An Adaptive Riemannian Gradient Method Without Function Evaluations

Geovani N. Grapiglia* & Gabriel F.D. Stella†

April 06, 2022

Abstract

In this paper we propose an adaptive gradient method for optimization on Riemannian manifolds. The update rule for the stepsizes relies only on gradient evaluations. Assuming that the objective function is bounded from below and that its gradient field is Lipschitz continuous, we establish worst-case complexity bounds for the number of gradient evaluations that the method requires to generate approximate an stationary point. Preliminary numerical results illustrate the potential advantages of different versions of our method in comparison with a Riemannian gradient method with Armijo line search.

Keywords: Riemannian Optimization; Gradient Method; Worst-Case Complexity Bounds

1 Introduction

The gradient method is one of the classical iterative schemes for unconstrained smooth optimization problems [6]. Given a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, and a starting point $x_0 \in \mathbb{R}^n$, a sequence of iterates $\{x_k\}_{k\geq 0}$ is generated by the rule

$$x_{k+1} = x_k - t_k \nabla f(x_k), \quad k \geq 0,$$

where $t_k > 0$. To obtain global convergence guarantees, the choice of the stepsizes $\{t_k\}_{k\geq 0}$ is crucial. When the gradient of the objective function is L-Lipschitz continuous, the constant stepsize $t_k = 1/L$ $(k \geq 0)$ yields a globally convergent gradient method, in the sense that for any starting point x_0 and any $\epsilon > 0$, there exists $T \in \mathbb{N}$ such that $\|\nabla f(x_{T+1})\| \leq \epsilon$. However, more efficient gradient methods can be obtained by allowing the use of a different stepsize at each iteration. Traditionally, adaptive selection of stepsizes is implemented by using line search procedures [2, 11, 16, 13]. In this approach, the selection of a suitable stepsizes is done at the expense of evaluations of the objective function. For example, the Armijo line search [2] selects $t_k = 2^{-i_k}$ where $i_k \geq 0$ is the smallest integer such that

$$f(x_k) - f(x_k - 2^{-i_k}\nabla f(x_k)) \geq 2^{-(i_k+1)}\|\nabla f(x_k)\|^2.$$

*Université catholique de Louvain, ICTEAM/INMA, Avenue Georges Lemaître, 4-6/ L4.05.01, B-1348, Louvain-la-Neuve, Belgium (geovani.grapiglia@uclouvain.be).

†Programa de Pós-Graduação em Matemática, Universidade Federal do Paraná, Centro Politécnico, Cx. postal 19.081, 8153I1-980, Curitiba, Paraná, Brazil (gabrielstella28@gmail.com).

Recently, adaptive stepsize rules based only on gradient evaluations have been proposed in the context of Euclidean optimization [14, 15]. In particular, the method WNGrad [15] is a gradient method that uses $t_k = 1/b_k$ with the update rule

$$b_{k+1} = b_k + \frac{\|\nabla f(x_{k+1})\|^2}{b_k}, \quad k \geq 0.$$

Remarkably, it was shown in [15] that WNGrad performs at most $\mathcal{O}\left(\epsilon^{-2}\right)$ iterations to generate x_k such that $\|\nabla f(x_k)\| \leq \epsilon$.

Inspired by WNGrad, in this paper we propose an adaptive gradient method without function evaluations for optimization problems on Riemannian manifolds. Our general scheme admits a *conservative version* and a *flexible version*. The conservative version is a generalization of WNGrad in which $\{t_k\}_{k\geq 0}$ is monotonically decreasing. On the other hand, the flexible version allows the selection of $t_{k+1} > t_k$ when suitable conditions are satisfied. Assuming that the gradient field of the objective function is Lipschitz continuous, we show that the conservative version needs at most $\mathcal{O}\left(\epsilon^{-2}\right)$ gradient evaluations to generate an ϵ-approximate stationary point, while the flexible version needs at most $\mathcal{O}\left(|\log(\epsilon)|\epsilon^{-2}\right)$ gradient evaluations to achieve the same goal. We also report preliminary numerical results with some test problems on the manifold of symmetric positive definite matrices. They illustrate the potential advantages of our schemes in comparison with a Riemannian gradient method with Armijo line search.

The paper is organized as follows. In Section 2 we present the notation and some fundamental inequalities. In Section 3 we describe the general algorithm and present the worst-case complexity analysis for both versions of the algorithm. Finally, in Section 4 we report the numerical results.

2 Notation and Key Inequalities

Let M be a Riemannian manifold. The tangent space of M at a point p is denoted by T_pM, while $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ denote the Riemannian metric and the Riemannian norm, respectively. The symbol $\ell(\gamma)$ denotes the length of a piecewise smooth curve $\gamma : [a, b] \to M$. Let ∇ be the Levi-Civita connection of the Riemannian manifold $(M, \langle\cdot,\cdot\rangle)$. Given a piecewise smooth curve $\gamma : [a, b] \to M$ and $t \in [a, b]$, ∇ induces an isometry with respect to $\langle\cdot,\cdot\rangle$, $P_{\gamma,a,t} : T_{\gamma(a)}M \to T_{\gamma(t)}M$ such that $P_{\gamma,a,t}v = V(t)$, where V is the unique vector field along the curve γ satisfying $\nabla_{\dot\gamma(t)}V(t) = 0$ and $V(a) = v$. This isometry $P_{\gamma,a,t}$ is called parallel transport (along γ) joining $\gamma(a)$ and $\gamma(t)$. We say that a vector field V (along γ) is parallel when $\nabla_{\dot\gamma}V \equiv 0$. If $\dot\gamma$ is parallel to itself, it is called a geodesic.

The Riemannian distance between any two points $p, q \in M$, connected by a piecewise smooth curve in M, is defined by

$$d(p, q) = \inf\{\ell(\gamma)|\, \gamma \text{ is a piecewise smooth curve connecting } p \text{ and } q\}, \tag{1}$$

A geodesic γ, connecting p and q, is said to be minimal when $\ell(\gamma) = d(p, q)$. For each $p \in M$ the exponential map at p, defined in an open $U \subset T_pM$, is the function $\exp_p : U \to M$ such that $\exp_p(v) = \gamma(1)$ and γ is a geodesic defined in an interval containing $[0, 1]$ with $\gamma(0) = p$ and $\dot\gamma(0) = v$. We say that $(M, \langle\cdot,\cdot\rangle)$ is complete when for any $p \in M$, the map $\exp_p$ is defined for any $v \in T_pM$.

