1

THE RIEMANNIAN BARZILAI-BORWEIN METHOD WITH NONMONOTONE LINE SEARCH AND THE MATRIX GEOMETRIC MEAN COMPUTATION

BRUNO IANNAZZO† AND MARGHERITA PORCELLI‡

Abstract. The Barzilai-Borwein method, an effective gradient descent method with clever choice of the step-length, is adapted from nonlinear optimization to Riemannian manifold optimization. More generally, global convergence of a nonmonotone line-search strategy for Riemannian optimization algorithms is proved under some standard assumptions. By a set of numerical tests, the Riemannian Barzilai-Borwein method with nonmonotone line-search is shown to be competitive in several Riemannian optimization problems. When used to compute the matrix geometric mean, known as the Karcher mean of positive definite matrices, it notably outperforms existing first-order optimization methods. Riemannian optimization; manifold optimization; Barzilai-Borwein algorithm; nonmonotone line-search; Karcher mean; matrix geometric mean; positive definite matrix.

1. Introduction. Globally convergent Barzilai-Borwein (BB) methods are well-known optimization methods for the solution of unconstrained and constrained optimization problems formulated in the Euclidean space. These algorithms are rather appealing for their simplicity, low-cost per iteration (gradient-type algorithms) and good practical performance due to a clever choice of the step-length ([2, 11, 13, 9]).

The key of success of BB methods lies in the explicit use of first-order information of the cost function on one side and, on the other side, in the implicit use of second-order information embedded in the step-length through a rough approximation of the Hessian of the cost function. This is crucial in the solution of problems where computing the Hessian represents a heavy burden, as when the problem dimension is large.

For strictly convex quadratic problems, global convergence of the BB method has been established in [28] while, in the general nonquadratic case, this property is guaranteed if BB is incorporated in a globalization strategy, see [29]. Due to the nonmonotone behaviour of the cost function through the BB steps, BB methods are generally combined with nonmonotone line-search strategies that do not impose a sufficient decrease condition on the cost function at each step, but rather on the maximum of the cost functions over the last, say, M steps (with $M > 1$) ([16, 29, 17]). It has been observed that if M is sufficiently large this strategy does not spoil the BB average rate of convergence ([29, 17]) and yields impressive good practical performance, especially in comparison with classical conjugate gradient methods, see [29].

Motivated by the nice theoretical and numerical properties of the (globalized) BB methods in the Euclidean space, we consider here a more general setting: BB methods for Riemannian manifold optimization based on retractions. Therefore, in this work we address the following optimization problem

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

where f is a smooth real valued function (the cost function) defined over a Riemannian manifold $\mathcal{M}$.

To the best of our knowledge, globally convergent nonmonotone BB type methods have been already proposed in Riemannian optimization only for the special case of Stiefel manifolds,

*Version of February 14, 2017.

† Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy (bruno.iannazzo@dmi.unipg.it).

‡ Dipartimento di Ingegneria Industriale, Università degli Studi di Firenze, Viale Morgagni 40/44, 50134 Firenze, Italy (margherita.porcelli@unifi.it).

see [14, 33, 21]. More precisely, in [14] the global convergence of a trust-region nonmonotone Levenberg-Marquardt algorithm for minimization problems on closed domains is provided and the application of the proposed method for minimization on Stiefel manifolds results in a BB method with a nonmonotone trust-region. The papers [21] and [33] are concerned with the BB method with nonmonotone line-search and the global convergence of the procedure is proved in [21] for Stiefel manifolds.

As a first contribution of this paper, under mild assumptions we prove global convergence of general (gradient-related) Riemannian optimization algorithms, when equipped with a nonmonotone line-search strategy. Line-search will be suitably adjusted to fit the framework of Riemannian manifold optimization and will be performed on the tangent space at a point. We therefore extend to Riemannian optimization the convergence results proved for the Euclidean space in [16, 29, 8].

