Journal of Applied Analysis and Computation
Volume 16, Number 1, February 2026, 209–228

Website:http://www.jaac-online.com
DOI:10.11948/20240564

# A NOVEL RIEMANNIAN CONJUGATE GRADIENT METHOD WITH ITERATION COMPLEXITY GUARANTEES*

**Juan Zhou[1,2], Kangkang Deng[1], Hongxia Wang[1,†] and Zheng Peng[2]**

**Abstract** Conjugate gradient methods are important first-order optimization algorithms both in Euclidean spaces and on Riemannian manifolds. However, while various types of Riemannian conjugate gradient methods have been studied, the iteration complexity analysis remains unknown. This paper proposes a novel Riemannian conjugate gradient method for nonconvex problems with an iteration complexity guarantee. In particular, we introduce a novel restart condition, leading to a function decrease at each iteration. Our method converges to an $\epsilon$-stationary point with an iteration complexity of $\mathcal{O}(\epsilon^{-2})$. To the best of our knowledge, this is the first Riemannian conjugate gradient method with iteration complexity guarantees. Numerical experiments on Rayleigh-quotient and Brockett-cost-function minimization problems demonstrate the efficiency and practical applicability of the proposed method.

**Keywords** Iteration complexity, restart condition, Riemannian conjugate gradient methods, Riemannian optimization.

**MSC(2010)** 65K05, 58C05, 90C30.

## 1. Introduction

This paper focuses on the following manifold optimization problem

$$\min_{x\in\mathcal{M}} f(x), \tag{1.1}$$

where $\mathcal{M}$ is a Riemannian manifold, $f : \mathcal{M} \to \mathbb{R}$ is a continuously differentiable and nonconvex function. Given a tolerance $\epsilon \in (0, 1)$, our goal is to compute an $\epsilon$-approximate stationary point, that is, a vector $x \in \mathcal{M}$ satisfies

$$\|\mathrm{grad} f(x)\|_x \leq \epsilon, \tag{1.2}$$

where $\mathrm{grad} f$ is the Riemannian gradient of $f$ that will be defined in the next section, $\|\cdot\|_x$ denotes the Riemannian norm. Manifold optimization has recently drawn a lot of attention because of its success in a variety of important applications, including low-rank tensor completion [38], inverse eigenvalue problems [43], distributed learning [10, 11, 22], etc. A framework of Riemannian optimization has two stages in each iteration: (i) Find a tangent vector as the search direction

[†]The corresponding author.
[1]Department of Mathematics, National University of Defense Technology, Changsha 410073, China
[2]School of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, China
[*]The authors were supported by the National Key Research and Development Program of China (2020YFA0713504) and the National Natural Science Foundation of China (12471401 and 12401419).
Email: juanzhou425@gmail.com(J. Zhou), freedeng1208@gmail.com(K. Deng), wanghongxia@nudt.edu.cn(H. Wang), pzheng@xtu.edu.cn(Z. Peng)

or the trial step; (ii) Invoke a retraction that maps a tangent vector to a point on the manifold. Many classical optimization methods for smooth optimization problems in Euclidean space have been successfully generalized to the problems on Riemannian manifolds [3,5,12,20]. The reader is referred to [4,19,33] for a comprehensive review.

For conjugate gradient (CG) methods on Riemannian manifolds to solve problem (1.1), we call them Riemannian conjugate gradient (RCG) methods. Compared to gradient-type methods, RCG methods demonstrate a faster convergence rate for nonconvex smooth problems. Furthermore, they are more suitable for efficiently addressing large-scale problems compared to second-order methods like the Newton method and the quasi-Newton method. Several types of RCG methods have been proposed recently [32,35,40,44,45], validating the numerical efficiency of RCG methods across diverse applications. It should be noted that current RCG methods only have asymptotic convergence analysis, such as global convergence [34,36] and superlinear convergence rate [37]. They fail to obtain non-asymptotic convergence results, i.e., the iteration complexity analysis remains unknown, despite their numerical superiority over gradient-type methods [46].

In Euclidean space, the restart condition is widely employed in CG methods to enhance computational efficiency. When the CG direction satisfies the restart condition (indicating that it is not a descent direction), it is substituted with the gradient direction. Recent research [8,21,25] demonstrates that CG methods with a well-defined restart condition can derive iteration complexity results. These restart conditions typically encompass a descent condition and a bounded angle condition. Despite the extensive study of restart conditions in Euclidean space, RCG methods with restart conditions in Riemannian manifolds remain relatively unexplored. Inspired by the iterative complexity analysis process of the CG method in Euclidean space, this paper conducts a complexity analysis of the algorithm based on a novel restart condition of the RCG.

### 1.1. Contributions

This paper aims to design a restart condition and integrate it into an RCG method to establish the iteration complexity results. The main contributions are outlined as follows:

- We propose an RCG method with a novel restart condition. This condition allows us to establish the iteration complexity result. Moreover, to enhance the performance in practical implementations, we utilize the Riemannian Barzilai-Borwein (BB) step-size along with a non-monotone line-search strategy.
- To obtain the iteration complexity of our method, we first provide a crucial descent lemma, which ensures a decrease in function value at every iteration. Consequently, we show that the proposed algorithm finds an $\epsilon$-stationary point of (1.1) with the iteration complexity result of $\mathcal{O}(\epsilon^{-2})$.
- Furthermore, the empirical validation of the proposed RCG methods through numerical experiments involving the Rayleigh-quotient minimization problem and the Brockett-cost-function minimization problem underscores their efficiency and practical applicability.

### 1.2. Related works

The RCG method analyzes iteration complexity by constructing restart conditions, which is generalized from the CG method in Euclidean space. In [37], Smith proposed using the exponential map and parallel transport to generalize the optimization methods from Euclidean space to a Riemannian manifold. Absil et al. [1] proposed the utilization of retraction to approximate the exponential retraction and vector transport to approximate parallel transport. Although the theoretical foundation of RCG methods has been established, further development is ongoing. Recent research on the RCG method has employed the inverse retraction [44]. RCG methods have been applied to the unit sphere with respect to $p$-norms, where $p > 1$ [35], the symplectic Stiefel manifold [40], and the Grassmann manifolds [45]. These studies investigate the geometry of various manifolds and employ several geometric tools for Riemannian optimization. Furthermore, a novel general framework that unifies existing RCG methods has been proposed by Sato [32, 34]. Extensive analysis is conducted on the global convergence properties of several specific types of algorithms.

To ensure the global convergence of RCG methods, both a line-search strategy and a descent direction are necessary. Similar to the line-search method in Euclidean space, the step-size can be obtained through a curvilinear search on the manifold, such as the Armijo non-monotone search proposed by Sachs and Sachs [30]. Moreover, the search direction needs to satisfy the descent condition under certain line-search rules. In particular, each iteration needs to satisfy the sufficient descent condition [31]. However, in CG methods, the search direction is not necessarily a descent direction in every iteration. Instead, a proper restart strategy [29](restarting means discarding the previous search direction and choosing a new one) is critical in the CG approach. For example, Chan-Renous-Legoubin and Royer [21] proposed a CG method based on a simple line-search paradigm and a modified restart condition. These two ingredients allow for monitoring the descent properties of the search direction and obtaining the iterative complexity results of the proposed method.

In addition to global convergence, iteration complexity (i.e., the number of iterations, and evaluations of $f$ and its Riemannian gradient when the algorithm reaches an approximate criticality satisfying (1.2)) is related to the actual efficiency of the algorithm and is also a significant area of research focus [23, 24]. Such iteration complexity bounds are standard in Euclidean optimization. Around the same time, they have been generalized to Riemannian optimization algorithms. In particular, Zhang and Sra [42] give the iteration complexity of deterministic and stochastic (sub)gradient methods for optimizing smooth and nonsmooth $g$-convex functions. Bento et al. [2] analyze the iteration-complexity of gradient, subgradient and proximal point methods in the Riemannian setting. Boumal et al. [6] consider the nonconvex problem on the Riemannian manifold, and give the iteration complexity of Riemannian gradient descent and Riemannian trust-region methods. Therefore, currently, only the iterative complexity of the gradient descent method on manifolds has obtained theoretical results. Providing a complexity analysis of Riemannian gradient-type methods remains a challenging endeavor. To our knowledge, no results have been published for the RCG method. In this paper, we concentrate on the iteration complexity analysis for the RCG method.

## 1.3. Organization

The paper is organized as follows. Preliminaries on RCG methods are given in Section 2. In Section 3, we describe our framework based on a modified restart condition. Complexity results for this framework are obtained and discussed in Section 4. In Section 5, we then conduct a numerical study of our proposed method involving two Riemannian optimization problems. Conclusions are made in the last Section 6.

