [CrossRef]
7. Liu, H.; Yue, M.-C.; Man-Cho So, A. On the estimation performance and convergence rate of the generalized power method for phase synchronization. *SIAM J. Optim.* **2017**, *27*, 2426–2446. [CrossRef]
8. Huang, W.; Hand, P. Blind deconvolution by a steepest descent algorithm on a quotient manifold. *SIAM J. Imaging Sci.* **2018**, *11*, 2757–2785. [CrossRef]
9. Cherian, A.; Sra, S. Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2859–2871. [CrossRef] [PubMed]
10. Sun, J.; Qu, Q.; Wright, J. Complete dictionary recovery over the sphere I: Overview and the geometric picture. *IEEE Trans. Inf. Theory* **2016**, *63*, 853–884. [CrossRef]
11. Absil, P.-A.; Mahony, R.; Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*; Princeton University Press: Princeton, NJ, USA, 2008.
12. Ring, W.; Wirth, B. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM J. Optim.* **2012**, *22*, 596–627. [CrossRef]
13. Sato, H. A Dai–Yuan-type Riemannian conjugate gradient method with the weak Wolfe conditions. *Comput. Optim. Appl.* **2016**, *64*, 101–118. [CrossRef]
14. Sato, H.; Iwai, T. A new, globally convergent Riemannian conjugate gradient method. *Optimization* **2015**, *64*, 1011–1031. [CrossRef]
15. Huang, W. Optimization Algorithms on Riemannian Manifolds with Applications. Ph.D. Thesis, The Florida State University, Tallahassee, FL, USA, 2013.
16. Huang, W.; Absil, P.-A.; Gallivan, K.A. A Riemannian BFGS method without differentiated retraction for nonconvex optimization problems. *SIAM J. Optim.* **2018**, *28*, 470–495. [CrossRef]
17. Absil, P.-A.; Baker, C.G.; Gallivan, K.A. Trust-region methods on Riemannian manifolds. *Found. Comput. Math.* **2007**, *7*, 303–330. [CrossRef]
18. Huang, W.; Absil, P.-A.; Gallivan, K.A. A Riemannian symmetric rank-one trust-region method. *Math. Program.* **2015**, *150*, 179–216. [CrossRef]
19. Wang, B.; Ma, S.; Xue, L. Riemannian stochastic proximal gradient methods for nonsmooth optimization over the Stiefel manifold. *J. Mach. Learn. Res.* **2022**, *23*, 1–33.
20. Bonnabel, S. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Autom. Control* **2013**, *58*, 2217–2229. [CrossRef]
21. Tripuraneni, N.; Flammarion, N.; Bach, F.; Jordan, M.I. Averaging stochastic gradient descent on Riemannian manifolds. In *Proceedings of the Conference On Learning Theory*; PMLR: Cambridge, MA, USA, 2018; pp. 650–687.
22. Chen, S.; Ma, S.; Man-Cho So, A.; Zhang, T. Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM J. Optim.* **2020**, *30*, 210–239. [CrossRef]
23. Gao, B.; Liu, X.; Yuan, Y.-x. Parallelizable algorithms for optimization problems with orthogonality constraints. *SIAM J. Sci. Comput.* **2019**, *41*, A1949–A1983. [CrossRef]
24. Grohs, P.; Hosseini, S. ε-subgradient algorithms for locally Lipschitz functions on Riemannian manifolds. *Adv. Comput. Math.* **2016**, *42*, 333–360. [CrossRef]
25. Hu, J.; Milzarek, A.; Wen, Z.; Yuan, Y. Adaptive regularized Newton method for Riemannian optimization. *arXiv* **2017**, arXiv:1708.02016.
26. Nocedal, J.; Yuan, Y. Combining Trust Region and Line Search Techniques. In *Proceedings of the Advances in Nonlinear Programming*; Springer: Boston, MA, USA, 1998; pp. 153–175.
27. Sheng, Z.; Gonglin, Y.; Zengru, C. A new adaptive trust region algorithm for optimization problems. *Acta Math. Sci.* **2018**, *38*, 479–496. [CrossRef]
28. Zhang, X.; Zhang, J.; Liao, L. An adaptive trust region method and its convergence. *Sci. China Ser. A Math.* **2002**, *45*, 620–631. [CrossRef]
29. Kimiaei, M.; Ghaderi, S. A new restarting adaptive trust-region method for unconstrained optimization. *J. Oper. Res. Soc. China* **2017**, *5*, 487–507. [CrossRef]
30. Mor, U.; Shustin, B.; Avron, H. Solving trust region subproblems using Riemannian optimization. *Numer. Math.* **2023**, *154*, 1–33. [CrossRef]
31. Boumal, N. *An Introduction to Optimization on Smooth Manifolds*; Cambridge University Press: Cambridge, UK, 2023.
32. Chavel, I. *Riemannian Geometry*; Cambridge Studies in Advanced Mathematics Vol. 98; Cambridge University Press: Cambridge, UK, 2006.

33. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer: New York, NY, USA, 2006.
34. Steihaug, T. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.* **1983**, *20*, 626–637. [CrossRef]
35. Byrd, R.H.; Nocedal, J.; Schnabel, R.B. Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Program.* **1994**, *63*, 129–156. [CrossRef]
36. Zhao, S.; Yan, T.; Wang, K.; Zhu, Y. Adaptive Trust-Region Method on Riemannian Manifold. *J. Sci. Comput.* **2023**, *96*, 67. [CrossRef]
37. Theis, F.J.; Cason, T.P.; Absil, P.-A. Soft dimension reduction for ICA by joint diagonalization on the Stiefel manifold. In *Proceedings of the International Conference on Independent Component Analysis and Signal Separation*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 354–361.

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