Secondly, we introduce the Riemannian Barzilai-Borwein algorithm, which generalizes the Euclidean BB method, and discuss the corresponding algorithm embedded in a nonmonotone line-search strategy.

Thirdly, we apply the proposed algorithm to the computation of the geometric mean of positive definite matrices, the so called *Karcher mean*, that is very popular in many areas of applied mathematics and engineering, see, e.g., part II of [26]. The proposed method is a first-order algorithm, nevertheless, as pointed-out in [20], the Riemannian Hessian (or its full approximation) of the cost function defining the Karcher mean of positive definite matrices has a high computational complexity which makes the use of second-order algorithms prohibitive.

Finally, we benchmark the Riemannian BB over a set of problems involving different manifold structures and compare it with the trust-region and the steepest-descent line-search algorithms. Remarkably, in all our experiments, the performance of the proposed algorithm is superior to those of the existing first-order optimization algorithms for computing the Karcher mean of positive definite matrices ([31, 20, 5, 34]).

The paper is organized as follows. For later convenience, we recall in Section 2 some concepts from Riemannian optimization using the language and the tools described in the book by [1]. Section 3 is devoted to the global convergence analysis of the nonmonotone line-search strategy for gradient-related Riemannian optimization algorithms; then, in Section 4 we introduce the Riemannian BB algorithm and its globalization via nonmonotone line-search. Section 5 is devoted to the Karcher mean of positive definite matrices and to the implementation of the Riemannian BB method for its computation. In Section 6 numerical tests are performed to confirm the good behaviour of the Riemannian BB method for computing the Karcher mean of positive definite matrices and the possibility to apply the method to other problems of Riemannian optimization. Conclusions are drawn in Section 7.

2. Preliminaries on Riemannian optimization. The subject of Riemannian optimization is the computation of the minima of a differentiable function $f : \mathcal{M} \to \mathbb{R}$, where $\mathcal{M}$ is a real Riemannian manifold. The standard nonlinear optimization, which will be referred as Euclidean optimization, can be interpreted as a special case of Riemannian optimization, where $\mathcal{M} = \mathbb{R}^n$. For the ease of the reader, we briefly recall some of the basic concepts of Riemannian optimization based on retractions. We refer the reader to the book [1] for a thorough treatise of the topic.

In Riemannian optimization algorithms, the iterate x_k belongs to a manifold $\mathcal{M}$ while the gradient at x_k belongs to the tangent space $T_{x_k}\mathcal{M}$ at x_k, isomorphic to an Euclidean space. The gradient of f related to the geometry of $\mathcal{M}$ is denoted by $\nabla^{(\mathcal{R})} f$ and defined as follows: for any point $x \in \mathcal{M}$, $\nabla^{(\mathcal{R})} f(x)$ is the unique element of $T_x\mathcal{M}$ that satisfies

$$Df(x)[h] = \langle \nabla^{(\mathcal{R})} f(x), h \rangle_x, \quad \forall h \in T_x\mathcal{M},$$

where $\langle\cdot,\cdot\rangle_x$ is the Riemannian scalar product in $T_x\mathcal{M}$ and $Df(x) : T_x\mathcal{M} \to T_x\mathcal{M}$ is the differential of f at x. In the paper, we refer to $\nabla^{(\mathcal{R})}f(x)$ as the *Riemannian gradient* of f at x and frequently use the notation $g(x) := \nabla^{(\mathcal{R})}f(x)$ and $g_k := \nabla^{(\mathcal{R})}f(x_k)$.

