Comput Optim Appl
DOI 10.1007/s10589-015-9801-1

A Dai–Yuan-type Riemannian conjugate gradient method with the weak Wolfe conditions

Hiroyuki Sato[1]

Received: 8 September 2014

Abstract This article describes a new Riemannian conjugate gradient method and presents a global convergence analysis. The existing Fletcher–Reeves-type Riemannian conjugate gradient method is guaranteed to be globally convergent if it is implemented with the strong Wolfe conditions. On the other hand, the Dai–Yuan-type Euclidean conjugate gradient method generates globally convergent sequences under the weak Wolfe conditions. This article deals with a generalization of Dai–Yuan's Euclidean algorithm to a Riemannian algorithm that requires only the weak Wolfe conditions. The global convergence property of the proposed method is proved by means of the scaled vector transport associated with the differentiated retraction. The results of numerical experiments demonstrate the effectiveness of the proposed algorithm.

Keywords Riemannian optimization · Conjugate gradient method · Global convergence · Weak Wolfe conditions · Scaled vector transport

Mathematics Subject Classification 65K05 · 49M37 · 90C30

1 Introduction

The Euclidean non-linear conjugate gradient method [8] for minimizing a non-linear objective function $f : \mathbb{R}^n \to \mathbb{R}$ without any constraints is a generalization of the

✉ Hiroyuki Sato
hsato@ms.kagu.tus.ac.jp

[1] Department of Management Science, Tokyo University of Science, Tokyo 162-8601, Japan

Published online: 29 October 2015

linear conjugate gradient method proposed by Hestenes and Stiefel [6]. The steepest descent method, which is the simplest iterative optimization technique, does not need the Hessian of the objective function, but generally suffers from slow convergence. Newton's method has a property of locally quadratic convergence, but this does not extend to global convergence. Additionally, we need to compute the Hessian of the objective function at each step in Newton's method. On the other hand, the conjugate gradient method ensures global convergence and is much faster than the steepest descent method. Furthermore, it does not need the Hessian of the objective function. Therefore, the conjugate gradient method is one of the most important optimization methods, and has been intensively researched.

The non-linear conjugate gradient method in Euclidean space $\mathbb{R}^n$ is characterized by its computation of search directions. The search direction η_k at the current iterate $x_k \in \mathbb{R}^n$ is computed by

$$\eta_k = -\nabla f(x_k) + \beta_k \eta_{k-1}, \qquad k \geq 0, \tag{1}$$

where $\beta_0 = 0$ and β_k is a parameter that determines the property of the conjugate gradient method. There are various choices of β_k, and a good choice leads to better convergence. As with other line-search-based optimization methods, once a search direction is computed, the next iterate x_{k+1} is computed by

$$x_{k+1} = x_k + \alpha_k \eta_k, \tag{2}$$

where the step size $\alpha_k > 0$ is computed such that α_k approximately satisfies

$$f(x_k + \alpha_k \eta_k) \approx \min_{\alpha > 0} \{f(x_k + \alpha \eta_k)\}.$$

A frequently used rule for computing the step size is the Wolfe rule. Under the Wolfe rule, α_k at the k-th iterate is computed such that α_k satisfies the Wolfe conditions

$$f(x_k + \alpha_k \eta_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T \eta_k, \tag{3}$$

$$\nabla f(x_k + \alpha_k \eta_k)^T \eta_k \geq c_2 \nabla f(x_k)^T \eta_k \tag{4}$$

for predetermined constants c_1 and c_2 with $0 < c_1 < c_2 < 1$. In practice, c_1 and c_2 are often taken so as to satisfy $0 < c_1 < c_2 < 1/2$ in the conjugate gradient method. To avoid confusion with the strong Wolfe conditions (in which the inequality (4) is replaced by a stricter condition $|\nabla f(x_k + \alpha_k \eta_k)^T \eta_k| \leq c_2 |\nabla f(x_k)^T \eta_k|$), we refer to the Wolfe conditions (3) and (4) as the weak Wolfe conditions.

We are interested in how to choose a good β_k in (1). A well-known choice proposed by Fletcher and Reeves [5] is

$$\beta_k^{\mathrm{FR}} = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})}.$$

If the step sizes are computed so as to satisfy the strong Wolfe conditions, the conjugate gradient method with β_k^{FR} has global convergence. In [2], Dai and Yuan proposed the following refinement of β_k^{FR}:

$$\beta_k^{\mathrm{DY}} = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\eta_{k-1}^T y_k}, \qquad y_k = \nabla f(x_k) - \nabla f(x_{k-1}). \tag{5}$$

An advantage of β_k^{DY} is that it ensures the conjugate gradient method is globally convergent when implemented with only the weak Wolfe conditions. There is no longer a need to assume that each step size satisfies the strong Wolfe conditions.

Beyond unconstrained optimization methods in Euclidean space, the idea of Riemannian optimization, or optimization on Riemannian manifolds, has recently been developed [1,3]. Unconstrained optimization methods, such as the steepest descent method and Newton's method, have been generalized to Riemannian manifolds. The conjugate gradient method has been generalized to Riemannian manifolds to some extent, but remains in the developmental stage. In [1], Absil, Mahony, and Sepulchre introduced the notion of a vector transport to implement a Riemannian conjugate gradient method. By means of this vector transport, Ring and Wirth performed a global convergence analysis of the Fletcher–Reeves-type Riemannian conjugate gradient method under the assumption that the vector transport as the differentiated retraction does not increase the norm of the search direction vector [9]. In [10], Sato and Iwai introduced the notion of a scaled vector transport. This allowed them to develop an improved method, with a global convergence property that could be proved without the assumption made in [9].

The purpose of this article is to propose a new choice of β_k for the Riemannian conjugate gradient method based on Dai–Yuan's β_k^{DY} in the Euclidean conjugate gradient method. We will also prove the global convergence property of the proposed algorithm under the weak Wolfe conditions. Furthermore, we perform some numerical experiments to demonstrate that the proposed Dai–Yuan-type Riemannian conjugate gradient method is preferable to the existing Fletcher–Reeves-type method developed in [10], and show that a step size satisfying the weak Wolfe conditions is easier to find than one that satisfies the strong Wolfe conditions.