In this paper, we assume that M is manifold connected and complete. Let $f : U \to \mathbb{R}$ be a differentiable function with $U \subset M$ being an open set. The gradient of f, denoted by $\operatorname{grad} f$, is the vector field defined in U such that $\langle \operatorname{grad} f(p), v\rangle = df_p(v)$ for any $p \in M$ and $v \in T_pM$. We say that

the gradient vector field of f is L-Lipschitz [12] when, for any $p, q \in U$ and a geodesic $\gamma : [a, b] \to U$ with $\gamma(a) = p$ and $\gamma(b) = q$, we have

$$\|P_{\gamma,a,b}\text{grad}\, f(p) - \text{grad}\, f(q)\| \leq L\ell(\gamma).$$

Our analysis will be based on the following inequalities.

Lemma 2.1. *Let $f : M \to \mathbb{R}$ be a differentiable function such that its gradient vector field is L-Lipschitz continuous. Then*

$$f\left(\exp_p(v)\right) \leq f(p) + \langle \text{grad}\, f(p), v\rangle + \frac{L}{2}\|v\|^2, \quad p \in M,\ v \in T_pM. \tag{2}$$

Consequently,

$$f\left(\exp_p(-t\text{grad}\, f(p)\right) \leq f(p) - \left(t - \frac{L}{2}t^2\right)\|\text{grad}\, f(p)\|^2, \quad t \in \mathbb{R},\ p \in M. \tag{3}$$

Proof. See, e.g., Lemma 2.1 and Corollary 2.1 in [3]. □

3 Algorithm and Its Worst-Case Complexity Analysis

Let M be a Riemannian manifold. Consider the following optimization problem

$$\min_{x\in M} f(x), \tag{4}$$

where $f : M \to \mathbb{R}$ is continuously differentiable. Specifically, we will asume that:

A1. The gradient field of f is L-Lipschitz.

A2. There exists $f_{low} \in \mathbb{R}$ such that $f(x) \geq f_{low}$ for all $x \in M$.

In what follows we describe the main steps of our new adaptive algorithm for problem (4).

Algorithm 1. Adaptive Riemannian Gradient Method

Step 0. Given $x_0 \in M$, and parameters $\hat{c} \in \mathbb{N} \setminus \{0\}$, $b_{\min} > 0$, $b_0 \geq b_{\min}$ and $\alpha \in [0, 1)$, set $t_0 = 1/b_0$, $\omega_0 = \|\text{grad}\, f(x_0)\|$, $c_0 = 0$, and $k := 0$.

Step 1. Compute

$$x_{k+1} = \exp_{x_k}\left(-t_k\text{grad}\, f(x_k)\right). \tag{5}$$

Step 2. If $\|\text{grad}\, f(x_{k+1})\| \leq \alpha\omega_k$, set

$$(b_{k+1}, \omega_{k+1}, c_{k+1}) = \begin{cases} \left(\hat{b}_k, \|\text{grad}\, f(x_{k+1})\|, 0\right), & \text{if } c_k + 1 = \hat{c}, \\ \left(b_k + \dfrac{\|\text{grad}\, f(x_{k+1})\|^2}{b_k}, \omega_k, c_k + 1\right), & \text{if } c_k + 1 < \hat{c}, \end{cases} \tag{6}$$

where

$$\hat{b}_k \in [b_{\min}, b_k]. \tag{7}$$

Otherwise, set

$$(b_{k+1}, \omega_{k+1}, c_{k+1}) = \left(b_k + \frac{\|\text{grad}\, f(x_{k+1})\|^2}{b_k}, \omega_k, 0 \right). \tag{8}$$

Step 3. Define $t_{k+1} = 1/b_{k+1}$, set $k := k + 1$ and go to Step 1.

Remark 3.1. *When $\alpha = 0$, Algorithm 1 reduces to a Riemannian version of WNGrad [15], in which $t_k = 1/b_k$ with*

$$b_{k+1} = b_k + \frac{\|\text{grad}\, f(x_{k+1})\|^2}{b_k}, \quad \forall k \geq 0.$$

Notice that in this case $\{b_k\}_{k\geq 0}$ is monotonically increasing, which implies that $\{t_k\}_{k\geq 0}$ is monotonically decreasing. This version of Algorithm 1 is called the conservative version. On the other hand, when $\alpha \neq 0$ and $\hat{b}_k < b_k$, rule (6) allows the selection of $b_{k+1} < b_k$, which yields $t_{k+1} > t_k$. For this reason, the corresponding scheme is called the flexible version of Algorithm 1. The update rules (6)-(8) are inspired by similar rules recently proposed in the context of a trust-region method without function evaluations for Euclidean optimization [9]. The main difference is that here the selection of $b_{k+1} < b_k$ is done in a more cautious way: it happens only when the condition $\|\text{grad}\, f(x_{k+1})\| \leq \alpha\omega_k$ is verified during $\hat{c}$ consecutive iterations.

Given $\epsilon \in (0, 1)$, assume that

$$\|\text{grad}\, f(x_k)\| > \epsilon \quad \text{for} \quad k = 0, \ldots, T. \tag{9}$$

The goal of our analysis is to obtain an upper bound for T. Let us define the set

$$D_T = \{1 \leq k \leq T \,:\, \|\text{grad}\, f(x_k)\| \leq \alpha\omega_{k-1} \text{ and } c_{k-1} + 1 = \hat{c}\}. \tag{10}$$

Moreover, given $a, b \in \mathbb{N}$ with $a \leq b$, denote

$$I(a, b) = \{k \in \mathbb{N} \,:\, a \leq k \leq b\}. \tag{11}$$

If $D_T \neq \emptyset$, we can write

$$D_T = \left\{k_1, \ldots, k_{|D_T|}\right\}$$

where $k_i < k_j$ whenever $i < j$, and $|D_T|$ is the cardinality of D_T. Then, defining $k_0 = 0$ and $k_{|D_T|+1} = T + 1$, it follows that

$$\begin{aligned} T + 1 &= |\{0, 1, \ldots, T\}| = \left| \cup_{i=0}^{|D_T|} I(k_i, k_{i+1} - 1) \right| \\ &\leq \sum_{i=0}^{|D_T|} \max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)| \\ &= (|D_T| + 1) \max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)|. \end{aligned} \tag{12}$$

Notice that when $D_T = \emptyset$, we have

$$T + 1 = |I(k_0, k_{|D_T|+1} - 1)|,$$

and so inequality (12) also holds. In view of (12), to obtain an upper bound for T it is enough to get an upper bound for $|D_T|$ and an upper bound for

$$\max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)|.$$

Let us start analysing $|D_T|$.