## 2. Preliminaries

### 2.1. Riemannian optimization

An $n$-dimensional smooth manifold $\mathcal{M}$ is defined as an $n$-dimensional topological manifold equipped with a smooth structure, ensuring that every point has a neighborhood that can be diffeomorphic to an $n$-dimensional Euclidean space. Intuitively, the tangent space $T_x\mathcal{M}$ at a point $x$ on the manifold $\mathcal{M}$ represents the collection of tangent vectors associated with all curves passing through $x$. Mathematically,

**Definition 2.1** (Tangent vector [1])**.** A tangent vector $\xi_x$ to a manifold $\mathcal{M}$ at a point $x$ is a mapping from $\wp_x\mathcal{M}$ to $\mathbb{R}$ such that there exists a curve $\gamma$ on $\mathcal{M}$ with $\gamma(0) = x$, satisfying

$$\xi_x u := \dot{\gamma}(0)u = \left.\frac{d(u(\gamma(t)))}{dt}\right|_{t=0}, \forall u \in \wp_x\mathcal{M},$$

where $\wp_x\mathcal{M}$ denotes the set of smooth real-valued functions $f$ defined in a neighborhood of $x$ in $\mathcal{M}$.

The Riemannian manifold $\mathcal{M}$ is endowed with a smoothly varying inner product $\langle\cdot,\cdot\rangle_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ on the tangent space $T_x\mathcal{M}$ at each $x \in \mathcal{M}$. Additionally, the norm of $\xi_x \in T_x\mathcal{M}$ is defined as $\|\xi_x\|_x := \sqrt{\langle\xi_x, \xi_x\rangle_x}$. The Riemannian gradient $\mathrm{grad}f(x) \in T_x\mathcal{M}$ of a function $f$ at point $x$ is the unique tangent vector satisfying

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

where $Df(x)[\xi]$ represents the derivative of $f(\gamma(t))$ at $t = 0$, and $\gamma(t)$ denotes any curve on the manifold satisfying $\gamma(0) = x$ and $\dot{\gamma}(0) = \xi$. If $\mathcal{M}$ is a compact Riemannian manifold embedded in Euclidean space, it follows that $\mathrm{grad}f(x) = \mathcal{P}_{T_x\mathcal{M}}(\nabla f(x))$, where $\nabla f(x)$ is the Euclidean gradient, $\mathcal{P}_{T_x\mathcal{M}}$ is the projection operator onto the tangent space $T_x\mathcal{M}$.

**Definition 2.2** (Retraction [1])**.** A smooth mapping $R : T\mathcal{M} \to \mathcal{M}$ is called a retraction on a manifold $\mathcal{M}$ if its restriction at $x$, denoted as $R_x : T_x\mathcal{M} \to \mathcal{M}$, satisfies

- $R_x(0_x) = x$, where $0_x$ denotes the zero element of $T_x\mathcal{M}$.
- With the canonical identification $T_{0_x}T_x\mathcal{M} \simeq T_x\mathcal{M}$, $R_x$ satisfies

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

  where $\mathrm{id}_{T_x\mathcal{M}}$ denotes the identity mapping on $T_x\mathcal{M}$.

The retraction operator $\mathcal{R}$ plays a crucial role in manifold optimization by efficiently pulling points $x \in \mathcal{M}$ back from the tangent space $T_x\mathcal{M}$ onto the manifold $\mathcal{M}$ [19]. In addition, another important concept in Riemannian manifolds is the vector transport operator $\mathcal{T}$.

**Definition 2.3** (Vector transport [1])**.** A smooth mapping

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

is called a vector transport on a manifold $\mathcal{M}$, where $\oplus$ denotes the Whitney sum

$$T\mathcal{M} \oplus T\mathcal{M} = \{(\xi_x, \eta_x) : \xi_x, \eta_x \in T_x\mathcal{M}, x \in \mathcal{M}\}.$$

if it satisfies

- There exists a retraction $R$, known as the retraction associated with $\mathcal{T}$, such that $\mathcal{T}_\eta(\xi) \in T_{R_x(\eta)}\mathcal{M}$ for all $x \in \mathcal{M}$, and for all $\eta, \xi \in T_x\mathcal{M}$.
- $\mathcal{T}_{0_x}\xi = \xi$ for all $\xi \in T_x\mathcal{M}$.
- $\mathcal{T}_\eta(a\xi + b\zeta) = a\mathcal{T}_\eta(\xi) + b\mathcal{T}_\eta(\zeta)$ for all $a, b \in \mathbb{R}$, and for all $\eta, \xi, \zeta \in T_x\mathcal{M}$.

## 2.2. Riemannian conjugate gradient method

In this subsection, we provide a review of the RCG method for problem (1.1). The RCG techniques are iterative optimization schemes formulated as follows

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k), \tag{2.1}$$

where $R$ is the retraction operator on manifold $\mathcal{M}$, the search direction $\eta_k \in T_x\mathcal{M}$ is considered a descent direction if $\langle \text{grad} f(x_k), \eta_k \rangle_{x_k} < 0$ holds, and $\alpha_k > 0$ denotes a step-size typically determined via a line-search. Many studies concentrate on analyzing monotone line-search strategies [26], ensuring no increase in the values of the objective function in successive iterations.

In this paper, we consider non-monotone step-size rules [15]. One of the most widely used rules for selecting $\alpha_k$ is the non-monotone line-search condition proposed by Zhang and Hager [41], which aims to find the smallest non-negative integer $j_k$ such that

$$f(R_{x_k}(\alpha_k \eta_k)) \leq C_k + \rho \tau_k \theta^{j_k} \langle g_k, \eta_k \rangle_{x_k}, \tag{2.2}$$

for all $k \geq 0$, $\rho, \theta \in (0, 1)$, where $C_k$ is the convex combination between $C_{k-1}$ and $f(x_k)$, defined as $C_k = \varphi_k C_{k-1} + (1 - \varphi_k) f(x_k)$, with $\varphi_k \in [0, 1]$ for all $k$ ensuring convergence. Another widely used non-monotone condition in the literature is Grippo's non-monotonic strategy [16], which ensures that

$$f(R_{x_k}(\alpha_k \eta_k)) \leq \max_{0 \leq i \leq m(k)} [f(x_{k-i})] + \rho \tau_k \theta^{j_k} \langle g_k, \eta_k \rangle_{x_k}, \tag{2.3}$$

holds for all $k \geq 0$, where $m(0) = 0$ and $0 \leq m(k) \leq \min\{m(k-1) + 1, N\}$ for $N \in \mathbb{N}$ fixed.

The RCG direction involves combining local information (i.e. the negative gradient at the current point) with the previous direction, which can be defined as follows

$$\begin{cases} \eta_0 = -\text{grad} f(x_0), \\ \eta_{k+1} = -\text{grad} f(x_{k+1}) + \beta_{k+1} \mathcal{T}_{\alpha_k \eta_k}(\eta_k), \end{cases} \tag{2.4}$$

where $\beta_{k+1}$ is often called the conjugate parameter, $\mathcal{T}$ is the vector transport defined in Definition 2.3. A commonly used vector transport is the differentiated retraction with $R$

$$\mathcal{T}^R_\eta(\xi) = DR_x(\eta)[\xi] = \frac{\mathrm{d}}{\mathrm{d}t} R_x(\eta + t\xi)\bigg|_{t=0},$$

and consequently, we have

$$\mathcal{T}^R_{\alpha_k \eta_k}(\eta_k) = DR_{x_k}(\alpha_k \eta_k)[\eta_k] = \frac{\mathrm{d}}{\mathrm{d}t} R_{x_k}(t_k \eta_k)\bigg|_{t=\alpha_k}. \tag{2.5}$$

In (2.4), $\beta_{k+1}$ is given by generalizations of the formulas in Euclidean space, including the Fletcher–Reeves (FR) [14], Dai–Yuan (DY) [9], Polak–Ribière–Polyak (PRP) [27,28], Hestenes–Stiefel (HS) [18], and Hager–Zhang (HZ) [17] formulas

$$\beta^{FR}_{k+1} = \frac{\|g_{k+1}\|^2_{x_{k+1}}}{\|g_k\|^2_{x_k}}, \qquad \beta^{DY}_{k+1} = \frac{\|g_{k+1}\|^2_{x_{k+1}}}{\left\langle g_{k+1}, \mathcal{T}^R_{\alpha_k \eta_k}(\eta_k) \right\rangle_{x_{k+1}} - \langle g_k, \eta_k \rangle_{x_k}},$$

$$\beta_{k+1}^{PRP} = \frac{\langle g_{k+1}, y_{k+1}\rangle_{x_{k+1}}}{\|g_k\|_{x_k}^2}, \qquad \beta_{k+1}^{HS} = \frac{\langle g_{k+1}, y_{k+1}\rangle_{x_{k+1}}}{\left\langle g_{k+1}, \mathcal{T}_{\alpha_k\eta_k}^R(\eta_k)\right\rangle_{x_{k+1}} - \langle g_k, \eta_k\rangle_{x_k}},$$

$$\beta_{k+1}^{HZ} = \beta_{k+1}^{HS} - \mu \frac{\|y_{k+1}\|_{x_{k+1}}^2 \left\langle g_{k+1}, \mathcal{T}_{\alpha_k\eta_k}^R(\eta_k)\right\rangle_{x_{k+1}}}{\left(\left\langle g_{k+1}, \mathcal{T}_{\alpha_k\eta_k}^R(\eta_k)\right\rangle_{x_{k+1}} - \langle g_k, \eta_k\rangle_{x_k}\right)^2}, \tag{2.6}$$

where $y_{k+1} := g_{k+1} - \mathcal{T}_{\alpha_k\eta_k}^R(g_k)$. The choice of the formula for the parameter $\beta_{k+1}$ gives rise to various RCG methods.