A step of a gradient-like method in $\mathbb{R}^n$ involves summing points and gradient vectors. While in $\mathbb{R}^n$, the concept of moving in the direction of the negative gradient is straightforward, on a manifold, using the gradient $\nabla^{(\mathcal{R})}f(x_k)$ to update x_k is tricky since x_k lies on the manifold while the gradient belongs to $T_{x_k}\mathcal{M}$. Indeed, the next iterate x_{k+1} ought to be constructed following a geodesic starting at x_k and with tangent vector $-\nabla^{(\mathcal{R})}f(x_k)$, that is a curve $\gamma(t) : [0, t_0) \to \mathcal{M}$, with $t_0 \in (0, \infty]$. In contrast with the standard gradient-like methods for Euclidean unconstrained optimization, where t is an arbitrary positive number, in the Riemannian case, t must be chosen between 0 and t_0. Nevertheless, there are special manifolds, said to be geodesically complete, for which t_0 can be always chosen to be ∞.

Geodesics can be followed using the exponential map $\exp_x(v)$, which gives the point $\gamma(1)$ of a geodesic $\gamma(t)$ starting at $x = \gamma(0)$ and with tangent vector v. Since it is not always easy to find and follow geodesics, in manifold optimization the exponential map is approximated with a *retraction map* and we refer to *retraction-based* Riemannian optimization. The retraction is defined as a smooth map R from the tangent bundle $T\mathcal{M}$ to $\mathcal{M}$, such that $R_x(0_x) = x$ and $DR_x(0_x) = \mathrm{id}_{T_x\mathcal{M}}$, where R_x denotes the restriction of R to $T_x\mathcal{M}$, 0_x is the zero element of $T_x\mathcal{M}$, $\mathrm{id}_{T_x\mathcal{M}}$ is the identity map on $T_x\mathcal{M}$ and we have identified the tangent space to $T_x\mathcal{M}$ with itself.

The exponential map is a special case of retraction since it satisfies the above properties too. In practice using a retraction, in lieu of the exponential map, yields the same convergence results in an optimization algorithm, being possibly much easier to compute. In general, the domain of R is not necessarily the whole tangent bundle but for each point x we may assume that there is a neighbourhood of 0_x in $T_x\mathcal{M}$ such that R_x is defined at any point of this neighbourhood.

The step of a gradient-like method based on a retraction is of the form

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

where α_k is a step-length such that $-\alpha_k g_k$ belongs to the domain of R_{x_k}.

The extension of the Euclidean BB algorithm to the Riemannian setting requires the sum of gradients evaluated at different points (see Section 4), that is vectors belonging to different tangent spaces. In retraction-based optimization algorithms, the standard tool to move vectors from a tangent space to another is the *vector transport*. Roughly speaking, given two tangent vectors at x, say $v_x, w_x \in T_x\mathcal{M}$, a vector transport $\mathcal{T}$ associated with a retraction R, is a smooth map that generates a point $\mathcal{T}_{w_x}(v_x)$ that belongs to $T_{R_x(w_x)}\mathcal{M}$, and that is linear with respect to the argument. For the precise definition of the vector transport, we refer the reader to Sec. 8.1 of [1].

In many algorithms, a vector $v \in T_x\mathcal{M}$ has to be transported to the tangent space $T_y\mathcal{M}$, with $y \neq x$. Given a vector transport $\mathcal{T}$, for the sake of simplicity, we use the notation $\mathcal{T}_{x\to y}(v)$ to indicate the transport $\mathcal{T}_{w_x}(v)$, where w_x is such that $R_x(w_x) = y$. To ensure the existence of a w_x such that $R_x(w_x) = y$, we have to assume that the retraction has a sufficiently large domain so as to allow the search step and the vector transport.

A special case of vector transport, well-known in geometry, is the *parallel transport* that can be interpreted as a vector transport where the underlying retraction is the exponential map.

3. Nonmonotone line-search in Riemannian optimization. The Armijo line-search is a standard optimization strategy that consists in successively reducing the step-length (backtracking) until a sufficient decrease criterion, called *Armijo's condition*, is met. When combined with a gradient-type algorithm, this strategy guarantees global convergence of the overall procedure under mild assumptions, see e.g. [27].