Lemma 3.2. *Let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 for which (9) holds. Then*

$$|D_T| = 0 \quad \textit{if} \quad \alpha = 0, \tag{13}$$

and

$$|D_T| < \frac{\log\left(\|\text{grad}\, f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|} \quad \textit{if} \quad \alpha \in (0,1). \tag{14}$$

Proof. Suppose that $\alpha = 0$. Then, by (9) and (10) we have

$$D_T = \{1 \leq k \leq T \,:\, \|\text{grad}\, f(x_k)\| \leq 0 \text{ and } c_k = \hat{c}\} = \emptyset.$$

Thus, (13) is true. Suppose that $\alpha \in (0,1)$. If $D_T = \emptyset$ then

$$|D_T| = 0 < \frac{\log\left(\|\text{grad}\, f(x_0)\|\epsilon^{-1}\right)}{|\log(\alpha)|},$$

that is, (14) holds. From now on let us assume that $D_T \neq \emptyset$. By (8) and $\omega_0 = \|\text{grad}\, f(x_0)\|$, we have

$$\omega_k = \omega_0 = \|\text{grad}\, f(x_0)\| \quad \text{for } k = 0, \ldots, k_1 - 1, \tag{15}$$

$$\omega_k = \omega_{k_i} \quad \text{for } k = k_i, \ldots, k_{i+1} - 1, \tag{16}$$

and

$$\omega_{k_i} = \|\text{grad}\, f(x_{k_i})\| \leq \alpha\omega_{k_i - 1} \quad \text{for } i = 1, \ldots, |D_\ell|. \tag{17}$$

Combining (15)-(17), we obtain

$$\|\text{grad}\, f(x_{k_i})\| \leq \alpha^i \|\text{grad}\, f(x_0)\| \quad \text{for } i = 1, \ldots, |D_T|. \tag{18}$$

In particular, by (9) we have

$$\epsilon < \|\text{grad}\, f(x_{k_{|D_T|}})\| \leq \alpha^{|D_T|}\|\text{grad}\, f(x_0)\|,$$

which implies that

$$|D_T| < \frac{\log\left(\|\text{grad}\, f(x_0)\|\epsilon^{-1}\right)}{|\log(\epsilon)|}.$$

□

Now, we will focus of the derivation of an upper bound to

$$\max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1} - 1)|.$$

For that, we will need several auxiliary results. The next lemma gives a lower bound for $f(x_k) - f(x_{k+1})$ when $b_k \geq L$.

Lemma 3.3. *Suppose that A1 hold. If*

$$b_k \geq L, \tag{19}$$

then

$$f(x_k) - f(x_{k+1}) \geq \frac{\|\text{grad}\, f(x_k)\|^2}{2b_k}. \tag{20}$$

Proof. Using (5) and $t_k = 1/b_k$, it follows from inequality (3) in Lemma 2.1 with $p = x_k$ and $t = t_k$. □

The following two lemmas give upper bounds for the cardinality of some subsets of $I(k_i, k_{i+1}-1)$.

Lemma 3.4. *Suppose that A1 is true, and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that (9) holds. Given $i \in \{0, \ldots, |D_T|\}$ and $q \in I(k_i, k_{i+1} - 1)$, if*

$$b_j < L \quad \text{for all} \quad j \in I(k_i, q), \tag{21}$$

then

$$|I(k_i, q)| \leq L^2\epsilon^{-2} + 2. \tag{22}$$

Proof. We have $k_i \leq q$. If $k_i \geq q - 1$, then $|I(k_i, q)| \leq 2$ and (22) holds. Assume that $k_i < q - 1$. Since $I(k_i, q) \subset I(k_i, k_{i+1} - 1)$, it follows from (6) that

$$b_{j+1} - b_j = \frac{\|\text{grad}\, f(x_{j+1})\|^2}{b_j} \quad \text{for} \quad j = k_i, \ldots, q - 1.$$

Summing up these inequalities, it follows from (9) an (21) that

$$L > b_q - b_{k_i} = \sum_{k \in I(k_i, q-1)} \frac{\|\text{grad}\, f(x_{k+1})\|^2}{b_k} > |I(k_i, q-1)|L^{-1}\epsilon^2.$$

Therefore

$$|I(k_i, q)| = |I(k_i, q-1)| + 1 < L^2\epsilon^{-2} + 1,$$

and so (22) also holds in this case. □

Lemma 3.5. *Suppose that A1 and A2 are true, and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that (9) holds. Moreover, given $i \in \{0, \ldots, |D_T|\}$, assume*

$$\mathcal{U} = \{j \in I(k_i, k_{i+1} - 1) \,:\, b_j \geq L\} \neq \emptyset, \tag{23}$$

and let p be the smallest element of $\mathcal{U}$. If $k_i + 2 < k_{i+1} - 2$, and $p \in I(k_i, k_{i+1} - 2)$, then

$$|I(p, k_{i+1} - 1)| \leq A_i\epsilon^{-2}. \tag{24}$$

where

$$A_i = 4\left[\left(b_{k_i} + \frac{2\|\text{grad}\, f(x_{k_i})\|^2}{b_{k_i}} + 3L + \frac{2L^2}{b_{\min}}\right)\left(1 + \frac{2L}{b_{\min}}\right) + 4(f(x_{k_i}) - f_{low})\right]^2 \left(\frac{L^2}{b_{\min}^2} + 1\right)^2 \tag{25}$$

Proof. By (8) we have

$$b_j \geq L \quad \text{for } j = p, \ldots, k_{i+1} - 1.$$

Then, it follows from Lemma 3.3 that

$$f(x_j) - f(x_{j+1}) \geq \frac{\|\text{grad}\, f(x_j)\|^2}{2b_j} \quad \text{for all} \quad j = p, \ldots, k - 1.$$

Summing up these inequalities, and using A2, (9) and (8) we get

$$\begin{aligned} f(x_p) - f_{low} &\geq f(x_p) - f(x_{k_{i+1}}) \\ &\geq \sum_{j=p}^{k_{i+1}-1} \frac{\|\mathrm{grad}\, f(x_j)\|^2}{2b_j} \\ &\geq \frac{|I(p, k_{i+1}-1)|\epsilon^2}{b_{k_{i+1}-1}}. \end{aligned} \tag{26}$$

Therefore,

$$|I(p, k_{i+1}-1)| \leq 2(f(x_p) - f_{low}) b_{k_{i+1}-1} \epsilon^{-2}. \tag{27}$$