This article is organized as follows. In Sect. 2, we introduce several geometric objects necessary for Riemannian optimization. We also define a Riemannian version of the weak Wolfe conditions, which are important in our algorithm. In Sect. 3, we review the Dai–Yuan-type Euclidean conjugate gradient method, and discuss how to generalize β_k^{DY} to β_k on a Riemannian manifold. We take an approach based on another expression of (5), and propose a new algorithm. Section 4 provides a global convergence analysis of the present algorithm. This is analogous to a discussion in [2]. The notion of a scaled vector transport introduced in [10] plays an important role in our analysis. In Sect. 5, we describe some numerical experiments that are intended to evaluate the performance of the proposed algorithm. The results show that the proposed Dai–Yuan-type algorithm is preferable to the Fletcher–Reeves-type algorithm. Our concluding remarks are presented in Sect. 6.

2 General Riemannian optimization and Riemannian conjugate gradient method

In this section, we briefly review Riemannian optimization, especially the Riemannian conjugate gradient method. Our problem is as follows.

Problem 2.1

$$\begin{aligned} &\text{minimize} \quad f(x), \\ &\text{subject to} \quad x \in M, \end{aligned}$$

where M is a Riemannian manifold endowed with a Riemannian metric $\langle \cdot, \cdot \rangle$ and the norm of a tangent vector $\xi \in T_x M$ is defined to be $\|\xi\|_x = \sqrt{\langle \xi, \xi \rangle_x}$, and where f is a smooth objective function. Note that $\langle \cdot, \cdot \rangle_x$ denotes the inner product on $T_x M$.

In Riemannian optimization, we have to replace several quantities used in Euclidean optimization with appropriate quantities on the Riemannian manifold $(M, \langle \cdot, \cdot \rangle)$ in question. For example, the search direction η_k at the current point $x_k \in M$ must be a tangent vector to M at x_k. In iterative optimization methods, we perform a line search on appropriate curves on M. Such a curve should emanate from x_k in the direction of η_k, and can be defined by means of a retraction. A retraction is defined as follows [1].

Definition 2.1 Let M and TM be a manifold and the tangent bundle of M, respectively. Let $R : TM \to M$ be a smooth map and R_x be the restriction of R to $T_x M$. R is called a retraction on M if it has the following properties.

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

$$\mathrm{D}R_x(0_x) = \mathrm{id}_{T_x M},$$

where $\mathrm{D}R_x(0_x)$ denotes the derivative of R_x at 0_x, and $\mathrm{id}_{T_x M}$ is the identity map on $T_x M$.

Using a retraction, the updating formula for line-search-based Riemannian optimization methods can be written as

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k), \tag{6}$$

where the step size α_k is computed so as to satisfy a certain condition. Note that (6) replaces (2). Throughout this article, we consider the weak Wolfe conditions

$$f\left(R_{x_k}(\alpha_k \eta_k)\right) \leq f(x_k) + c_1 \alpha_k \langle \mathrm{grad} f(x_k), \eta_k \rangle_{x_k}, \tag{7}$$

$$\langle \mathrm{grad} f\left(R_{x_k}(\alpha_k \eta_k)\right), \mathrm{D}R_{x_k}(\alpha_k \eta_k)[\eta_k] \rangle_{R_{x_k}(\alpha_k \eta_k)} \geq c_2 \langle \mathrm{grad} f(x_k), \eta_k \rangle_{x_k}, \tag{8}$$

where $0 < c_1 < c_2 < 1$ [9,10]. Note that, on a general Riemannian manifold M, $\mathrm{grad} f$ is no longer the Euclidean gradient. In fact, $\mathrm{grad} f$ is a vector field on M, and depends on the Riemannian metric.

In generalizing the Euclidean conjugate gradient method to that on a manifold M, the right-hand side of (1) cannot be computed, since $\operatorname{grad} f(x_k) \in T_{x_k}M$ and $\eta_{k-1} \in T_{x_{k-1}}M$; that is, the two terms on the right-hand side of (1) belong to different tangent spaces.

In [1], the notion of a vector transport was introduced to transport a tangent vector to another tangent space.

Definition 2.2 A vector transport $\mathcal{T}$ on a manifold M is a smooth map

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

satisfying the following properties for all $x \in M$, where $\oplus$ is the Whitney sum [1], that is, $TM \oplus TM = \{(\eta, \xi) \mid \eta, \xi \in T_xM, x \in M\}$.

1. There exists a retraction R, called the retraction associated with $\mathcal{T}$, such that

$$\pi\left(\mathcal{T}_\eta(\xi)\right) = R_x(\eta), \qquad \eta, \xi \in T_xM,$$

   where $\pi\left(\mathcal{T}_\eta(\xi)\right)$ denotes the foot of the tangent vector $\mathcal{T}_\eta(\xi)$,
2. $\mathcal{T}_{0_x}(\xi) = \xi$ for all $\xi \in T_xM$,
3. $\mathcal{T}_\eta(a\xi + b\zeta) = a\mathcal{T}_\eta(\xi) + b\mathcal{T}_\eta(\zeta)$ for all $a, b \in \mathbb{R}$, $\eta, \xi, \zeta \in T_xM$.

We can generalize (1) using a vector transport $\mathcal{T}$ on M as

$$\eta_k = -\operatorname{grad} f(x_k) + \beta_k \mathcal{T}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1}), \qquad k \geq 0, \tag{9}$$

or equivalently,

$$\eta_k = -\operatorname{grad} f(x_k) + \beta_k \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1}), \qquad k \geq 0, \tag{10}$$

where $\mathcal{T}^{(k)} := c^{(k)}\mathcal{T}$ and $c^{(k)}$ is a positive number. Thus, we have replaced β_k with $\beta_k c^{(k)}$ in (9) to obtain (10). However, the latter expression (10) is more useful in our discussion. We will propose a new choice of β_k in Sect. 3.