It is important to note that regardless of the chosen variant, an RCG method may produce iterates where $\langle \text{grad} f\,(x_{k+1})\,, \eta_{k+1}\rangle\, x_{k+1} \geq 0$, indicating no guarantee for a decrease in the direction $\eta_{k+1}$. A common approach to address this issue is to redefine the search direction as $\eta_{k+1} = -\text{grad} f\,(x_{k+1})$, a process known as restarting. Nevertheless, to the best of our knowledge, there is no literature discussing the iterative complexity of the RCG approach. In the next section, we propose an alternative to this restart condition that provides complexity results for the RCG method.

## 3. Riemannian conjugate gradient method with modified restart condition

In this section, we describe an RCG method with a modified restart condition. In our framework, we build on the restarting idea by monitoring the value of $\langle \text{grad} f\,(x_k)\,, \eta_k\rangle_{x_k}$ and that of $\|\eta_k\|_{x_k}$. The detailed algorithm is described in Algorithm 1.

At every iteration, we perform an Armijo non-monotone line-search (3.1) to compute a step-size that yields a suitable decrease in the objective function, where the initial step is set to be the BB step-size given as follows

$$\bar{\tau}_k := \frac{\langle s_{k-1}, s_{k-1}\rangle_{x_k}}{|\,\langle s_{k-1}, y_{k-1}\rangle_{x_k}\,|}, \tag{3.5}$$

where $s_{k-1} = \alpha_{k-1} \cdot \mathcal{T}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1})\,, \quad y_{k-1} = \text{grad}\, f\,(x_k) + \alpha_{k-1}^{-1} \cdot s_{k-1}$ and $\mathcal{T}_{\alpha_{k-1}\eta_{k-1}} : T_{x_{k-1}}\mathcal{M} \mapsto T_{x_k}\mathcal{M}$. In addition, $v_k$ is the term that controls the degree of non-monotonicity of the proposed Algorithm 1. One of the core ideas of non-monotone rules is to allow the iterates to escape from local minima and increase the probability of finding a global minimum. This term depends on $k$, which means that we can adjust it at each iteration. Notice that by selecting $v_k = C_k - f(x_k)$ in Algorithm 1, we recover the Zhang and Hager's non-monotone rule (2.2). Similarly, choosing $v_k = \max\limits_{0\leq i\leq m(k)} [f\,(x_{k-i})] - f(x_k)$, we obtain the Grippo's non-monotone condition (2.3). The same argument applies to the rest of the non-monotone strategies.

Once the new point has been computed, we evaluate the gradient at the next iterate, as well as the parameter $\beta_{k+1}$, which is typically chosen from one of the formulas (2.6). Both the gradient and the parameter are then used to define the new search direction $\eta_{k+1}$, and we know that it is crucial for the RCG direction $\eta_{k+1}$ to satisfy the restart conditions for complexity analysis. Therefore, the key ingredient to Algorithm 1 is the modified restart condition (3.4), which determines whether the RCG direction is kept for the next iteration. If iteration $k$ does not end with a restart for any $k \geq 0$, we have

$$\langle g_{k+1}, \eta_{k+1}\rangle_{x_{k+1}} \leq -\sigma\, \|g_{k+1}\|_{x_{k+1}}^{1+p} \;\text{ and }\; \|\eta_{k+1}\|_{x_{k+1}} \leq \kappa\, \|g_{k+1}\|_{x_{k+1}}^{q}\,. \tag{3.6}$$

**Algorithm 1** RCG with modified restart condition.

```
Require: $x_0 \in \mathcal{M}, \rho \in (0,1), \sigma \in (0,1], \kappa \geq 1, p, q \geq 0$.
 1: Set $\eta_0 = -g_0 := -\mathrm{grad} f(x_0)$.
 2: for $k = 0, 1, 2, \cdots$ do
 3:     Compute the initial step-size $\tau_k = \max\{\tau_{\min}, \min\{\tau_{\max}, \bar{\tau}_k\}\}$, where $\bar{\tau}_k$ is defined by (3.5)
        and $\tau_{\min} < \tau_{\max}$.
 4:     Choose the non-monotone term $v_k \geq 0$.
 5:     Find the smallest non-negative integer $j_k$ such that
            $$f(R_{x_k}(\tau_k \theta^{j_k} \eta_k)) < [f(x_k) + v_k] + \rho \tau_k \theta^{j_k} \langle g_k, \eta_k \rangle_{x_k}. \quad (3.1)$$
 6:     Set $\alpha_k = \tau_k \theta^{j_k}$ and compute the next point
            $$x_{k+1} = R_{x_k}(\alpha_k \eta_k). \quad (3.2)$$
 7:     Set $g_{k+1} = \mathrm{grad} f(x_{k+1})$ and choose a parameter $\beta_{k+1}$ from (2.6).
 8:     Compute the conjugate direction $\eta_{k+1}$ as follows
            $$\eta_{k+1} = -g_{k+1} + \beta_{k+1} \mathcal{T}_{\alpha_k \eta_k}(\eta_k). \quad (3.3)$$
 9:     If the restart condition
            $$\langle g_{k+1}, \eta_{k+1} \rangle_{x_{k+1}} \geq -\sigma \|g_{k+1}\|_{x_{k+1}}^{1+p} \text{ or } \|\eta_{k+1}\|_{x_{k+1}} \geq \kappa \|g_{k+1}\|_{x_{k+1}}^{q} \quad (3.4)$$
        holds, restart the algorithm by setting $\eta_{k+1} = -g_{k+1}$.
10: end for
```

This means that the RCG direction satisfies both a descent condition and a bounded angle condition. Note that (3.6) can be viewed as a generalization of the following condition (obtained for $p = q = 1$ ):

$$\langle g_{k+1}, \eta_{k+1} \rangle \leq -\sigma \|g_{k+1}\|^2 \text{ and } \|\eta_{k+1}\| \leq \kappa \|g_{k+1}\|.$$

This condition is typical of gradient-related directions and has been instrumental in obtaining complexity guarantees for gradient-type methods [7]. Our algorithm has been endowed with additional parameters $p$ and $q$, which enhance its flexibility and adaptability to the specified problem.

## 4. Complexity analysis

In this section, we derive a complexity result for our restarted variant of the RCG method. Section 4.1 provides the necessary assumptions as well as intermediate results, while Section 4.2 establishes and discusses complexity bounds on the number of iterations necessary for our Algorithm 1.

### 4.1. Decrease lemma

We make the following assumptions regarding the objective function of problem (1.1).

**Assumption 4.1.** *The objective function $f : \mathcal{M} \to \mathbb{R}$ is continuously differentiable and its gradient* grad$f$ *is $L$-Lipschitz continuous for $L > 0$.*

**Assumption 4.2.** *There exists $f_{low} : \mathcal{M} \to \mathbb{R}$ such that $f(x) \geq f_{low}$ for every $x \in \mathcal{M}$.*

Denote

$$\begin{aligned} &\mathcal{N} = \left\{ k \in \mathbb{N} \mid \langle g_k, \eta_k \rangle_{x_k} \leq -\sigma \|g_k\|_{x_k}^{1+p} \quad \text{and} \quad \|\eta_k\|_{x_k} \leq \kappa \|g_k\|_{x_k}^{q} \right\}, \\ &\mathcal{R} = \mathbb{N} \setminus \mathcal{N}. \end{aligned} \tag{4.1}$$

For $k \geq 1$, our algorithm explicitly checks whether $k \in \mathcal{N}$. If $k \in \mathcal{R}$, the restarting process is triggered, and the search direction becomes the negative gradient. Hence, we classify $k \in \mathcal{R}$ as the index of a restarted iteration, while $k \in \mathcal{N}$ represents a non-restarted iteration. Depending on the nature of each iteration, we can establish an upper bound on the number of backtracking steps required to compute a suitable step-size. Let us first focus on the non-restarted iterations, as their proof encompasses that of the restarted iterations.