Given $j \in I(k_i, k_{i+1}-2)$, denote $P_j = P_{\gamma_j,0,1}$, where $\gamma_j(s) = \exp(-st_j \mathrm{grad}\, f(x_j))$. Then, by (8), A1 and the isometry property of P_j we have

$$\begin{aligned} b_{j+1} - b_j &= \frac{\|\mathrm{grad}\, f(x_{j+1})\|^2}{b_j} \\ &\leq \frac{1}{b_j}(\|\mathrm{grad}\, f(x_{j+1}) - P_j \mathrm{grad}\, f(x_j)\| + \|P_j \mathrm{grad}\, f(x_j)\|)^2 \\ &\leq \frac{2}{b_j}(\|\mathrm{grad}\, f(x_{j+1}) - P_j \mathrm{grad}\, f(x_j)\|^2 + \|P_j \mathrm{grad}\, f(x_j)\|^2) \\ &\leq \frac{2}{b_j}\left(L^2 \frac{\|\mathrm{grad}\, f(x_j)\|^2}{b_j^2} + \|\mathrm{grad}\, f(x_j)\|^2\right) \\ &\leq \frac{2\|\mathrm{grad}\, f(x_j)\|^2}{b_j}\left(\frac{L^2}{b_{\min}^2} + 1\right), \end{aligned} \tag{28}$$

where the last inequality follows from the fact that $b_j \geq b_{\min}$. Then, in view of (8), A2 and (26), it follows that

$$\begin{aligned} b_{k_{i+1}-1} - b_p &= \sum_{j=p}^{k_{i+1}-2} b_{j+1} - b_j \\ &\leq 2\left(\sum_{j=p}^{k_{i+1}-2} \frac{\|\mathrm{grad}\, f(x_j)\|^2}{b_j}\right)\left(\frac{L^2}{b_{\min}^2} + 1\right) \\ &\leq 4\left(f(x_p) - f_{low}\right)\left(\frac{L^2}{b_{\min}^2} + 1\right), \end{aligned}$$

which gives

$$b_{k_{i+1}-1} \leq b_p + 4\left(f(x_p) - f_{low}\right)\left(\frac{L^2}{b_{\min}^2} + 1\right). \tag{29}$$

Now, we can divide the proof in three cases.

Case I: $p = k_i$.

In this case, it follows directly from (27) and (29) that

$$|I(p, k_{i+1}-1)| \leq 2\left(f(x_{k_i}) - f_{low}\right)\left[b_{k_i} + 4\left(f(x_{k_i}) - f_{low}\right)\left(\frac{L^2}{b_{\min}^2} + 1\right)\right]\epsilon^{-2},$$

and so (24) holds.

Case II: $p = k_i + 1$.

In this case, by Lemma 2.1 we have

$$\begin{aligned} f(x_p) &\leq f(x_{k_i}) + \left\langle \operatorname{grad} f(x_{k_i}), -\frac{\operatorname{grad} f(x_{k_i})}{b_j} \right\rangle + \frac{L}{2}\frac{\|\operatorname{grad} f(x_j)\|^2}{b_j^2} \\ &\leq f(x_{k_i}) + \frac{L}{2b_{\min}^2}\|\operatorname{grad} f(x_j)\|^2. \end{aligned} \tag{30}$$

Moreover, it follows from (28) that

$$\begin{aligned} b_p &\leq b_{k_i} + \frac{2\|\operatorname{grad} f(x_{k_i})\|^2}{b_{k_i}}\left(\frac{L^2}{b_{\min}^2} + 1\right) \\ &\leq b_{k_i} + \frac{2\|\operatorname{grad} f(x_{k_i})\|^2}{b_{\min}}\left(\frac{L^2}{b_{\min}^2} + 1\right). \end{aligned} \tag{31}$$

Thus, combining (27), (29), (30) and (31), we get

$$|I(p, k_{i+1} - 1)| \leq A_i \epsilon^{-2},$$

for A_i defined in (25). Therefore, (24) holds in this case.

Case III: $p \in I(k_i + 2, k_{i+1} - 2)$.

Using the triangle inequality and A2 we obtain

$$\begin{aligned} \|\operatorname{grad} f(x_p)\|^2 &\leq \left(\|P_{p-1}\operatorname{grad} f(x_{p-1})\| + \|\operatorname{grad} f(x_p) - P_{p-1}\operatorname{grad} f(x_{p-1})\|\right)^2 \\ &\leq 2\left(\|P_{p-1}\operatorname{grad} f(x_{p-1})\|^2 + \|\operatorname{grad} f(x_p) - P_{p-1}\operatorname{grad} f(x_{p-1})\|^2\right) \\ &\leq 2\left(\|\operatorname{grad} f(x_{p-1})\|^2 + L^2\frac{\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-1}^2}\right). \end{aligned}$$

Therefore,

$$\begin{aligned} b_p &= b_{p-1} + \frac{\|\operatorname{grad} f(x_p)\|^2}{b_{p-1}} \\ &\leq b_{p-1} + 2\frac{\|\operatorname{grad} f(x_{p-1})\|^2 + L^2\left(\frac{\|\operatorname{grad} f(x_{p-1})\|}{b_{p-1}}\right)^2}{b_{p-1}} \\ &= b_{p-1} + 2\frac{\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-1}} + 2\frac{L^2\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-1}^3}. \end{aligned} \tag{32}$$

Notice that

$$b_{p-1} = b_{p-2} + \frac{\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-2}} \geq \frac{\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-2}}, \tag{33}$$

which implies that

$$\frac{1}{b_{p-1}} \leq \frac{b_{p-2}}{\|\operatorname{grad} f(x_{p-1})\|^2}. \tag{34}$$

Thus, by (32) and $b_{p-1} \geq b_{p-2}$, we get

$$\begin{aligned} b_p &\leq b_{p-1} + 2\frac{\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-2}} + 2\frac{L^2\|\operatorname{grad} f(x_{p-1})\|^2}{b_{p-1}^2 b_{p-2}} \\ &\leq b_{p-1} + 2b_{p-1} + 2\frac{L^2}{b_{p-1}^2} b_{p-1} \\ &= 3b_{p-1} + 2\frac{L^2}{b_{p-1}}. \end{aligned}$$

Then, using the inequalities $b_{p-1} < L$ and $b_{p-1} \geq b_{\min}$, we get

$$b_p \leq 3L + \frac{2L^2}{b_{\min}}. \tag{35}$$

On the other hand, by A1 and Lemma 2.1, for $j = k_i, \ldots, p-1$ we have

$$\begin{aligned} f(x_{j+1}) &\leq f(x_j) + \left\langle \operatorname{grad} f(x_j), -\frac{\operatorname{grad} f(x_j)}{b_j} \right\rangle + \frac{L}{2}\frac{\|\operatorname{grad} f(x_j)\|^2}{b_j^2} \\ &\leq f(x_j) - \frac{\|\operatorname{grad} f(x_j)\|^2}{b_j} + \frac{L}{2b_j}\frac{\|\operatorname{grad} f(x_j)\|^2}{b_j} \\ &\leq f(x_j) + \frac{L}{2b_{\min}}\frac{\|\operatorname{grad} f(x_j)\|^2}{b_j}. \end{aligned}$$