A reasonable choice of a vector transport is the differentiated retraction $\mathcal{T}^R$ defined by

$$\mathcal{T}^R_\eta(\xi) := \mathrm{D}R_x(\eta)[\xi], \qquad x \in M, \ \eta, \xi \in T_xM. \tag{11}$$

Note that the second condition (8) of the weak Wolfe conditions can be rewritten using $\mathcal{T}^R$ as

$$\langle \operatorname{grad} f\left(R_{x_k}(\alpha_k\eta_k)\right), \mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\rangle_{R_{x_k}(\alpha_k\eta_k)} \geq c_2\langle \operatorname{grad} f(x_k), \eta_k\rangle_{x_k}. \tag{12}$$

Furthermore, the scaled vector transport $\mathcal{T}^0$ associated with $\mathcal{T}^R$ [10], which is defined by

$$\mathcal{T}^0_\eta(\xi) = \frac{\|\xi\|_x}{\|\mathcal{T}^R_\eta(\xi)\|_{R_x(\eta)}}\mathcal{T}^R_\eta(\xi), \qquad x \in M, \ \eta, \xi \in T_xM, \tag{13}$$

is important for analyzing the global convergence of our new algorithm. Note that $\mathcal{T}^0$ is not a vector transport, as it does not satisfy the third condition of Definition 2.2. However, $\mathcal{T}^0$ has the important property that

$$\|\mathcal{T}^0_\eta(\xi)\|_{R_x(\eta)} = \|\xi\|_x, \qquad \eta, \xi \in T_x M.$$

3 Dai–Yuan-type Euclidean conjugate gradient method and its Riemannian generalization

3.1 Dai–Yuan-type Euclidean conjugate gradient method

The conjugate gradient method on $\mathbb{R}^n$ with β_k^{DY} defined by (5) was proposed by Dai and Yuan [2]. This method is globally convergent under the assumption that each step size α_k satisfies the weak Wolfe conditions (3) and (4). Since the Fletcher–Reeves-type conjugate gradient method must be implemented with the strong Wolfe conditions, β_k^{DY} is an improved version of β_k^{FR}. We wish to develop a good analogy of β_k^{DY} for Riemannian manifolds.

Note that, in Euclidean space, we can show the equality

$$\beta_k^{\mathrm{DY}} = \frac{\nabla f(x_k)^T \eta_k}{\nabla f(x_{k-1})^T \eta_{k-1}} \tag{14}$$

using Eq. (1) as

$$\begin{aligned}\beta_k^{\mathrm{DY}} &= \frac{\beta_k^{\mathrm{DY}} (\nabla f(x_k) - y_k)^T \eta_{k-1}}{\nabla f(x_{k-1})^T \eta_{k-1}} = \frac{\nabla f(x_k)^T (-\nabla f(x_k) + \beta_k^{\mathrm{DY}} \eta_{k-1})}{\nabla f(x_{k-1})^T \eta_{k-1}} \\ &= \frac{\nabla f(x_k)^T \eta_k}{\nabla f(x_{k-1})^T \eta_{k-1}}.\end{aligned}$$

The equivalent expressions (5) and (14) for β_k^{DY} are useful in analyzing the global convergence of the Dai–Yuan-type algorithm in [2].

3.2 New Riemannian conjugate gradient method based on Euclidean Dai–Yuan β

Throughout this subsection, we assume that all quantities that appear in the denominator of a fraction are nonzero. However, this assumption can be removed after we propose a new algorithm at the end of this section; see Proposition 4.1 for more details. For simplicity, in some of the following computations, we use the notation

$$g_k = \operatorname{grad} f(x_k), \qquad k \geq 0.$$

We expect a Riemannian analogy of the Dai–Yuan-type Euclidean conjugate gradient method to have global convergence if the step sizes satisfy the weak Wolfe conditions.

Let $\mathcal{T}$ be a general vector transport on M. Assume that $\mathcal{T}^{(k)} := c^{(k)}\mathcal{T}$, and use the update formula (10) at the k-th iteration, where $c^{(k)}$ is a positive number.

Note that $\operatorname{grad} f(x_k) \in T_{x_k}M$ and $\operatorname{grad} f(x_{k-1}) \in T_{x_{k-1}}M$ belong to different tangent spaces. There are several possible ways of generalizing the right-hand side of (5), e.g., $\|g_k\|^2_{x_k}/\langle \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1}),\, g_k - \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(g_{k-1})\rangle_{x_k}$ or $\|g_k\|^2_{x_k}/\langle \eta_{k-1},\, (\mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}})^{-1}(g_k) - g_{k-1}\rangle_{x_{k-1}}$.

On the other hand, it seems natural to generalize the right-hand side of (14), which is equivalent to (5) in the Euclidean case, to

$$\beta_k := \frac{\langle \operatorname{grad} f(x_k),\, \eta_k\rangle_{x_k}}{\langle \operatorname{grad} f(x_{k-1}),\, \eta_{k-1}\rangle_{x_{k-1}}}. \tag{15}$$

Note also that Eq. (15), which is an analogy of (14) in the Euclidean case, is helpful in our global convergence analysis, as we will discuss later. Therefore, we start with Eq. (15) to generalize the Dai–Yuan β_k. We should state that Eq. (15) itself cannot be used in a conjugate gradient algorithm, since η_k in the right-hand side of (15) is computed using β_k itself, as in (10). We wish to derive an expression for β_k that does not contain η_k. To this end, we obtain from (10) and (15) that

$$\begin{aligned}\beta_k &= \frac{\langle g_k,\, -g_k + \beta_k \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1})\rangle_{x_k}}{\langle g_{k-1},\, \eta_{k-1}\rangle_{x_{k-1}}} \\ &= \frac{-\|g_k\|^2_{x_k} + \beta_k \langle g_k,\, \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1})\rangle_{x_k}}{\langle g_{k-1},\, \eta_{k-1}\rangle_{x_{k-1}}}.\end{aligned}$$

It follows that

$$\beta_k = \frac{\|\operatorname{grad} f(x_k)\|^2_{x_k}}{\langle \operatorname{grad} f(x_k),\, \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1})\rangle_{x_k} - \langle \operatorname{grad} f(x_{k-1}),\, \eta_{k-1}\rangle_{x_{k-1}}}. \tag{16}$$

We can further show that this β_k in fact satisfies

$$\beta_k = \frac{\|\operatorname{grad} f(x_k)\|^2_{x_k}}{\langle \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1}),\, y_k\rangle_{x_k}}, \tag{17}$$

with $y_k \in T_{x_k}M$ defined by

$$\begin{aligned}y_k = \operatorname{grad} f(x_k) \\ - \frac{\langle \operatorname{grad} f(x_{k-1}),\, \eta_{k-1}\rangle_{x_{k-1}}}{\langle \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\operatorname{grad} f(x_{k-1})),\, \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\eta_{k-1})\rangle_{x_k}} \mathcal{T}^{(k-1)}_{\alpha_{k-1}\eta_{k-1}}(\operatorname{grad} f(x_{k-1})).\end{aligned} \tag{18}$$

This is because, from (18), it follows that

$$\begin{aligned}
&\langle \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k), y_{k+1}\rangle_{x_{k+1}} \\
&= \left\langle \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k), g_{k+1} - \frac{\langle g_k, \eta_k\rangle_{x_k}}{\langle \mathcal{T}^{(k)}_{\alpha_k\eta_k}(g_k), \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k)\rangle_{x_{k+1}}} \mathcal{T}^{(k)}_{\alpha_k\eta_k}(g_k) \right\rangle_{x_{k+1}} \\
&= \langle \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k), g_{k+1}\rangle_{x_{k+1}} - \langle g_k, \eta_k\rangle_{x_k},
\end{aligned}$$

which implies that the denominators in the right-hand sides of (16) and (17) are the same.