**Lemma 4.1.** *Let Assumption 4.1 hold, and let $k \in \mathcal{N}$ such that $\|g_k\|_{x_k} > 0$. Then, the line-search process* (3.1) *terminates after at most $\lfloor \bar{j}_{\mathcal{N},k} + 1 \rfloor$ iterations, where*

$$\bar{j}_{\mathcal{N},k} := \left[ \log_\theta \left( \frac{2(1-\rho)\sigma}{\kappa^2 L \tau_{\max}} \right) \|g_k\|_{x_k}^{1+p-2q} \right]_+ . \tag{4.2}$$

*Moreover, the resulting decrease at the $k$-th iteration satisfies*

$$f(x_k) - f(R_{x_k}(\alpha_k \eta_k)) > c_{\mathcal{N}} \min \left\{ \|g_k\|_{x_k}^{1+p}, \|g_k\|_{x_k}^{2(1+p-q)} \right\} - v_k, \tag{4.3}$$

*where*

$$c_{\mathcal{N}} := \rho\sigma \min \left\{ \tau_{\min}, \frac{2(1-\rho)\tau_{\min}\sigma\theta}{\kappa^2 L \tau_{\max}} \right\}. \tag{4.4}$$

**Proof.** Since $R$ is a retraction, we have

$$\begin{aligned} D(f \circ R_{x_k})(0_{x_k})[\eta_k] &= \frac{\mathrm{d}}{\mathrm{d}t} f(R_{x_k}(t\eta_k)) \Big|_{t=0} \\ &= Df(x_k)[DR_{x_k}(0_{x_k})[\eta_k]] \\ &= Df(x_k)[\eta_k] \\ &= \langle g_k, \eta_k \rangle_{x_k}. \end{aligned}$$

It follows from Taylor's theorem of $f$ at $x_k$ that there exists a constant $L$ such that

$$\begin{aligned} f(R_{x_k}(\alpha_k\eta_k)) - f(x_k) &\leq f(R_{x_k}(\alpha_k\eta_k)) - f(R_{x_k}(0_{x_k})) \\ &= \alpha_k \langle g_k, \eta_k \rangle_{x_k} + \int_0^{\alpha_k} (\mathrm{D}(f \circ R_{x_k})(t\eta_k)[\eta_k] - \mathrm{D}(f \circ R_{x_k})(0_{x_k})[\eta_k]) \mathrm{d}t \\ &\leq \alpha_k \langle g_k, \eta_k \rangle_{x_k} + \int_0^{\alpha_k} |\mathrm{D}(f \circ R_{x_k})(t\eta_k)[\eta_k] - \mathrm{D}(f \circ R_{x_k})(0_{x_k})[\eta_k]| \mathrm{d}t \\ &\leq \alpha_k \langle g_k, \eta_k \rangle_{x_k} + \frac{1}{2} L \alpha_k^2 \|\eta_k\|_{x_k}^2 . \end{aligned} \tag{4.5}$$

If the decrease condition (3.1) holds for $\alpha_k = \tau_k$, then the bound (4.2) holds. Moreover, combining (3.1) with the definition of $\mathcal{N}$ in (4.1) gives

$$v_k + f(x_k) - f(R_{x_k}(\alpha_k \eta_k)) > -\rho\tau_k \langle g_k, \eta_k\rangle_{x_k} \geq \rho\sigma\tau_{\min} \|g_k\|_{x_k}^{1+p},$$

hence (4.3) also holds. Now suppose that the line-search condition (3.1) fails for some $\alpha_k = \tau_k\theta^{j_k}$ with $j \in \mathbb{N}$. It follows from (4.5) that

$$\begin{aligned} v_k + \rho\alpha_k \langle g_k, \eta_k\rangle_{x_k} \leq & f(R_{x_k}(\alpha_k\eta_k)) - f(x_k) \\ \leq & \alpha_k \langle g_k, \eta_k\rangle_{x_k} + \frac{1}{2}L\alpha_k^2 \|\eta_k\|_{x_k}^2 \\ \leq & \alpha_k \langle g_k, \eta_k\rangle_{x_k} + \frac{\kappa^2 L}{2}\alpha_k^2 \|g_k\|_{x_k}^{2q}, \end{aligned}$$

where the last inequality comes from (4.1). Then we have that

$$\frac{\kappa^2 L}{2}\alpha_k^2 \|g_k\|_{x_k}^{2q} \geq v_k - (1-\rho)\alpha_k \langle g_k, \eta_k\rangle_{x_k} \geq v_k + (1-\rho)\alpha_k\sigma\|g_k\|_{x_k}^{1+p}, \tag{4.6}$$

where the second inequality is due to that $k \in \mathcal{N}$. This implies that

$$\alpha_k \geq \frac{2v_k}{\kappa^2 L\alpha_k}\|g_k\|_{x_k}^{-2q} + \frac{2(1-\rho)\sigma}{\kappa^2 L}\|g_k\|_{x_k}^{1+p-2q} \geq \frac{2(1-\rho)\sigma}{\kappa^2 L}\|g_k\|_{x_k}^{1+p-2q}.$$

Since $\alpha_k = \tau_k\theta^{j_k} \leq \tau_{\max}\theta^{j_k}$, it follows from (4.6) that $j_k \leq \bar{j}_{\mathcal{N},k}$. As a result, the line-search process must terminate after $j_k \leq \lfloor \bar{j}_{\mathcal{N},k} + 1 \rfloor$ iterations, where $\bar{j}_{\mathcal{N}}$ is defined in (4.2). Moreover, since the line-search does not terminate after $j_k - 1$ iterations, we have

$$\theta^{j_k - 1} \geq \frac{2(1-\rho)\sigma}{\kappa^2 L\tau_{\max}} \|g_k\|_{x_k}^{1+p-2q},$$

which is equivalent to

$$\alpha_k = \tau_k\theta^{j_k} \geq \frac{2(1-\rho)\tau_{\min}\theta\sigma}{\kappa^2 L\tau_{\max}} \|g_k\|_{x_k}^{1+p-2q}.$$

Consequently, the function decreases at iteration $k$ satisfies

$$\begin{aligned} v_k + f(x_k) - f(R_{x_k}(\alpha_k\eta_k)) > & -\rho\alpha_k \langle g_k, \eta_k\rangle_{x_k} \\ \geq & \frac{2\rho(1-\rho)\tau_{\min}\theta\sigma^2}{\kappa^2 L\tau_{\max}} \|g_k\|_{x_k}^{2(1+p-q)} \\ \geq & c_{\mathcal{N}} \|g_k\|_{x_k}^{2(1+p-q)}. \end{aligned}$$

Hence (4.3) also holds in this case, where $c_{\mathcal{N}}$ is defined as (4.4). □

We now consider the restarted iterations. In that case, the search direction satisfies a property analogous to that defining non-restarted iterations in (4.1) with $\sigma = \kappa = 1$ and $p = q = 1$. A reasoning identical to that used in the proof of Lemma 4.1 leads to the following result.