Thus,

$$\begin{aligned} f(x_p) - f(x_{k_i}) &= \sum_{j=k_i}^{p-1} f(x_{j+1}) - f(x_j) \\ &\leq \frac{L}{2b_{\min}} \sum_{j=k_i}^{p-1} \frac{\|\operatorname{grad} f(x_j)\|^2}{b_j} \\ &\leq \frac{L}{2b_{\min}} \sum_{j=k_i+1}^{p-1} \frac{\|\operatorname{grad} f(x_j)\|^2}{b_{j-1}} + \frac{L}{2b_{\min}}\frac{\|\operatorname{grad} f(x_{k_i})\|^2}{b_{k_i}} \\ &\leq \frac{L}{2b_{\min}} \sum_{j=k_i+1}^{p-1} (b_j - b_{j-1}) + \frac{L}{2b_{\min}}\frac{\|\operatorname{grad} f(x_{k_i})\|^2}{b_{k_i}} \\ &= \frac{L}{2b_{\min}}(b_{p-1} - b_{k_i}) + \frac{L}{2b_{\min}}\frac{\|\operatorname{grad} f(x_{k_i})\|^2}{b_{k_i}} \\ &\leq \frac{L}{2b_{\min}} b_{p-1} + \frac{L}{2b_{\min}}\frac{\|\operatorname{grad} f(x_{k_i})\|^2}{b_{k_i}}, \end{aligned}$$

Consequently, by the definition of p we get

$$f(x_p) \leq f(x_{k_i}) + \frac{L}{2b_{\min}}\left(L + \frac{\|\operatorname{grad} f(x_{k_i})\|^2}{b_{k_i}}\right). \tag{36}$$

Finally, combining (27), (29), (35) and (38) we obtain

$$|I(p, k_{i+1} - 1)| \leq A_i \epsilon^{-2},$$

and so (24) also holds in this case. □

Lemma 3.6. *Suppose that A1 and A2 are true, and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 2 such that (12) holds. Then, given $i \in \{0, \ldots, |D_T|\}$ we have*

$$|I(k_i, k_{i+1} - 1)| \leq C_i \epsilon^{-2} \tag{37}$$

where

$$C_i = L^2 + 4 + A_i. \tag{38}$$

where A_i is given in the equation (25).

Proof. Let $\mathcal{U}$ be the set defined in (23). If $\mathcal{U} = \emptyset$, then

$$b_j < L \quad \text{for all} \quad j \in I(k_i, k_{i+1} - 1).$$

Then, it follows from Lemma 3.4 with $q = k_{i+1} - 1$ that

$$|I(k_i, k_{i+1} - 1)| \leq L^2 \epsilon^{-2} + 2 \leq C_i \epsilon^{-2},$$

and so (37) holds. Assume that $\mathcal{U} \neq \emptyset$, and let $\hat{p}$ be the smallest element of $\mathcal{U}$. If $k_i + 2 \geq k_{i+1} - 2$, then $k_{i+1} - 1 \leq k_i + 3$ and consequently

$$|I(k_i, k_{i+1} - 1)| \leq 4 \leq C_i \epsilon^{-2}.$$

Therefore, assume that $\mathcal{U} \neq \emptyset$ and $k_i + 2 < k_{i+1} - 2$. Under these conditions, we can divide the rest of the proof in two cases.

Case I: $\hat{p} = k_{i+1} - 1$.

Because of the definition of $\hat{p}$ we have

$$b_j < L \quad \text{for all} \quad j \in I(k_i, \hat{p} - 1).$$

Consequently, it follows from Lemma 3.4 with $q = \hat{p} - 1$ that

$$|I(k_i, k_{i+1} - 1)| = |I(k_i, \hat{p} - 1)| + |\{\hat{p}\}| \leq L^2 \epsilon^{-2} + 3 \leq C_i \epsilon^{-2},$$

that is, (37) holds.

Case II: $\hat{p} \in I(k_i, k_{i+1} - 2)$.

If $\hat{p} = k_i$, then by Lemma 3.5 (with $p = \hat{p}$) and (38) we have

$$|I(k_i, k_{i+1} - 1)| = |I(\hat{p}, k_{i+1} - 1)| \leq C_i \epsilon^{-2}.$$

On the other hand, if $\hat{p} \in I(k_i + 1, k_{i+1} - 2)$, then it follows from Lemma 3.4 (with $q = \hat{p} - 1$) and Lemma 3.5 (with $p = \hat{p}$) that

$$|I(k_i, k_{i+1} - 1)| = |I(k_i, \hat{p} - 1)| + |I(\hat{p}, k_{i+1} - 1)| \leq L^2 \epsilon^{-2} + 2 + A_i \epsilon^{-2} \leq C_i \epsilon^{-2}.$$

Thus, in both cases (37) holds. □

Theorem 3.7. *Suppose that A1 and A2 are true, and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 such that (9) holds. Then,*

$$T < C_0 \epsilon^{-2} \tag{39}$$

if $\alpha = 0$, and

$$T < \left(\max_{i=0,\ldots,|D_T|} C_i \right) \left(\frac{\log\left(\|\operatorname{grad} f(x_0)\| \epsilon^{-1} \right)}{|\log(\alpha)|} + 1 \right) \epsilon^{-2} \tag{40}$$

if $\alpha \in (0,1)$, where C_i is defined in (38).

Proof. Indeed, combining (12) and Lemma 3.6, we get

$$\begin{aligned} T+1 &\leq (|D_T|+1) \max_{i=0,\ldots,|D_T|} |I(k_i, k_{i+1}-1)| \\ &\leq (|D_T|+1) \max_{i=0,\ldots,|D_T|} C_i \epsilon^{-2} \\ &= \left(\max_{i=0,\ldots,|D_T|} C_i \right) (|D_T|+1)\, \epsilon^{-2}. \end{aligned} \tag{41}$$

If $\alpha = 0$, it follows from Lemma 3.2 that $|D_T| = 0$, and so (39) is true. On the other hand, if $\alpha \in (0,1)$, it follows from Lemma 3.2 that

$$|D_T| < \frac{\log\left(\|\operatorname{grad} f(x_0)\| \epsilon^{-1} \right)}{|\log(\alpha)|}.$$

Combining this inequality with (41) we conclude that (40) is also true. □

As a direct consequence of Theorem 3.7 we have the following convergence result for Algorithm 1.