Equations (17) and (18) would appear to be a natural generalization of (5). However, y_k, and hence the right-hand side of (17), are not always guaranteed to be well defined, since we cannot ensure that

$$\langle \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\operatorname{grad} f(x_k)), \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k)\rangle_{x_{k+1}} \neq 0 \tag{19}$$

for all $k \geq 0$. On the other hand, as discussed in Sect. 4, the right-hand side of Eq. (16) is well defined when we use step sizes satisfying the weak Wolfe conditions (7) and (8). Defining β_k as (16) has an advantage over (17), because (16) is valid without assumption (19).

Algorithm 3.1 A scaled Dai–Yuan-type Riemannian conjugate gradient method for Problem 2.1 on a Riemannian manifold M

1: Choose an initial point $x_0 \in M$.
2: Set $\eta_0 = -\operatorname{grad} f(x_0)$.
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: Compute the step size $\alpha_k > 0$ satisfying the weak Wolfe conditions (7) and (8) with $0 < c_1 < c_2 < 1$. Set

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

where R is a retraction on M.
5: Set

$$\beta_{k+1} = \frac{\|\operatorname{grad} f(x_{k+1})\|^2_{x_{k+1}}}{\langle \operatorname{grad} f(x_{k+1}), \mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k)\rangle_{x_{k+1}} - \langle \operatorname{grad} f(x_k), \eta_k\rangle_{x_k}}, \tag{20}$$

$$\eta_{k+1} = -\operatorname{grad} f(x_{k+1}) + \beta_{k+1}\mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k), \tag{21}$$

where $\mathcal{T}^{(k)}$ is defined by

$$\mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k) = \begin{cases} \mathcal{T}^{R}_{\alpha_k\eta_k}(\eta_k), & \text{if } \|\mathcal{T}^{R}_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}} \leq \|\eta_k\|_{x_k}, \\ \mathcal{T}^{0}_{\alpha_k\eta_k}(\eta_k), & \text{otherwise,} \end{cases} \tag{22}$$

and where $\mathcal{T}^R$ and $\mathcal{T}^0$ are the differentiated retraction and the associated scaled vector transport defined by (11) and (13), respectively.
6: **end for**

Therefore, our strategy is to define a new β_k using (16), rather than (17). Furthermore, similar to the Fletcher–Reeves-type Riemannian conjugate gradient method proposed in [10], we use the scaled vector transport $\mathcal{T}^0$ associated with the differentiated retraction $\mathcal{T}^R$ only when $\mathcal{T}^R$ increases the norm of the search vector. We now propose a new algorithm as Algorithm 3.1.

Note that (22) is well defined because, when we choose $\mathcal{T}^0$ at the k-th iteration, it holds that $\|\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}} > \|\eta_k\|_{x_k} \geq 0$ and the quantity $\|\eta_k\|_{x_k}/\|\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}}$ is well defined. Furthermore, (22) can also be written as

$$\mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k) = c^{(k)}\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k), \quad c^{(k)} = \min\left\{1, \frac{\|\eta_k\|_{x_k}}{\|\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}}}\right\}. \tag{23}$$

Thus, we obtain the important inequality [10]

$$\|\mathcal{T}^{(k)}_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}} \leq \|\eta_k\|_{x_k}, \tag{24}$$

which is essential in the global convergence analysis of our new algorithm.

4 Global convergence analysis of the proposed new algorithm

In this section, we prove the global convergence property of the proposed algorithm. We first describe our assumption about the objective function f.

Assumption 4.1 The objective function f is bounded below and of C^1-class, and there exists a Lipschitzian constant $L > 0$ such that

$$|\mathrm{D}(f \circ R_x)(t\eta)[\eta] - \mathrm{D}(f \circ R_x)(0)[\eta]| \leq Lt,\ \eta \in T_xM,\ \|\eta\|_x = 1,\ x \in M,\ t \geq 0.$$

Examples of f and the conditions under which this assumption holds are given in [10].

We review a Riemannian analogy of Zoutendijk's theorem. See [9,10] for more details.

Theorem 4.1 *Suppose that a sequence $\{x_k\}$ on a Riemannian manifold M is generated by a general line-search-based optimization algorithm; that is, by Eq. (6) with a retraction R on M. Suppose also that each search direction η_k satisfies $\langle \mathrm{grad}\, f(x_k), \eta_k\rangle_{x_k} < 0$ and that α_k satisfies the weak Wolfe conditions (7) and (8). If Assumption 4.1 is satisfied, then the following series converges:*

$$\sum_{k=0}^{\infty} \cos^2\theta_k \|\mathrm{grad}\, f(x_k)\|^2_{x_k} < \infty,$$

where $\cos\theta_k$ is defined by

$$\cos\theta_k = -\frac{\langle \mathrm{grad}\, f(x_k), \eta_k\rangle_{x_k}}{\|\mathrm{grad}\, f(x_k)\|_{x_k}\|\eta_k\|_{x_k}}.$$

To show that β_k in Algorithm 3.1, and hence the algorithm itself, is well defined, we prove that the search direction η_k is a descent direction; that is, $\langle g_k, \eta_k \rangle_{x_k} < 0$. Note that in [2], the definition (5) of β_k^{DY}, which is a Euclidean analogy of the expression in (17), is fully used to prove that the Euclidean Dai–Yuan-type conjugate gradient method generates descent search directions. In general, however, we cannot use (17), as it is not well defined. Therefore, we can only use (16). Note that if $\langle g_k, \eta_k \rangle_{x_k} \neq 0$ for all $k \geq 0$, (16) is equivalent to (15). In the proof of the next proposition, we use a different approach from that in [2].