**Lemma 4.2.** *Let Assumption 4.1 hold, and let $k \in \mathcal{R}$ such that $\|g_k\|_{x_k} > 0$. Then, the line-search process terminates after at most $\bar{j}_{\mathcal{R}} + 1$ iterations, where*

$$\bar{j}_{\mathcal{R}} = \left[\log_\theta\left(\frac{2(1-\rho)}{L\tau_{\max}}\right)\right]_+. \tag{4.7}$$

*Moreover, the resulting decrease at the k-th iteration satisfies*

$$f\left(x_k\right) - f(R_{x_k}(\alpha_k \eta_k)) > c_{\mathcal{R}} \left\|g_k\right\|_{x_k}^2 - v_k, \tag{4.8}$$

*where*

$$c_{\mathcal{R}} = \rho \min \left\{ \tau_{\min}, \frac{2(1-\rho)\tau_{\min}\theta}{L\tau_{\max}} \right\}. \tag{4.9}$$

**Proof.** Since $k \in \mathcal{R}$, the search direction is given by $\eta_k = -g_k$, which satisfies

$$\left\langle g_k, \eta_k \right\rangle_{x_k} = -\|g_k\|_{x_k}^2, \quad \|\eta_k\|_{x_k} = \|g_k\|_{x_k}. \tag{4.10}$$

If the decrease condition (3.1) holds for $\alpha_k = \tau_k$, then the bound (4.7) holds. Moreover, combining (3.1) with (4.10) gives

$$v_k + f(x_k) - f(R_{x_k}(\alpha_k \eta_k)) > -\rho \tau_k \left\langle g_k, \eta_k \right\rangle_{x_k} \geq \rho \tau_{\min} \left\|g_k\right\|_{x_k}^2,$$

hence (4.8) also holds. Now suppose that the line-search condition (3.1) fails at some step. Then, from the Taylor expansion inequality (4.5) and (4.10) yields

$$v_k - \rho \alpha_k \|g_k\|_{x_k}^2 \leq f(R_{x_k}(\alpha_k \eta_k)) - f(x_k) \leq -\alpha_k \|g_k\|_{x_k}^2 + \frac{L}{2}\alpha_k^2 \|g_k\|_{x_k}^2. \tag{4.11}$$

Then we have that

$$\frac{L}{2}\alpha_k^2 \|g_k\|_{x_k}^2 \geq (1-\rho)\alpha_k \|g_k\|_{x_k}^2.$$

This implies that $\alpha_k \geq \frac{2(1-\rho)}{L}$. Since $\alpha_k = \tau_k \theta^{j_k} \leq \tau_{\max}\theta^{j_k}$, it follows from (4.11) that $j_k \leq \bar{j}_{\mathcal{R},k}$. Therefore, the line-search process terminates in at most $j_k \leq \lfloor \bar{j}_{\mathcal{R},k} + 1 \rfloor$ steps, where $\bar{j}_{\mathcal{R}}$ is defined in (4.7). Moreover, since the line-search does not terminate after $j_k - 1$ iterations, we have

$$\theta^{j_k - 1} \geq \frac{2(1-\rho)}{L\tau_{\max}},$$

which is equivalent to

$$\alpha_k = \tau_k \theta^{j_k} \geq \frac{2(1-\rho)\tau_{\min}\theta}{L\tau_{\max}}.$$

Consequently, the function decreases at iteration $k$ satisfies

$$v_k + f(x_k) - f(R_{x_k}(\alpha_k \eta_k)) \geq \rho \alpha_k \|g_k\|_{x_k}^2 \geq \frac{2\rho(1-\rho)\tau_{\min}\theta}{L\tau_{\max}} \|g_k\|_{x_k}^2 \geq c_{\mathcal{R}} \|g_k\|_{x_k}^2.$$

Hence, the decrease condition (4.8) follows, where $c_{\mathcal{R}}$ is defined as (4.9). □

The results of Lemmas 4.1 and 4.2 are instrumental to bound the number of iterations necessary to reach an approximate stationary point.

## 4.2. Main results

With respect to $\{v_k\}_{k=0}^{+\infty}$ that controls the amount of the non-monotonicity, we shall consider the following assumption.

**Assumption 4.3.** $\sum_{k=0}^{+\infty} v_k < +\infty$.

Our main result is a bound on the number of iterations performed by the algorithm prior to reaching an $\epsilon$-stationary point. This bound also applies to the number of gradient evaluations.

**Theorem 4.1.** *Let Assumptions 4.1, 4.2, 4.3 hold, and* $1+p-q \geq 0$. *Then, the number of iterations (and objective gradient evaluations) required by Algorithm 1 to reach a point satisfying* (1.2) *is at most*

$$K_\epsilon := \left\lfloor \frac{f\left(x_0\right)-f_{low}\ +M}{c_{\mathcal{R}}\epsilon^2}+\frac{f\left(x_0\right)-f_{low}\ +M}{c_{\mathcal{N}}\epsilon^{\max\{1+p,2(1+p-q)\}}} \right\rfloor. \tag{4.12}$$

**Proof.** Let $K \in N$ be such that $\|\mathrm{grad}f\left(x_k\right)\| > \epsilon$ for any $k = 0,\ldots,K-1$. Following our partitioning (4.1), we define the index sets

$$\mathcal{N}_K := \mathcal{N} \cap \{0,\ldots,K-1\},\ \ \mathcal{R}_K := \mathcal{R} \cap \{0,\ldots,K-1\}.$$

For any $k \in \mathcal{N}_K$, the result of Lemma 4.1 applies, and we have

$$\begin{aligned} f\left(x_k\right)+v_k-f(R_{x_k}(\alpha_k\eta_k)) \geq & c_{\mathcal{N}}\min\left\{\|g_k\|_{x_k}^{1+p},\|g_k\|_{x_k}^{2(1+p-q)}\right\} \\ \geq & c_{\mathcal{N}}\epsilon^{\max\{1+p,2(1+p-q)\}}. \end{aligned}$$

On the other hand, if $k \in \mathcal{R}_K$, applying Lemma 4.2 gives

$$f\left(x_k\right)+v_k-f(R_{x_k}(\alpha_k\eta_k)) \geq c_{\mathcal{R}}\|g_k\|_{x_k}^2 \geq c_{\mathcal{R}}\epsilon^2.$$

We now consider the sum of function changes over all $k \in \{0,\ldots,K-1\}$. By Assumption 4.2, we obtain

$$\begin{aligned} &\sum_{k\in\mathcal{N}_K} c_{\mathcal{N}}\epsilon^{\max\{1+p,2(1+p-q)\}}+\sum_{k\in\mathcal{R}_K} c_{\mathcal{R}}\epsilon^2 \\ \leq &\sum_{k\in\mathcal{N}_K}\left[f\left(x_k\right)+v_k-f(R_{x_k}(\alpha_k\eta_k))\right]+\sum_{k\in\mathcal{R}_K}\left[f\left(x_k\right)+v_k-f(R_{x_k}(\alpha_k\eta_k))\right] \\ \leq &\sum_{k=0}^{K-1}\left[f\left(x_k\right)-f(R_{x_k}(\alpha_k\eta_k))\right]+\sum_{k=0}^{K-1} v_k \\ \leq & f\left(x_0\right)-f\left(x_K\right)+\sum_{k=0}^{K-1} v_k \\ \leq & f\left(x_0\right)-f_{\text{low}}\ +\sum_{k=0}^{K-1} v_k. \end{aligned}$$

According to Assumption 4.3, there exists $M > 0$ such that $\sum_{k=0}^{K-1} v_k \leq M < +\infty$, then

$$\sum_{k\in\mathcal{N}_K} c_{\mathcal{N}}\epsilon^{\max\{1+p,2(1+p-q)\}}+\sum_{k\in\mathcal{R}_K} c_{\mathcal{R}}\epsilon^2 \leq f\left(x_0\right)-f_{\text{low}}\ +M.$$

Since the left-hand side consists of two sums of positive terms, the above inequality implies that

$$f\left(x_0\right)-f_{\text{low}}\ +M > \sum_{k\in\mathcal{N}_K} c_{\mathcal{N}}\epsilon^{\max\{1+p,2(1+p-q)\}},$$

which is equivalent to

$$|\mathcal{N}_K| < \frac{f(x_0) - f_{\text{low}} + M}{c_{\mathcal{N}}\epsilon^{\max\{1+p,2(1+p-q)\}}},$$

and

$$f\left(x_0\right) - f_{\text{low}} + M > \sum_{k\in\mathcal{R}_K} c_{\mathcal{R}}\epsilon^2,$$

which is equivalent to

$$|\mathcal{R}_K| < \frac{f\left(x_0\right) - f_{\text{low}} + M}{c_{\mathcal{R}}\epsilon^2}.$$

Using $|\mathcal{N}_K| + |\mathcal{R}_K| = K$ finally yields

$$K < \frac{f\left(x_0\right) - f_{\text{low}} + M}{c_{\mathcal{R}}}\epsilon^{-2} + \frac{f\left(x_0\right) - f_{\text{low}} + M}{c_{\mathcal{N}}}\epsilon^{-\max\{1+p,2(1+p-q)\}}.$$

Hence $K \le K_\epsilon$. □

Note that the complexity bound of Theorem 4.1 also guarantees global convergence of the algorithmic framework, since it holds for $\epsilon$ arbitrarily close to 0. More precisely, it is possible to show that

$$\liminf_{k\to\infty} \|\text{grad} f\left(x_k\right)\|_{x_k} = 0,$$

which is a typical convergence result for RCG using line-search.

By combining the result of Theorem 4.1 with that of Lemmas 4.1 and 4.2, we can also provide an evaluation complexity bound of Algorithm 1.