Corollary 3.8. *Suppose that A1 and A2 are true and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1. Then*

$$\liminf_{k\to+\infty} \|\operatorname{grad} f(x_k)\| = 0. \tag{42}$$

Notice that each iteration of Algorithm 1 takes only one gradient evaluation. Then, it follows from Theorem 3.7 that the conservative version ($\alpha = 0$) needs at most $\mathcal{O}(\epsilon^{-2})$ gradient evaluations to generate an ϵ-approximate stationary point. This bound agrees in order with evaluation complexity bounds obtained in [4, 3, 8] for different first-order Riemannian methods. In order to obtain a complexity bound for the flexible version of Algorithm (the one with $\alpha \in (0,1)$) let us consider the following additional assumptions:

A3. There exist $a_1, a_2, \theta > 0$ such that $f(x) \leq a_1 + a_2 \|\operatorname{grad} f(x)\|^\theta$ for all $x \in M$.

A4. There exists $\hat{b}_{\max} > 0$ such that $\hat{b}_k \leq \hat{b}_{\max}$ for all k, where $\hat{b}_k$ is defined in (7)

Theorem 3.9. *Suppose that A1-A4 are true, and let $\{x_k\}_{k\geq 0}$ be a sequence generated by Algorithm 1 with $\alpha \in (0,1)$, such that (9) holds. Then,*

$$T < C \left(\frac{\log\left(\|\operatorname{grad} f(x_0)\| \epsilon^{-1} \right)}{|\log(\alpha)|} + 1 \right) \epsilon^{-2}, \tag{43}$$

where

$$C = L^2 + 4 + A, \tag{44}$$

and

$$A = 4\left[\left(\hat{b} + \frac{2\|\text{grad}\, f(x_0)\|^2}{b_{\min}} + 3L + \frac{2L^2}{b_{\min}}\right)\left(1 + \frac{2L}{b_{\min}}\right) + 4(\tilde{f} - f_{low})\right]^2 \left(\frac{L^2}{b_{\min}^2} + 1\right)^2, \tag{45}$$

with

$$\tilde{f} = a_1 + a_2\|\text{grad}\, f(x_0)\|^\theta, \tag{46}$$

and $\hat{b} = \max\{\hat{b}_{\max}, b_0\}$.

Proof. Let $i \in \{1, \ldots, |D_T|\}$. By (18) we have

$$\|\text{grad}\, f(x_{k_i})\| \leq \|\text{grad}\, f(x_0)\|. \tag{47}$$

Moreover, by (6) and A4 we have

$$b_{k_i} \leq \begin{cases} b_0, & \text{if } i = 0, \\ \hat{b}_{\max}, & \text{if } k \geq 1. \end{cases}$$

Thus,

$$b_{k_i} \leq \max\left\{b_0, \hat{b}_{\max}\right\} = \hat{b}. \tag{48}$$

Finally, combining A3, (46) and (47) it follows that

$$f(x_{k_i}) \leq \tilde{f}. \tag{49}$$

Combining (47)-(49) with (38) and (24), we see that $C_i \leq L^2 + 4 + A$ with A given in (45). Consequently, (43) follows from (40) in Theorem 3.7. □

4 Numerical Results

In order to investigate the performance of Algorithm 1 we performed numerical experiments comparing the following MATLAB implementations:

- **RWNGrad**: Algorithm 1 with $\alpha = 0$ and $b_0 = \|\text{grad}\, f(x_0)\|$.
- **ARWNGrad**: Algorithm 1 with $\alpha = 0.9$, $\hat{c}$ variable, and

$$\hat{b}_k = \min\left\{\hat{b}_{\max}, \max\left\{b_{\min}, \frac{b_k}{\lambda\hat{c}}\right\}\right\} \tag{50}$$

  where $\lambda = 3$, $\hat{b}_{\max} = b_0 = \|\text{grad}\, f(x_0)\|$ and $b_{\min} = 10^{-4}$.

- **ARMIJO**: an implementation of the gradient method with Armijo line search, freely available from the Manopt Toobox [5][1].

[1]This toolbox is freely available in the website `https://www.manopt.org/`. Specifically, we used the codes `steepestdescent.m` and `linesearch.m`. In the initialization, we substituted the manifold retraction (`M.retr`) by the exponential map (`M.exp`).

The numerical experiments were performed with MATLAB 9.6.0 (R2019a) on a PC with microprocessor Intel(R) Core(TM) i7-3770 (3.40 GHz) and 16 GB of RAM memory. In all codes, we used the stopping conditions:

$$\|\text{grad}\, f(x_k)\| \leq \epsilon \equiv 10^{-4}, \tag{51}$$

and

$$k = 1,000. \tag{52}$$

We considered that a problem p was solved by a solver s when the execution of s was interrupted due to (51). In this sense, let

$$t_{p,s} = \text{time required by solver } s \text{ to solve problem } p,$$

where $t_{p,s} \equiv +\infty$ if solver s stopped due to (52). Then, as proposed in [7], the relative performance of solver s on problem p can be measured in terms of the *performance ratio*

$$r_{p,s} = \frac{t_{p,s}}{t_p^*},$$

with $t_p^* = \min\{t_{p,s} : s \in \mathcal{S}\}$, where $\mathcal{S}$ is the set of solvers being compared. In view of $r_{p,s}$, the *performance profile* for each code s is defined as:

$$\rho_s(\tau) = \frac{\text{no. of problems s.t. } r_{p,s} \leq \tau}{\text{total no. of problems}}, \quad \tau \geq 1. \tag{53}$$

In particular, $\rho_s(1)$ is the percentage of problems for which solver s was the fastest. In what follows, this number will be referred as the efficiency of solver s.

Our tests were made in the context of the manifold $M = (\mathbb{P}^n_{++}, \langle\cdot,\cdot\rangle)$, where $\mathbb{P}^n_{++}$ is the set of $(n \times n)$ symmetric positive definite matrices, with metric at the point X given by

$$\langle U, V\rangle = \text{tr}(VX^{-1}UX^{-1}), \quad X \in M, \quad U, V \in T_X M.$$

In this Riemannian manifold the exponential map $\exp_X : T_X M \to M$ is given by

$$\exp_X(V) = X^{1/2} \exp_m(X^{-1/2} V X^{-1/2}) X^{1/2},$$

where $\exp_m$ denotes the matrix exponential.

4.1 Problem Class 1

We considered the family of problems of the form

$$\min_{X \in \mathbb{P}^n_{++}} \quad f(X) \equiv \ln(\det(X))^2 - \ln(\det(X)), \tag{54}$$

The Riemannian gradient of f is given by[2]

$$\operatorname{grad} f(X) = [2\ln(\det(X)) - 1]X.$$

In our first experiment, we applied ARWNGrad with $\hat{c} \in \{1, 2, 3, 4, 5\}$ to solve (52) with $n \in \{10, 20, 50, 100, 150\}$. For each choice of n, the code was run for 100 starting points randomly generated with eigenvalues in the interval $(0, 20)$. Specifically, each starting matrix X_0 was constructed as $X_0 = Q^T \Lambda Q$, where each element of the diagonal matrix Λ was a random scalar drawn from the uniform distribution in the interval $(0, 20)$, and Q was given by the QR decomposition of a matrix with entries uniformly generated in the interval $(0, 1)$. Considering the five variants of ARWNGrad as our set of solvers $\mathcal{S}$, we computed the efficiency $\rho_s(1)$ of each choice of $\hat{c}$. The results are shown in Table 1.