Proposition 4.1 *If* $\operatorname{grad} f(x_k) \neq 0$ *for all* $k \geq 0$*, then Algorithm 3.1 is well defined and the following two inequalities hold:*

$$\langle \operatorname{grad} f(x_k), \eta_k \rangle_{x_k} < 0, \tag{25}$$

$$\langle \operatorname{grad} f(x_k), \eta_k \rangle_{x_k} < \langle \operatorname{grad} f(x_{k+1}), \mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k) \rangle_{x_{k+1}}. \tag{26}$$

Proof The proof is by induction. For $k = 0$, the first inequality (25) follows directly from $\eta_0 = -g_0$. We shall prove (26) for $k = 0$. If $\langle g_1, \mathcal{T}^{(0)}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} \geq 0$, it immediately follows that

$$\langle g_1, \mathcal{T}^{(0)}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} \geq 0 > \langle g_0, \eta_0 \rangle_{x_0}. \tag{27}$$

If $\langle g_1, \mathcal{T}^{(0)}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} < 0$, and hence $\langle g_1, \mathcal{T}^{R}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} < 0$, then the second condition (8) of the weak Wolfe conditions with $0 < c_2 < 1$ and (23) gives

$$\begin{aligned} \langle g_1, \mathcal{T}^{(0)}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} &= \min \left\{ 1, \frac{\|\eta_0\|_{x_0}}{\|\mathcal{T}^{R}_{\alpha_0 \eta_0}(\eta_0)\|_{x_1}} \right\} \langle g_1, \mathcal{T}^{R}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} \\ &\geq \langle g_1, \mathcal{T}^{R}_{\alpha_0 \eta_0}(\eta_0) \rangle_{x_1} \geq c_2 \langle g_0, \eta_0 \rangle_{x_0} > \langle g_0, \eta_0 \rangle_{x_0}, \end{aligned} \tag{28}$$

where we have used the fact that (8) is equivalent to (12). Thus, (26) has been proved for $k = 0$, and β_1 is well defined by (20).

Next, suppose that β_k is well defined by the right-hand side of (16), and that both inequalities (25) and (26) hold for some k. Note that β_{k+1} is well defined from the assumption in (26) for k. The left-hand side of (25) for $k+1$ can then be calculated as

$$\begin{aligned} &\langle g_{k+1}, \eta_{k+1} \rangle_{x_{k+1}} = \langle g_{k+1}, -g_{k+1} + \beta_{k+1} \mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k) \rangle_{x_{k+1}} \\ &= - \|g_{k+1}\|^2_{x_{k+1}} + \frac{\|g_{k+1}\|^2_{x_{k+1}}}{\langle g_{k+1}, \mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k) \rangle_{x_{k+1}} - \langle g_k, \eta_k \rangle_{x_k}} \langle g_{k+1}, \mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k) \rangle_{x_{k+1}} \\ &= \frac{\|g_{k+1}\|^2_{x_{k+1}} \langle g_k, \eta_k \rangle_{x_k}}{\langle g_{k+1}, \mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k) \rangle_{x_{k+1}} - \langle g_k, \eta_k \rangle_{x_k}} < 0, \end{aligned}$$

where we have used inequalities (25) and (26) for k. This means that (25) holds for $k+1$. It then follows from a similar argument to (27) and (28) that

$$\langle g_{k+1}, \eta_{k+1} \rangle_{x_{k+1}} < \langle g_{k+2}, \mathcal{T}^{(k+1)}_{\alpha_{k+1} \eta_{k+1}}(\eta_{k+1}) \rangle_{x_{k+2}}.$$

This completes the proof. □

The following corollary immediately follows from (26).

Corollary 4.1 *In Algorithm 3.1, if* $\operatorname{grad} f(x_k) \neq 0$ *for all* $k \geq 0$, *then we have* $\beta_k > 0$ *for all* $k \geq 1$.

We now proceed to our main theorem. The proof is performed in a manner analogous to the Euclidean version in [2] with the aid of a scaled vector transport, as in [10].

Theorem 4.2 *Let* $\{x_k\}$ *be a sequence generated by Algorithm 3.1. If Assumption 4.1 is satisfied, then we have*

$$\liminf_{k\to\infty} \|\operatorname{grad} f(x_k)\|_{x_k} = 0. \tag{29}$$

Proof We first note that, from Proposition 4.1, all the assumptions in Theorem 4.1 are satisfied. Therefore, we have

$$\sum_{k=0}^{\infty} \frac{\langle g_k, \eta_k\rangle_{x_k}^2}{\|\eta_k\|_{x_k}^2} < \infty. \tag{30}$$

If $g_{k_0} = 0$ for some k_0, then $\beta_{k_0} = 0$, followed by $\eta_{k_0} = 0$ and $x_{k_0+1} = R_{x_{k_0}}(0) = x_{k_0}$. Thus, it is sufficient to prove (29) only when $g_k \neq 0$ for all $k \geq 0$. In such a case, it follows from (21) that

$$\eta_{k+1} + g_{k+1} = \beta_{k+1} \mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k).$$

Taking the norms and squaring, we obtain

$$\|\eta_{k+1}\|_{x_{k+1}}^2 = \beta_{k+1}^2 \|\mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k)\|_{x_{k+1}}^2 - 2\langle g_{k+1}, \eta_{k+1}\rangle_{x_{k+1}} - \|g_{k+1}\|_{x_{k+1}}^2.$$

This equality, together with (25), yields

$$\begin{aligned}
\frac{\|\eta_{k+1}\|_{x_{k+1}}^2}{\langle g_{k+1}, \eta_{k+1}\rangle_{x_{k+1}}^2} &= \frac{\|\mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k)\|_{x_{k+1}}^2}{\langle g_k, \eta_k\rangle_{x_k}^2} - \frac{2}{\langle g_{k+1}, \eta_{k+1}\rangle_{x_{k+1}}} - \frac{\|g_{k+1}\|_{x_{k+1}}^2}{\langle g_{k+1}, \eta_{k+1}\rangle_{x_{k+1}}^2} \\
&= \frac{\|\mathcal{T}^{(k)}_{\alpha_k \eta_k}(\eta_k)\|_{x_{k+1}}^2}{\langle g_k, \eta_k\rangle_{x_k}^2} - \left(\frac{1}{\|g_{k+1}\|_{x_{k+1}}} + \frac{\|g_{k+1}\|_{x_{k+1}}}{\langle g_{k+1}, \eta_{k+1}\rangle_{x_{k+1}}}\right)^2 + \frac{1}{\|g_{k+1}\|_{x_{k+1}}^2} \\
&\leq \frac{\|\eta_k\|_{x_k}^2}{\langle g_k, \eta_k\rangle_{x_k}^2} + \frac{1}{\|g_{k+1}\|_{x_{k+1}}^2},
\end{aligned}$$

where we have used the inequality (24) and the fact that β_{k+1} is equal to the right-hand side of (15) for $k+1$, since $\langle g_k, \eta_k\rangle_{x_k} \neq 0$ from (25). We thus arrive at the relation

$$\frac{\|\eta_k\|_{x_k}^2}{\langle g_k, \eta_k\rangle_{x_k}^2} \leq \sum_{i=1}^{k} \frac{1}{\|g_i\|_{x_i}^2} + \frac{\|\eta_0\|_{x_0}^2}{\langle g_0, \eta_0\rangle_{x_0}^2} = \sum_{i=0}^{k} \frac{1}{\|g_i\|_{x_i}^2}. \tag{31}$$