**Corollary 4.1.** *Under the assumptions of Theorem 4.1, suppose further that* $1 + p - 2q = 0$. *Then, the number of function evaluations required by Algorithm 1 to reach a point satisfying* (1.2) *is at most*

$$\left\lfloor \left[\log_\theta\left(\frac{2(1-\rho)\sigma}{\kappa^2 L\tau_{\max}}\right)\right]_+ + 1\right\rfloor K_\epsilon,$$

*where* $K_\epsilon$ *is defined in* (4.12).

**Proof.** Since $1 + p - 2q = 0$, we have

$$\forall k \in \mathcal{N}, \quad \bar{j}_{\mathcal{N},k} = \left[\log_\theta\left(\frac{2(1-\rho)\sigma}{\kappa^2 L\tau_{\max}}\right)\right]_+,$$

hence this quantity is independent of the iteration index $k$. Moreover,

$$\max\left\{\left[\log_\theta\left(\frac{2(1-\rho)\sigma}{\kappa^2 L\tau_{\max}}\right)\right]_+, \bar{j}_{\mathcal{R}}\right\} = \left[\log_\theta\left(\frac{2(1-\rho)\sigma}{\kappa^2 L\tau_{\max}}\right)\right]_+.$$

Since $\kappa \ge 1$ and $\sigma \le 1$. As a result, any iteration requires at most

$$\left\lfloor \left[\log_\theta\left(\frac{2(1-\rho)\sigma}{\kappa^2 L\tau_{\max}}\right)\right]_+ + 1\right\rfloor.$$

function evaluations. Combining this number with the result of Theorem 4.1 completes the proof. □

Figure 1 displays the choices of $p$ and $q$ satisfying the condition $1 + p \le 2$, as these choices lead to a more favorable complexity bound for non-restarted iterations. Opting for $p$ values between 0 and 1 implies that the bound ranges between $\epsilon^{-2}$ and $\epsilon^{-1}$, suggesting that the overall number of iterations could potentially outperform that of gradient descent. In the worst case, this approach attains a prescribed tolerance $\epsilon > 0$ with complexity of $\mathcal{O}(\epsilon^{-2})$.

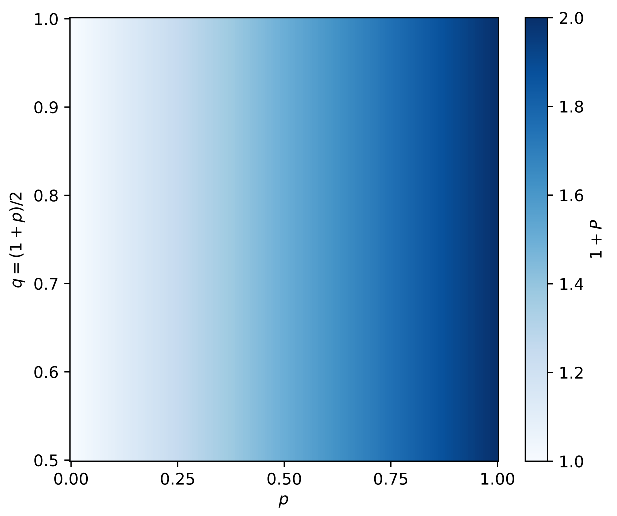


**Figure 1.** Possible complexity values for different values of $p$ and $q$ satisfying $1 + p - 2q = 0$ and $1 + p - q \geq 0$.

## 5. Numerical experiments

This section compares the performances of the existing RCG methods with those of the proposed methods. We solve two types of Riemann optimization problems (Problems 5.1 and 5.2) on several manifolds and objective functions. All experiments use a Huawei Matebook 14 (2021) with a 2.42 GHz Intel Core i5, 16 GB. The algorithms were written in Python 3.7.6 with the NumPy 1.19.0 package, the Matplotlib 3.2.2 package and Pymanopt 0.2.6rc1 package [39].

Problem 5.1 is the Rayleigh-quotient minimization problem on the unit sphere [31]. In the experiments, we set $n = 10$ and generate a matrix $A \in \mathcal{S}^n_{++}$ with randomly chosen by *randn(n)*.

**Problem 5.1.** *For $A \in \mathcal{S}^n_{++}$,*

$$\begin{aligned} \min \quad & f(x) = x^\top Ax, \\ s.t. \quad & x \in \mathbb{S}^{n-1} := \{x \in \mathbb{R}^n : \|x\| = 1\}, \end{aligned}$$

*where $\mathcal{S}^n_{++}$ denotes the set of all symmetric positive-defnite matrices.*

Problem 5.2 is the Brockett-cost-function minimization problem on a Stiefel manifold [31]. In the experiments, we set $(n, p) = (20, 5)$ and $N := \operatorname{diag}(1, 2, \cdots, p)$. The matrix $A = M^\top M$ and $M$ is generated randomly by *randn(n)*.

**Problem 5.2.** *For $A \in \mathcal{S}^n_{++}$, and $N = \operatorname{diag}(\mu_0, \cdots, \mu_p)(0 \leq \mu_0 \leq \cdots \leq \mu_p)$,*

$$\begin{aligned} \min \quad & f(X) = \operatorname{tr}(X^\top AXN), \\ s.t. \quad & X \in \operatorname{St}(p, n) := \{X \in \mathbb{R}^{n\times p} : X^\top X = I_p\}, \end{aligned}$$

*where $\mathcal{S}^n_{++}$ denotes the set of all symmetric positive-defnite matrices.*

We solved the above two problems 100 times with each algorithm, that is, 200 times in total. If the stopping condition

$$\|\operatorname{grad} f(X_k)\|_{X_k} < 10^{-6}$$

was satisfied, we determined that a sequence had converged to an optimal solution. For numerical comparison, we calculate a performance profile [13] for each algorithm to show the advantages of our algorithms.

The performance profile $P_s : \mathbb{R} \to [0,1]$ is defined as follows: Let $\mathcal{P}$ and $\mathcal{S}$ be the set of problems and solvers, respectively. For each $p \in \mathcal{P}$ and $s \in \mathcal{S}$, we denote

$$t_{p,s} := (\text{iterations or time required to solve problem p by solver s}).$$

Furthermore, we define the performance ratio $r_{p,s}$ as

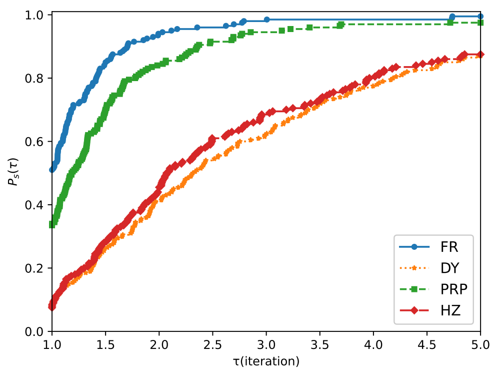
$$r_{p,s} := \frac{t_{p,s}}{\min_{s' \in \mathcal{S}} \ t_{p,s'}}$$

and define the performance profile, for all $\tau \in \mathbb{R}$, as

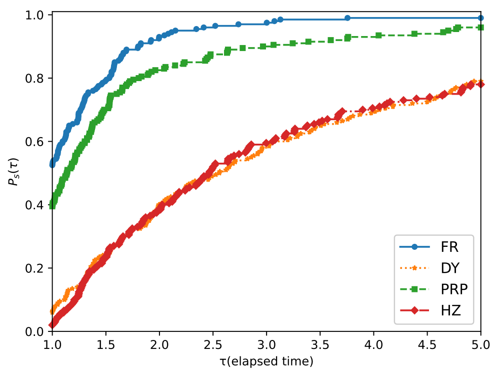
$$P_s(\tau) := \frac{\#\{p \in \mathcal{P} : r_{p,s} \leq \tau\}}{\#\mathcal{P}},$$

where $\#\mathcal{P}$ denotes the number of elements of a set $\mathcal{P}$.

To compare the numerical performance of the Restart RCG ($p = 1$) by Algorithm 1, Armijo RCG [31], and Strong Wolfe RCG [31] in practice, we proceed as follows. First, set the Restart RCG with $p = 1$, and compare the performance profiles of the FR, DY, PRP, and HZ formulas given in (2.6) using Figure 2. Second, report the numerical results of the above three algorithms in Figure 3. Finally, compare the results for Problem 5.1 with $n = 10, 20, 50, 100, 1000$, and Problem 5.2 with $(n, p) = (10, 5), (20, 5), (50, 10), (100, 10), (1000, 50)$ and $N := \text{diag}(1, 2, \cdots, p)$, which are summarized in Tables 1, 2, 3, and 4 to enhance the persuasiveness of the results.