Dimension/ARWNGrad	$\hat{c} = 1$	$\hat{c} = 2$	$\hat{c} = 3$	$\hat{c} = 4$	$\hat{c} = 5$
$n = 10$	0.00	0.05	0.15	0.23	**0.57**
$n = 20$	0.00	0.04	**0.39**	0.36	0.21
$n = 50$	0.00	0.09	0.18	**0.50**	0.23
$n = 100$	0.00	0.06	0.26	0.16	**0.52**
$n = 150$	0.00	0.11	0.18	0.23	**0.48**

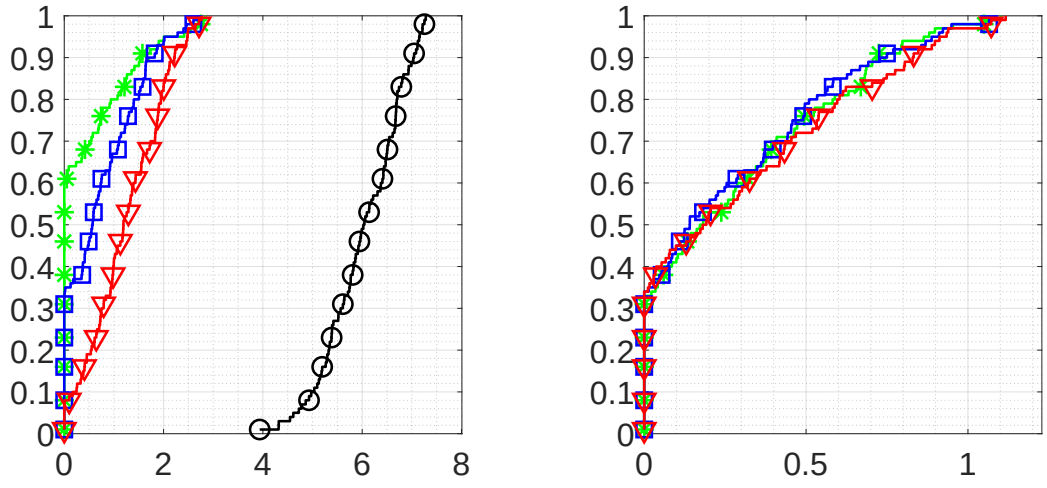
Table 1: Efficiency of variants of ARWNGrad in terms of CPU time, with respect to the parameter $\hat{c}$ and the dimension n.

Remarkably, the most efficient variants of ARWNGrad were the ones with $\hat{c} = 3$, $\hat{c} = 4$ or $\hat{c} = 5$. In particular, the choices ARWNGrad with $\hat{c} = 4$ and $\hat{c} = 5$ had the best overall performance, while ARWNGrad with $\hat{c} = 1$ had the worst performance. This result suggests that the selection of $t_{k+1} > t_k$ need to be done with caution (i.e., with $\hat{c} > 1$).

In our second experiment, we applied RWNGrad, ARMIJO and ARWNGrad with $\hat{c} \in \{4, 5\}$ to problem (52) with $n = 50$. Each code was run for 100 starting points randomly generated with eigenvalues in the interval $(0, r)$, considering $r \in \{20, 2000\}$. Figure 1 shows the performance profiles[3] (in $\log_2$ scale) for each value of r. As we can see, for $r = 20$, ARWNGrad with $\hat{c} = 4$ was the most efficient solver, while for $r = 2000$ the solvers had a similar performance, the exception being RWNGrad, which did not solve any problem.

[2] See Example 4 in [8].

[3] The performance profiles were generated using the code **perf.m** freely available in the website `http://www.mcs.anl.gov/~more/cops/`.



(a) $r = 20$ (b) $r = 2000$

Figure 1: Performance profiles (in terms of CPU time) for the solvers RWNGrad (black circle), ARWNGrad with $\hat{c} = 4$ (green asterisk), ARWNGrad with $\hat{c} = 5$ (blue square) and ARMIJO (red downtriangle).

4.2 Problem Class 2

We also considered the family of problems of the form

$$\min_{X \in \mathbb{P}^n_{++}} \; f(X) \equiv \frac{1}{2} \sum_{j=1}^{m} \left\| \ln(X^{-1/2} A_i X^{-1/2}) \right\|_F^2, \tag{55}$$

for fixed $A_1, \ldots, A_m \in \mathbb{P}^n_{++}$. The Riemannian gradient of f is given by[4]

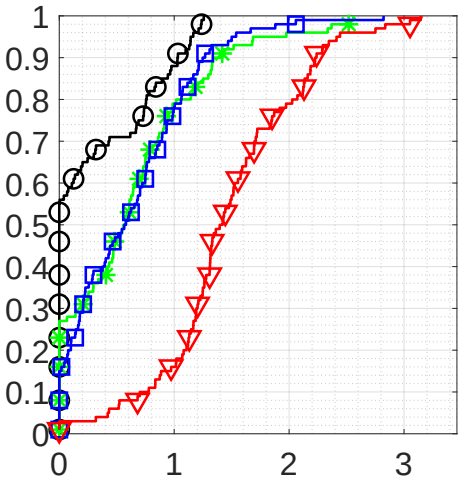
$$\operatorname{grad} f(X) = \sum_{i=1}^{m} X^{1/2} \ln(X^{1/2} A_i^{-1} X^{1/2}) X^{1/2}.$$

We tested the problems with $n \in \{20, 150\}$ and $m \in \{5, 10\}$. For each pair (n, m), 100 test problems were generated by randomly constructing 100 sets of matrices $A_1, \ldots, A_m \in \mathbb{P}^n_{++}$. Specifically, each matrix A_i was constructed as $A_i = Q_i^T \Lambda_i Q_i$, where each element of the diagonal matrix Λ_i was a random scalar drawn from the uniform distribution in the interval $(0, 20)$, while Q_i was given by the QR decomposition of a matrix with entries uniformly generated in the interval $(0, 1)$. For each problem, the initial point X_0 used was

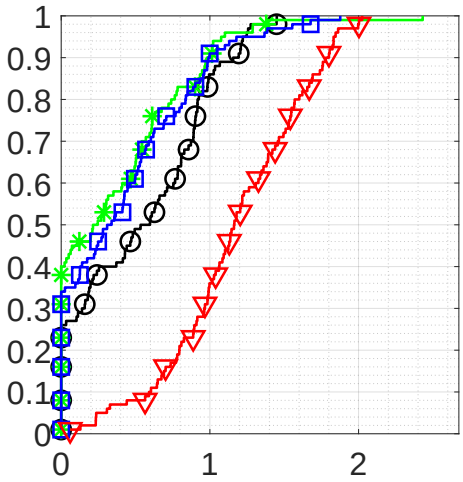
$$X_0 = \exp_m \left(\frac{1}{m} \sum_{i=1}^{m} \ln(A_i) \right).$$

Figure 2 presents the performance profiles in terms of CPU time for all combinations of n and m. As we can see, RWNGrad (which can be regarded as the limit case of ARWNGrad with $\hat{c} = +\infty$) was the most efficient solver in three of the four cases.