We are now in a position to prove (29) by contradiction. Given that we are now assuming $\operatorname{grad} f(x_k) \neq 0$ for all $k \geq 0$, if (29) does not hold, then there exists a constant $C > 0$ such that

$$\|g_k\|_{x_k} \geq C, \qquad k \geq 0. \tag{32}$$

From (31) and (32), we obtain

$$\frac{\|\eta_k\|_{x_k}^2}{\langle g_k, \eta_k \rangle_{x_k}^2} \leq \frac{k+1}{C^2}.$$

This leads to

$$\sum_{k=0}^{N} \frac{\langle g_k, \eta_k \rangle_{x_k}^2}{\|\eta_k\|_{x_k}^2} \geq C^2 \sum_{k=0}^{N} \frac{1}{k+1} \to \infty, \qquad N \to \infty,$$

which contradicts (30). This proves the theorem. □

5 Numerical experiments

In this section, we describe some numerical experiments designed to show the effectiveness of the proposed algorithm. The experiments are carried out by using MATLAB R2014b on a PC with Intel Core i7-4790 3.60 GHz CPU, 16 GB of RAM memory and Windows 8.1 Pro 64-bit operating system. In the experiments, we determine that a sequence converges to an optimal solution if $\|\operatorname{grad} f(x_k)\| < \varepsilon$ for a predetermined tolerance $\varepsilon > 0$. We refer to β_k in Algorithm 3.1 as β_k^{DY}, and to the Fletcher–Reeves-type β_k, which is discussed in [10], as β_k^{FR}, that is,

$$\beta_{k+1}^{\mathrm{DY}} = \frac{\|g_{k+1}\|_{x_{k+1}}^2}{\langle g_{k+1}, \mathcal{T}_{\alpha_k \eta_k}^{(k)}(\eta_k) \rangle_{x_{k+1}} - \langle g_k, \eta_k \rangle_{x_k}}, \quad \beta_{k+1}^{\mathrm{FR}} = \frac{\|g_{k+1}\|_{x_{k+1}}^2}{\|g_k\|_{x_k}^2}.$$

5.1 Line search algorithms

Let ϕ_k denote a one-variable function defined by $\phi_k(\alpha) = f(R_{x_k}(\alpha \eta_k))$, where η_k is a search direction at $x_k \in M$. Then, the Wolfe conditions (7) and (8) can be rewritten as

$$\phi_k(\alpha_k) \leq \phi_k(0) + c_1 \phi_k'(0), \tag{33}$$
$$\phi_k'(\alpha_k) \geq c_2 \phi_k'(0). \tag{34}$$

In terms of finding some α_k that satisfies (33) and (34), the problem setting is the same as that in Euclidean optimization problems. Therefore, we can use existing algorithms such as those in [7] and [8]. Note that (34) is replaced by

$$|\phi_k'(\alpha_k)| \leq c_2 |\phi_k'(0)|$$

in the strong Wolfe conditions.

In the following numerical experiments, unless otherwise noted, we set $c_1 = 10^{-4}$ and $c_2 = 0.1$ and use line search algorithms for the weak and strong Wolfe conditions based on [7] and [8], respectively. See Fig. 1 in [7] and Algorithms 3.5 and 3.6 in [8] for further details.

In the algorithm in [7], we must interpolate a lower bound t_{L} and an upper bound t_{R} of an appropriate step size to find t_{interpol}. Furthermore, when no upper bound is available, we have to extrapolate the lower bound t_{L} to find t_{extrapol}. To this end, we use the simple formula

$$t_{\mathrm{interpol}} := \frac{t_{\mathrm{L}} + t_{\mathrm{R}}}{2}, \quad t_{\mathrm{extrapol}} := 2t_{\mathrm{L}}.$$

In Algorithm 3.5 in [8], we have to extrapolate an estimated step size $\alpha^{(i)}$ to determine a value $\alpha^{(i+1)}$, where the superscript is the inner iteration number of Algorithm 3.5 in [8]. We use the formula introduced in [8]:

$$\alpha := \alpha^{(i)} - (\alpha^{(i)} - \alpha^{(i-1)})\frac{\phi'(\alpha^{(i)}) + d_2 - d_1}{\phi'(\alpha^{(i)}) - \phi'(\alpha^{(i-1)}) + 2d_2}, \tag{35}$$

with

$$\begin{aligned} d_1 &= \phi'(\alpha^{(i-1)}) + \phi'(\alpha^{(i)}) - 3\frac{\phi(\alpha^{(i-1)}) - \phi(\alpha^{(i)})}{\alpha^{(i-1)} - \alpha^{(i)}}, \\ d_2 &= \mathrm{sign}(\alpha^{(i)} - \alpha^{(i-1)})\sqrt{d_1^2 - \phi'(\alpha^{(i-1)})\phi'(\alpha^{(i)})}, \end{aligned}$$

where sign is the sign function. This formula is based on the cubic interpolation of $\alpha^{(i-1)}$ and $\alpha^{(i)}$. Note that the α obtained in (35) may be too small or too large. To safeguard this, we use a rule introduced in [4], that is, we choose $\alpha^{(i+1)}$ from the interval $[2\alpha^{(i)} - \alpha^{(i-1)}, \alpha^{(i)} + 9(\alpha^{(i)} - \alpha^{(i-1)})]$. Together with Eq. (35), this rule gives the formula

$$\alpha^{(i+1)} := \min\{\max\{\alpha, 2\alpha^{(i)} - \alpha^{(i-1)}\}, \alpha^{(i)} + 9(\alpha^{(i)} - \alpha^{(i-1)})\}. \tag{36}$$

We implement Algorithms 3.5 and 3.6 in [8] using (36).