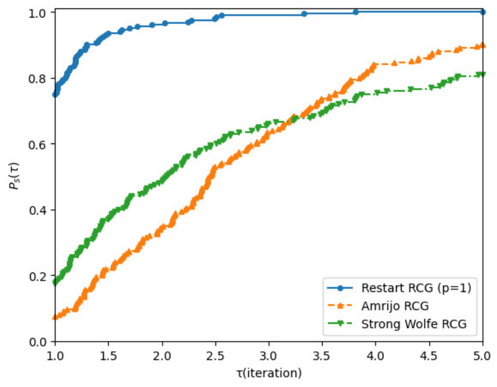
(a) iteration



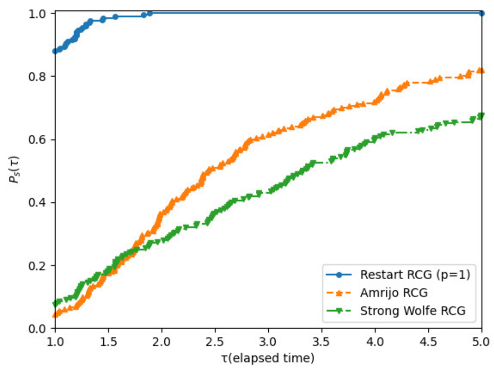
(b) elapsed time

**Figure 2.** Performance profiles of each algorithm versus the number of iterations (a) and the elapsed time (b) by Algorithm 1 using the FR, DY, PRP, and HZ methods from (2.6).

Figure 2 (a) and (b) display the performance profiles of each algorithm in terms of the number of iterations and the elapsed time, respectively, obtained using Algorithm 1 with the FR, DY, PRP, and HZ methods. The results indicate that the FR method solved the most problems, comparable to the number solved by the PRP method. It is clear that the DY and HZ methods are not compatible with Algorithm 1. Overall, the Restart RCG ($p = 1$) with the FR method exhibits superior performance.



(a) iteration



(b) elapsed time

**Figure 3.** Performance profiles of FR methods versus the number of iterations (a) and elapsed time (b) by Algorithm 1 using the modified restart condition, the Amrijo RCG and the Strong Wolfe RCG.

Figure 3 (a) and (b) illustrate the performance profiles of the FR methods in terms of the number of iterations and the elapsed time, respectively, obtained using Algorithm 1 with modified restart condition, the Amrijo RCG and the Strong Wolfe RCG. We can observe that the Restart RCG ($p = 1$) solves more problems than the other two algorithms in fewer iterations and less time. Therefore, our proposed Restart RCG ($p = 1$) has good numerical performance.

**Table 1.** Numerical results of our algorithm using the FR, DY, PRP, and HZ methods from (2.6) on Problem 5.1 with different $n$.

| $n$ | FR | | DY | | PRP | | HZ | |
|---|---|---|---|---|---|---|---|---|
| | iter | time | iter | time | iter | time | iter | time |
| 10 | **106** | **0.099** | 1244 | 1.166 | 110 | 0.101 | 139 | 0.134 |
| 20 | **155** | **0.145** | 641 | 0.621 | 182 | 0.173 | 305 | 0.299 |
| 50 | **562** | **0.543** | 2780 | 2.747 | 722 | 0.715 | 1717 | 1.681 |
| 100 | **681** | **1.063** | 5583 | 8.850 | 750 | 1.188 | 5069 | 8.705 |
| 1000 | **4639** | 28.044 | 10000 | 52.056 | 4902 | **26.365** | 10000 | 52.843 |

**Table 2.** Numerical results of our algorithm using the FR, DY, PRP, and HZ methods from (2.6) on Problem 5.2 with different $n$ and $p$.

| $(n, p)$ | FR | | DY | | PRP | | HZ | |
|---|---|---|---|---|---|---|---|---|
| | iter | time | iter | time | iter | time | iter | time |
| (10, 5) | **568** | **0.656** | 1150 | 1.344 | 844 | 0.983 | 1951 | 2.426 |
| (20, 5) | **670** | **0.782** | 2424 | 2.952 | 1076 | 1.188 | 2204 | 2.516 |
| (50, 10) | **8462** | **9.092** | 10959 | 11.897 | 11279 | 12.357 | 59785 | 67.687 |
| (100, 10) | **20705** | **26.496** | 30077 | 40.906 | 21472 | 27.338 | 94539 | 126.551 |
| (1000, 50) | 19449 | 1000.027 | 18049 | **1000.015** | 18590 | 1000.042 | **17595** | 1000.023 |

Tables 1 and 2 show the experimental results of our algorithm using the FR, DY, PRP, and

HZ methods on Problems 5.1 and 5.2 with different $n$ and $p$. We can observe that Restart RCG ($p = 1$) with FR method performs best in most cases, while on larger problems, the performance of several different $\beta$ is comparable.

**Table 3.** Numerical results of our algorithm compared with other algorithms on Problem 5.1 with different $n$.

| $n$ | Restart RCG ($p = 1$) | | Armijo RCG | | Strong Wolfe RCG | |
|---|---|---|---|---|---|---|
| | iter/restart | time | iter | time | iter | time |
| 10 | **103**/64 | **0.0979** | 439 | 0.4557 | 308 | 0.4557 |
| 20 | **213**/131 | **0.2009** | 495 | 0.4870 | 1508 | 2.3085 |
| 50 | **415**/249 | **0.3987** | 900 | 0.9155 | 1549 | 2.3761 |
| 100 | **610**/345 | **1.0157** | 2610 | 5.2304 | 2136 | 6.0333 |
| 1000 | 6949/3688 | **28.3746** | 5528 | 36.6770 | **3641** | 29.1726 |

**Table 4.** Numerical results of our algorithm compared with other algorithms on Problem 5.2 with different $n$ and $p$.

| $(n, p)$ | Restart RCG ($p = 1$) | | Armijo RCG | | Strong Wolfe RCG | |
|---|---|---|---|---|---|---|
| | iter/restart | time | iter | time | iter | time |
| (10, 5) | **580**/369 | **0.569** | 1161 | 1.529 | 1108 | 1.857 |
| (20, 5) | **1164**/730 | **1.428** | 2157 | 3.730 | 3631 | 8.733 |
| (50, 10) | **7819**/4941 | **9.532** | 10151 | 20.521 | 15633 | 40.121 |
| (100, 10) | **42107**/26866 | **52.693** | 49471 | 105.715 | 62339 | 163.262 |
| (1000, 50) | 17926/10463 | **1000.030** | **5488** | 1000.083 | 7312 | 1000.086 |

**Table 5.** Numerical efficiency on large-scale Problems 5.1 and 5.2. AST: Average Step Time (seconds), LST%: Line-Search Time Percentage.

| | Restart RCG ($p = 1$) | | Armijo RCG | | Strong Wolfe RCG | |
|---|---|---|---|---|---|---|
| | AST | LST% | AST | LST% | AST | LST% |
| Problem 5.1 $n = 1000$ | **0.0039** | **68.7%** | 0.0065 | 78.9% | 0.0074 | 81.8% |
| Problem 5.2 $(n, p) = (1000, 50)$ | **0.0557** | **83.9%** | 0.1822 | 96.3% | 0.1367 | 94.9% |

Tables 2 and 3 present the numerical results of our algorithm compared with other algorithms on Problems 5.1 and 5.2 with different $n$ and $p$. It is observed that the average percentage of restarted iterations for the two problems are 58.79% and 63.31% respectively. That is, half of the RCG directions are not of descent type, for which we employed a restart procedure to improve the computational efficiency. Thereby, the Restart RCG ($p = 1$) outperforms the Armijo RCG and the Strong Wolfe RCG.

**Remark 5.1.** In large-scale problems, although the Restart RCG method requires slightly more iterations, its concise iterative structure avoids frequent function and gradient evaluations associated with complex line search procedures. As shown in Table 5, it achieves a lower line-search time percentage compared to the Armijo and Strong Wolfe variants, thereby maintaining competitive total runtime. This demonstrates its practical effectiveness and efficiency in large-scale optimization scenarios.

All in all, the results obtained in the figures and tables are consistent, indicating that the proposed method the Restart RCG ($p = 1$) by Algorithm 1 enhances the efficiency and practical applicability of solving Rayleigh-quotient and Brockett-cost-function minimization problems.