[4] See Section 5.2.1 in [8].

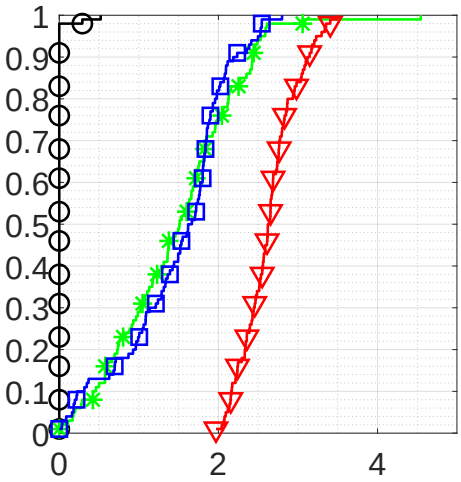


(a) $m = 5$

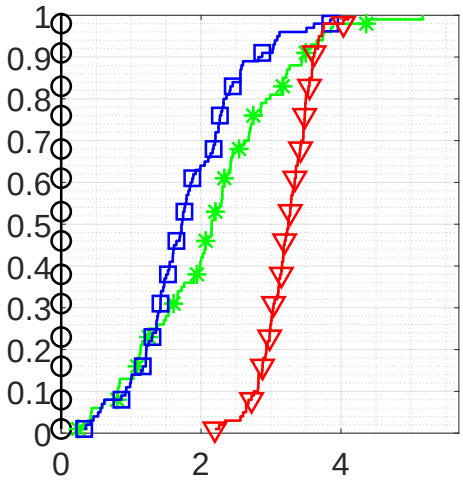


(b) $m = 10$

$n = 20$



(c) $m = 5$



(d) $m = 10$

$n = 150$

Figure 2: Performance profiles (in terms of CPU time) for the solvers RWNGrad (black circle), ARWNGrad with $\hat{c} = 4$ (green asterisk), ARWNGrad with $\hat{c} = 5$ (blue square) and ARMIJO (red downtriangle).

5 Conclusions

In this paper we proposed an adaptive Riemannian gradient method that works without function evaluations. The method is designed to minimize differentiable functions whose gradient vector field is L-Lipschitz continuous. It is adaptive in the sense that the stepsizes $\{t_k\}$ are dynamically adjusted over the iterations without requiring the knowledge of the Lipschitz constant L. However, different from line searches, which rely on function evaluations, our update rule for the stepsizes is based only on gradient information. The conservative version of our method (in which $\{t_k\}$ is monotonically decreasing) corresponds to a Riemannian version of the method WNGrad [15] recently proposed for Euclidean optimization. We showed that this version of the method performs at most $\mathcal{O}\left(\epsilon^{-2}\right)$ gradient evaluations to generate an ϵ-approximate stationary point of the objective function. For the flexible version of the method (in which $\{t_k\}$ is allowed to be nonmonotone) we showed a complexity bound of $\mathcal{O}\left(|\log(\epsilon)|\epsilon^{-2}\right)$ gradient evaluations. We also reported preliminary numerical results for test probems on the manifold of symmetric positive definite matrices. Some implementations of the proposed method compared favorably with a Riemannian gradient method with Armijo line search. The numerical results suggest that adaptive methods based only on gradient evaluations may be a competitive alternative to line search methods in the context of Riemannian optimization. An interesting topic for future research is the development of Riemannian trust-region methods [1] without function evaluations (in the lines of [9, 10]).

References

[1] P. A. Absil, C. G. Baker, K. A. Gallivan: Trust-Region Methods on Riemannian Manifolds. Foundations of Computational Mathematics 7, 303–330 (2007)

[2] L. Armijo: Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of Mathematics 16 (1966)

[3] G.C. Bento, O.P. Ferreira, J.G. Melo: Iteration-Complexity of Gradient, Subgradient and Proximal Point Methods on Riemannian Manifolds. Journal of Optimization Theory and Applications 173, 548–562 (2017)

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

[5] N. Boumal, B. Mishra, P-A. Absil, and R. Sepulchre, "Manopt, a MATLAB toolbox for optimization on manifolds," J. Mach. Learn. Res., vol. 15, no. 1, pp. 1455–1459, 2014

[6] A. Cauchy: Méthode générale pour la résolution des systemes d'équations simultanées. C. R. Acad. Sci. Paris 25, 536–538 (1847)

[7] E. Dolan, J.J. Moré: Benchmarking optimization software with performance profiles. Mathematical Programming 91, 201–2013 (2002)

[8] O. P. Ferreira, M. S. Louzeiro, L. F. Prudente: Gradient Method for Optimization on Riemannian Manifolds with Lower Bounded Curvature. SIAM Journal on Optimization 29, 2517–2541 (2019)

[9] G.N. Grapiglia, and G.F.D. Stella: An adaptive trust-region method without function evaluations. Computational Optimization and Applications. DOI: https://doi.org/10.1007/s10589-022-00356-0 (2022)

[10] S. Gratton, S. Jerad, Ph.L. Toint: First-Order Objective-Free Optimization Algorithms and Their Complexity. Optimization Online, March 2022.

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

[12] J.X. da Cruz Neto, L.L. de Lima, P.R. Oliveira: Geodesic Algorithms in Riemannian Geometry. Balkan Journal of Geometry and Its Applications, vol. 3, no. 2, pp.89-100 (1998)

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

[14] R. Ward, X. Wu, and L. Bottou: Adagrad stepsizes: Sharp convergence over nonconvex landscapes. Journal of Machine Learning Research 21, 1–30 (2020)

[15] X. Wu, R. Ward, and L. Bottou: WNGrad: Learn the Learning Rate in Gradient Descent. ArXiv:1803.02865, November 2020.

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