5.2 Rayleigh quotient minimization on the sphere

Let A be an $n \times n$ symmetric matrix. Throughout this section, we consider the problem of minimizing

$$f(x) = x^T Ax$$

on the unit sphere $S^{n-1} = \{x \in \mathbb{R}^n \mid x^T x = 1\}$ endowed with the natural Riemannian metric

$$\langle \xi, \eta \rangle_x = \xi^T \eta, \quad \xi, \eta \in T_x S^{n-1}, \ x \in S^{n-1},$$

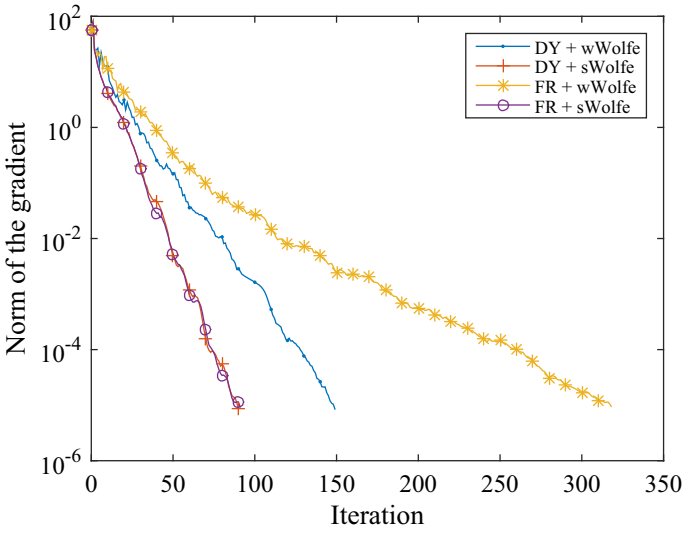

Fig. 1 The sequence of $\|\mathrm{grad}\, f(x_k)\|_{x_k}$ evaluated on the sequence $\{x_k\}$ generated by Algorithm 3.1 with $n = 100$

in which case the gradient of the objective function f is written as

$$\mathrm{grad}\, f(x) = 2(I_n - xx^T)Ax.$$

For simplicity, we use a retraction R defined by

$$R_x(\xi) = \frac{x+\xi}{\|x+\xi\|}, \quad \xi \in T_x S^{n-1},\ x \in S^{n-1},$$

where $\|\cdot\|$ denotes the Euclidean norm, and use the associated differentiated retraction [1]

$$\begin{aligned}\mathcal{T}_\eta(\xi) &= \mathrm{D}R_x(\eta)[\xi] \\ &= \frac{1}{\|x+\eta\|}\left(I_n - \frac{1}{\|x+\eta\|^2}(x+\eta)(x+\eta)^T\right)\xi, \quad \eta, \xi \in T_x S^{n-1},\ x \in S^{n-1}.\end{aligned}$$

For means of reproducibility, we first set $A = \mathrm{diag}(1, 2, \ldots, n)$, $x_0 = \mathbf{1}_n/\sqrt{n}$, and $\varepsilon = 10^{-5}$ without using random values, where $\mathbf{1}_n$ is an n-dimensional vector whose elements are all equal to 1. We then apply conjugate gradient methods with β^{FR} and β^{DY} under the weak and strong Wolfe conditions, and with $n = 100$ and $n = 500$. Figures 1 (for $n = 100$) and 2 (for $n = 500$) show the results of the experiments. In Fig. 2, the graph corresponding to the method with β^{FR} and the weak Wolfe conditions implies that this method is much slower than the others. In fact, with some other initial points, no appropriate step size satisfying the weak Wolfe conditions can be found for this method. For example, with $x_0 = (\mathbf{1}_{35}^T \ \mathbf{0}_{465}^T)^T/\sqrt{35}$, we have

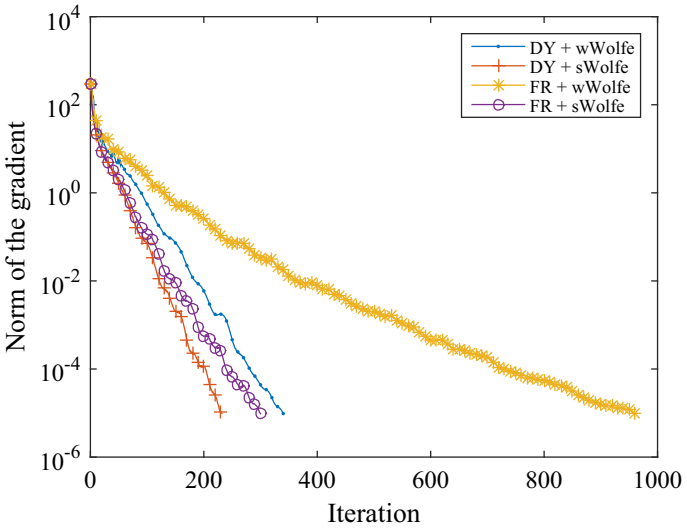

Fig. 2 The sequence of $\|\mathrm{grad}\, f(x_k)\|_{x_k}$ evaluated on the sequence $\{x_k\}$ generated by Algorithm 3.1 with $n = 500$

Table 1 Comparison of several quantities between different conjugate gradient methods with $n = 100$

Method	Iterations	Function evals.	Gradient evals.	Computational time
DY + wWolfe	149	210	206	0.0175
DY + sWolfe	90	288	244	0.0187
FR + wWolfe	318	619	577	0.0429
FR + sWolfe	91	293	258	0.0191

$\langle \mathrm{grad}\, f(x_k), \eta_k \rangle_{x_k} = 1.2646 \times 10^{-4} > 0$ at $k = 37$, which means that the search direction η_k is not a descent direction for f at x_k. In contrast, under Proposition 4.1, the method with β^{DY} and the weak Wolfe conditions is guaranteed to have the property $\langle \mathrm{grad}\, f(x_k), \eta_k \rangle_{x_k} < 0$ for all $k \geq 0$. The number of iterations, function evaluations, and gradient evaluations, as well as the average computational time (1000 times) required for convergence, are shown in Tables 1 (for $n = 100$) and 2 (for $n = 500$). Note that we have abbreviated the weak and strong Wolfe conditions as wWolfe and sWolfe, respectively.

These results indicate that we should not use the Fletcher–Reeves-type method with the weak Wolfe conditions, because it is not always guaranteed to generate a converging sequence. In Tables 1 and 2, we can observe that "DY + wWolfe" requires the shortest computational time. In the following, we compare three methods, namely "DY + wWolfe", "DY + sWolfe", and "FR + sWolfe", using more general 1000 problems.