## 6. Conclusion

In this paper, we presented a novel RCG method with a modified restart condition, enhancing its performance by utilizing the Riemannian BB step-size along with a non-monotone line-search strategy. Theoretically, we proved that the algorithms take at most $\mathcal{O}(\epsilon^{-2}) + \mathcal{O}(\epsilon^{-\max\{1+p,2(1+p-q)\}})$ iterations to find an $\epsilon$-critical point, where $p, q > 0$. In particular, when $p = q = 1$, the complexity reduces to $\mathcal{O}(\epsilon^{-2})$, matching the iteration complexity result of the Riemannian gradient method. To our knowledge, this is the first RCG method with iteration complexity guarantees. Our experiments on the Rayleigh quotient minimization problem and the Brockett cost function minimization problem show that the proposed RCG method is empirically verified and improves the efficiency and practical applicability of solving the problem.

As a topic for future research, it would be interesting to investigate iteration complexity estimates even when $\{v_k\}$ is not convergence. Additionally, the modified restart condition in the algorithm can also be extended to the strong Wolfe line-search.

## Acknowledgements

This work was conceived and completed by all authors collectively. The authors sincerely appreciate the anonymous reviewers and editors for the invaluable suggestions, which significantly improved the quality of this paper.

## References

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

[2] G. C. Bento, O. P. Ferreira and J. G. Melo, *Iteration-complexity of gradient, subgradient and proximal point methods on Riemannian manifolds*, Journal of Optimization Theory and Applications, 2017, 173(2), 548–562.

[3] M. Bortoloti, T. A. Fernandes and O. P. Ferreira, *An efficient damped Newton-type algorithm with globalization strategy on Riemannian manifolds*, Journal of Computational and Applied Mathematics, 2022, 403, 113853.

[4] N. Boumal, *An Introduction to Optimization on Smooth Manifolds*, Cambridge University Press, Cambridge, UK, 2023.

[5] N. Boumal, P.-A. Absil and C. Cartis, *Global rates of convergence for nonconvex optimization on manifolds*, IMA Journal of Numerical Analysis, 2019, 39(1), 1–33.

[6] N. Boumal, P.-A. Absil and C. Cartis, *Global rates of convergence for nonconvex optimization on manifolds*, IMA Journal of Numerical Analysis, 2019, 39(1), 1–33.

[7] C. Cartis, P. R. Sampaio and P. L. Toint, *Worst-case evaluation complexity of non-monotone gradient-related algorithms for unconstrained optimization*, Optimization, 2015, 64(5), 1349–1361.

[8] Y. Chen, K. Kuang and X. Yan, *A modified PRP-type conjugate gradient algorithm with complexity analysis and its application to image restoration problems*, Journal of Computational and Applied Mathematics, 2023, 427, 115105.

[9] Y.-H. Dai and Y.-X. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property*, SIAM Journal on Optimization, 1999, 10(1), 177–182.

[10] K. Deng and J. Hu, *Decentralized projected Riemannian gradient method for smooth optimization on compact submanifolds*, arXiv preprint, 2023. arXiv:2304.08241.

[11] K. Deng, J. Hu and H. Wang, *Decentralized douglas-rachford splitting methods for smooth optimization over compact submanifolds*, arXiv preprint, 2023. arXiv:2311.16399.

[12] K. Deng and Z. Peng, *A manifold inexact augmented Lagrangian method for nonsmooth optimization on Riemannian submanifolds in Euclidean space*, IMA Journal of Numerical Analysis, 2023, 43(3), 1653–1684.

[13] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 2002, 91, 201–213.

[14] R. Fletcher and C. M. Reeves, *Function minimization by conjugate gradients*, The Computer Journal, 1964, 7(2), 149–154.

[15] G. N. Grapiglia and E. W. Sachs, *A generalized worst-case complexity analysis for non-monotone line searches*, Numerical Algorithms, 2021, 87, 779–796.

[16] L. Grippo, F. Lampariello and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM Journal on Numerical Analysis, 1986, 23(4), 707–716.

[17] W. W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM Journal on Optimization, 2005, 16(1), 170–192.

[18] M. R. Hestenes, E. Stiefel, et al., *Methods of conjugate gradients for solving linear systems*, Journal of Research of the National Bureau of Standards, 1952, 49(6), 409–436.

[19] J. Hu, X. Liu, Z.-W. Wen and Y.-X. Yuan, *A brief introduction to manifold optimization*, Journal of the Operations Research Society of China, 2020, 8, 199–248.

[20] W. Huang, P.-A. Absil and K. A. Gallivan, *A Riemannian BFGS method without differentiated retraction for nonconvex optimization problems*, SIAM Journal on Optimization, 2018, 28(1), 470–495.

[21] R. Chan-Renous-Legoubin and C. W. Royer, *A nonlinear conjugate gradient method with complexity guarantees and its application to nonconvex regression*, EURO Journal on Computational Optimization, 2022, 10, 100044.

[22] G. Montúfar and Y. G. Wang, *Distributed learning via filtered hyperinterpolation on manifolds*, Foundations of Computational Mathematics, 2022, 22(4), 1219–1271.

[23] A. S. Nemirovsky and D. B. Yudin, *Problem complexity and method efficiency in optimization*, Journal of the Operational Research Society, 1984, 35(5).

[24] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, 87, Springer Science & Business Media, Berlin, Germany, 2013.

[25] A. Neumaier, M. Kimiaei and B. Azmi, *Globally linearly convergent nonlinear conjugate gradients without wolfe line search*, Numerical Algorithms, 2024, 1–27.

[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, Berlin, 1999.

[27] E. Polak and G. Ribiere, *Note sur la convergence de méthodes de directions conjuguées*, Revue Française D'informatique et de Recherche Opérationnelle. Série Rouge, 1969, 3(16), 35–43.

[28] B. T. Polyak, *The conjugate gradient method in extremal problems*, USSR Computational Mathematics and Mathematical Physics, 1969, 9(4), 94–112.

[29] M. J. D. Powell, *Restart procedures for the conjugate gradient method*, Mathematical Programming, 1977, 12, 241–254.

[30] E. W. Sachs and S. M. Sachs, *Nonmonotone line searches for optimization algorithms*, Control and Cybernetics, 2011, 40(4), 1059–1075.

[31] H. Sakai and H. Iiduka, *Sufficient descent riemannian conjugate gradient methods*, Journal of Optimization Theory and Applications, 2021, 190(1), 130–150.

[32] H. Sakai, H. Sato and H. Iiduka, *Global convergence of Hager–Zhang type Riemannian conjugate gradient method*, Applied Mathematics and Computation, 2023, 441, 127685.

[33] H. Sato, *Riemannian Optimization and its Applications*, SpringerBriefs in Electrical and Computer Engineering, Springer International Publishing, Cham, Switzerland, 2021.

[34] H. Sato, *Riemannian conjugate gradient methods: General framework and specific algorithms with convergence analyses*, SIAM Journal on Optimization, 2022, 32(4), 2690–2717.

[35] H. Sato, *Riemannian optimization on unit sphere with p-norm and its applications*, Computational Optimization and Applications, 2023, 1–39.

[36] H. Sato and T. Iwai, *A new, globally convergent riemannian conjugate gradient method*, Optimization, 2015, 64(4), 1011–1031.

[37] S. T. Smith, *Optimization techniques on Riemannian manifolds*, arXiv preprint, 2014. arXiv:1407.5965.

[38] G.-J. Song, X.-Z. Wang and M. K. Ng, *Riemannian conjugate gradient descent method for fixed multi rank third-order tensor completion*, Journal of Computational and Applied Mathematics, 2023, 421, 114866.

[39] J. Townsend, N. Koep and S. Weichwald, *Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation*, Journal of Machine Learning Research, 2016, 17(137), 1–5.

[40] M. Yamada and H. Sato, *Conjugate gradient methods for optimization problems on symplectic stiefel manifold*, IEEE Control Systems Letters, 2023, 7, 2719–2724.

[41] H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM Journal on Optimization, 2004, 14(4), 1043–1056.

[42] H. Zhang and S. Sra, *First-order methods for geodesically convex optimization*, in *Conference on Learning Theory*, PMLR, 2016, 1617–1638.

[43] Z. Zhao, X.-Q. Jin and T.-T. Yao, *The Riemannian two-step perturbed Gauss–Newton method for least squares inverse eigenvalue problems*, Journal of Computational and Applied Mathematics, 2022, 405, 113971.

[44] X. Zhu and H. Sato, *Riemannian conjugate gradient methods with inverse retraction*, Computational Optimization and Applications, 2020, 77, 779–810.

[45] X. Zhu and H. Sato, *Cayley-transform-based gradient and conjugate gradient algorithms on Grassmann manifolds*, Advances in Computational Mathematics, 2021, 47, 1–28.

[46] X. Zhu and C. Shen, *Practical gradient and conjugate gradient methods on flag manifolds*, Computational Optimization and Applications, 2024, 1–34.

Received December 2024; Accepted July 2025; Available online August 2025.