We next set $n = 100$, generate 1000 symmetric matrices A and initial points x_0 with randomly chosen elements, and then apply the above three methods. Table 3 shows the average values of the quantities in Tables 1 and 2 over the 1000 matrices A.

Table 2 Comparison of several quantities between different conjugate gradient methods with $n = 500$

Method	Iterations	Function evals.	Gradient evals.	Computational time
DY + wWolfe	340	373	367	0.0522
DY + sWolfe	232	657	467	0.0658
FR + wWolfe	960	1902	1757	0.1988
FR + sWolfe	300	723	529	0.0730

Table 3 Comparison of the averages of several quantities between different conjugate gradient methods with 1000 randomly chosen matrices A

Method	Iterations	Function evals.	Gradient evals.	Computational time
DY + wWolfe	242.751	538.177	469.628	0.0334
DY + sWolfe	160.270	529.736	410.278	0.0322
FR + sWolfe	201.441	649.900	513.879	0.0390

Even though "DY + wWolfe" requires the most iterations, it generates fewer function and gradient evaluations per iteration. In our experiments, "DY + sWolfe" had the shortest average computational time. In fact, however, "DY + sWolfe" was not the fastest method for all 1000 matrices, as "DY + wWolfe" converged faster with 379 of the 1000 matrices. Furthermore, we can observe that the Dai–Yuan-type Riemannian conjugate gradient method is more efficient than the Fletcher–Reeves-type method.

We also perform the same experiments as in Table 3 with other choices of c_1 and c_2 in the (strong) Wolfe conditions: $(c_1, c_2) = (10^{-4}, 10^{-2}), (10^{-4}, 0.2), (10^{-4}, 0.3), (10^{-4}, 0.4), (0.1, 0.2), (0.1, 0.3), (0.1, 0.4)$. The results show that the choice of c_1 does not affect the performance of the proposed method very much, while the choice of c_2 can improve or worsen the performance. In fact, in our experiments, the proposed method with $(c_1, c_2) = (0.1, 0.4)$ was the fastest. The three methods "DY + wWolfe", "DY + sWolfe", and "FR + sWolfe" with $(c_1, c_2) = (0.1, 0.4)$ required 0.0255, 0.0310, and 0.0465 seconds (on average) for convergence, respectively. In this case, "DY + wWolfe" converged faster than "DY + sWolfe" with 842 of the 1000 matrices. Even though a better choice of c_1 and c_2 can improve conjugate gradient methods, we observe that the Dai–Yuan-type method is always more efficient than the Fletcher–Reeves-type method.

6 Concluding remarks

We have proposed a new Riemannian conjugate gradient method based on the Dai–Yuan-type Euclidean conjugate gradient algorithm. To generalize $\beta_k^{\mathrm{DY}} = \nabla f(x_k)^T \nabla f(x_k) / \eta_{k-1}^T y_k$ in the Dai–Yuan algorithm, we used another expression $\beta_k^{\mathrm{DY}} = \nabla f(x_k)^T \eta_k / \nabla f(x_{k-1})^T \eta_{k-1}$ and the notion of a vector transport. We thus proposed a new β_k, and hence a new Riemannian conjugate gradient method.

We have proved that the proposed algorithm is well defined and that sequences generated by the algorithm are globally convergent. The notion of a scaled vector transport plays an essential role in the convergence analysis, that is, property (24) leads to a contradiction with Zoutendijk's theorem if we assume that the assertion of Theorem 4.2 is false. One advantage of our algorithm is that we do not have to search for step sizes satisfying the strong Wolfe conditions, which are necessary conditions for the existing Fletcher–Reeves-type algorithm, and need only compute the step sizes with the weak Wolfe conditions.

Through a series of numerical experiments, we demonstrated that the Dai–Yuan-type method is better than the Fletcher–Reeves-type approach. Implementing the Dai–Yuan-type method with the weak Wolfe conditions is not always better than that with the strong Wolfe conditions, but is preferable for many problems. Nevertheless, we can choose which conditions we use according to the form of the objective function, the size of the problem, and so on.

It is also worth pointing out that the role of the scaled vector transport in Algorithm 3.1 may be imposed on β_k. Putting the scaling factor $\min\left\{1, \|\eta_k\|_{x_k}/\|\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}}\right\}$ on β_{k+1}, we can rewrite the computations in Step 5 of Algorithm 3.1 as

$$
\begin{aligned}
c^{(k)} &= \min\left\{1, \frac{\|\eta_k\|_{x_k}}{\|\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\|_{x_{k+1}}}\right\},\\
\bar{\beta}_{k+1} &= \frac{c^{(k)}\|\operatorname{grad} f(x_{k+1})\|^2_{x_{k+1}}}{c^{(k)}\langle \operatorname{grad} f(x_{k+1}), \mathcal{T}^R_{\alpha_k\eta_k}(\eta_k)\rangle_{x_{k+1}} - \langle \operatorname{grad} f(x_k), \eta_k\rangle_{x_k}},\\
\eta_{k+1} &= -\operatorname{grad} f(x_{k+1}) + \bar{\beta}_{k+1}\mathcal{T}^R_{\alpha_k\eta_k}(\eta_k).
\end{aligned}
$$

Although these formulas stem from the notion of a scaled vector transport, they can be regarded as if we are not using a scaled vector transport, but are simply applying another type of β_k with a usual vector transport. These may be easier to deal with in the common framework of the conjugate gradient method discussed in [1].

Acknowledgments The author would like to thank the anonymous referees for their valuable comments that helped improve the paper significantly. This work was supported by JSPS (Japan Society for the Promotion of Science) KAKENHI Grant Number 26887037.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton (2008)
2. Dai, Y.H., Yuan, Y.: A nonlinear conjugate gradient method with a strong global convergence property. SIAM J. Optim. **10**(1), 177–182 (1999)
3. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. **20**(2), 303–353 (1998)
4. Fletcher, R.: Practical Methods of Optimization. Wiley, New York (2013)
5. Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. **7**(2), 149–154 (1964)
6. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952)

7. Lemaréchal, C.: A view of line-searches. Optimization and Optimal Control, pp. 59–78. Springer, Berlin (1981)
8. Nocedal, J., Wright, S.: Numerical Optimization. Series in Operations Research and Financial Engineering. Springer, New York (2006)
9. Ring, W., Wirth, B.: Optimization methods on Riemannian manifolds and their application to shape space. SIAM J. Optim. **22**(2), 596–627 (2012)
10. Sato, H., Iwai, T.: A new, globally convergent Riemannian conjugate gradient method. Optimization **64**(4), 1011–1031 (2015)