In Euclidean geometry, given a point $x_k \in \mathbb{R}^n$ and a search direction d_k, the Armijo line-search strategy is as follows: starting from a fixed step-length $\lambda_k > 0$, compute the smallest nonnegative integer h_k such that

$$f(x_k + \sigma^{h_k}\lambda_k d_k) \leqslant f(x_k) + \gamma\sigma^{h_k}\lambda_k \nabla f(x_k)^T d_k,$$

for a suitable $\gamma \in (0,1)$ and reduction factor $\sigma \in (0,1)$, set the current step-length $\alpha_k = \sigma^{h_k}\lambda_k$ and update $x_{k+1} = x_k + \alpha_k d_k$. We have denoted by $\nabla f(x_k)$ the (Euclidean) gradient of f at x_k, that is the unique vector such that $\nabla f(x_k)^T h = Df(x_k)[h]$ for $h \in \mathbb{R}^n$.

A generalization of the Armijo line-search to Riemannian manifold optimization can be found in [1, Sec. 4.2], where the condition to find h_k is

$$f(R_{x_k}(\sigma^{h_k}\lambda_k d_k)) \leqslant f(x_k) + \gamma\sigma^{h_k}\lambda_k \langle g_k, d_k\rangle_{x_k}, \tag{3.1}$$

where R is a retraction on $\mathcal{M}$ and g_k is the Riemannian gradient of f at x_k. Then the iterate is updated as $x_{k+1} = R_{x_k}(\alpha_k d_k)$ with $\alpha_k = \sigma^{h_k}\lambda_k$. This strategy is sometimes referred to as curvilinear search since, in some sense, it is a search on a curve in $\mathcal{M}$ ([33]). Alternatively, it can be seen as a line-search on the tangent space.

In certain optimization algorithms, such as the BB method, global convergence is still ensured if one allows the cost function to increase *from time to time*, while asking that the maximum of the cost function over the last $M > 1$ steps decreases. This can be achieved using a nonmonotone Armijo line-search strategy and choosing h_k as the smallest nonnegative integer such that

$$f(x_k + \sigma^{h_k}\lambda_k d_k) \leqslant \max_{1\leqslant j\leqslant \min\{k+1,M\}} \{f(x_{k+1-j})\} + \gamma\sigma^{h_k}\lambda_k \nabla f(x_k)^T d_k,$$

where $\gamma \in (0,1)$ and $\sigma \in (0,1)$.

The adaptation of the nonmonotone line-search in Riemannian optimization consists in finding h_k as the smallest nonnegative integer such that

$$f(R_{x_k}(\lambda_k\sigma^{h_k} d_k)) \leqslant \max_{1\leqslant j\leqslant \min\{k+1,M\}} \{f(x_{k+1-j})\} + \gamma\sigma^{h_k}\lambda_k \langle g_k, d_k\rangle_{x_k}. \tag{3.2}$$

We now give our main result on the global convergence of the nonmonotone Armijo line-search in Riemannian optimization when the search direction d_k satisfies a general set of suitable conditions as, e.g., that of being *gradient-related*. The proof closely follows and generalizes the ones of [16], [8] and [29], where the case $\mathcal{M} = \mathbb{R}^n$ is considered.

THEOREM 3.1. *Let $\mathcal{M}$ be a Riemannian manifold with metric $\langle\cdot,\cdot\rangle_x$ at $x \in \mathcal{M}$, and let R be a retraction on $\mathcal{M}$ whose domain is the whole tangent bundle. Let $f : \mathcal{M} \to \mathbb{R}$ be a smooth function and $x_0 \in \mathcal{M}$ be such that $\mathcal{M}^{(x_0)} := \{x \in \mathcal{M} : f(x) \leqslant f(x_0)\}$ is a compact set.*

Let $\{x_k\} \subset \mathcal{M}$ be a sequence defined by

$$x_{k+1} = R_{x_k}(\alpha_k d_k), \quad k = 0, 1, 2, \ldots$$

with $\alpha_k \in \mathbb{R}$ and $d_k \in T_{x_k}\mathcal{M}$, and let g_k denote the Riemannian gradient of f at x_k. Let $\sigma \in (0,1)$, $\gamma \in (0,1)$, M be a nonnegative integer and $\{\lambda_k\}$ be a sequence such that $0 < a \leqslant \lambda_k \leqslant b$, where a and b are independent of k. Assume that:

(a) d_k is a descent direction, that is, $\langle g_k, d_k\rangle_{x_k} < 0$ for each k;

(b) the sequence $\{d_k\}$ is gradient-related, *namely, for any subsequence $\{x_k\}_{k\in K}$ of $\{x_k\}$ that converges to a nonstationary point of f, the corresponding subsequence $\{d_k\}_{k\in K}$ is bounded and satisfies*

$$\limsup_{k\in K} \langle g_k, d_k\rangle_{x_k} < 0;$$

(c) for any subsequence $\{x_k\}_{k\in K}$ of $\{x_k\}$ that converges to a stationary point, then there exists a subsequence $\{d_k\}_{k\in K_2}$ of $\{d_k\}$, with $K_2 \subset K$, such that

$$\lim_{k\in K_2} d_k = 0;$$

(d) $\alpha_k = \sigma^{h_k}\lambda_k$, where, for each k, h_k is the smallest nonnegative integer such that

$$f(R_{x_k}(\sigma^{h_k}\lambda_k d_k)) \leqslant \max_{1\leqslant j\leqslant \min\{k+1,M\}} \{f(x_{k+1-j})\} + \gamma\sigma^{h_k}\lambda_k\langle g_k, d_k\rangle_{x_k}. \tag{3.3}$$

Then every limit point of the sequence $\{x_k\}$ is stationary.

We prove the theorem using a few lemmas.

LEMMA 3.2. *In the hypotheses of Theorem 3.1, for $j = 1, 2, 3, \ldots$ let*

$$V_j = \max\{f(x_{jM-M+1}), f(x_{jM-M+2}), \ldots, f(x_{jM})\},$$

and $\nu(j) \in \{jM - M + 1, jM - M + 2, \ldots, jM\}$ be such that

$$f(x_{\nu(j)}) = V_j.$$

Then,

$$V_{j+1} \leqslant V_j + \gamma\alpha_{\nu(j+1)-1}\langle g_{\nu(j+1)-1}, d_{\nu(j+1)-1}\rangle_{x_{\nu(j+1)-1}}, \tag{3.4}$$

for all $j = 1, 2, \ldots$.

Proof. We prove that, for $\ell = 1, 2, \ldots, M$, and for all $j = 1, 2, \ldots$, we have

$$f(x_{jM+\ell}) \leqslant V_j + \gamma\alpha_{jM+\ell-1}\langle g_{jM+\ell-1}, d_{jM+\ell-1}\rangle_{x_{jM+\ell-1}} \leqslant V_j. \tag{3.5}$$

Observe that from assumption (a) and the definition of $\alpha_{jM+\ell-1}$ and γ we have the latter inequality.

For any j, the case $\ell = 1$ follows by the nonmonotone line-search assumption (d), while assuming that the statement is true for any $\ell' < \ell$, with $1 < \ell \leqslant M$, we can write

$$f(x_{jM+\ell}) \leqslant \max\{f(x_{jM-M+\ell}), \ldots, f(x_{jM}), f(x_{jM+1}), \ldots, f(x_{jM+\ell-1})\} + \gamma\alpha_{jM+\ell-1}\langle g_{jM+\ell-1}, d_{jM+\ell-1}\rangle_{x_{jM+\ell-1}}.$$

Since $f(x_{jM-M+\ell}), \ldots, f(x_{jM}) \leqslant V_j$ by definition and $f(x_{jM+1}), \ldots, f(x_{jM+\ell-1}) \leqslant V_j$ by inductive hypothesis, we have that (3.5) holds for ℓ.

Since $x_{\nu(j+1)} = x_{jM+\ell}$ for some $\ell \in \{1, \ldots, M\}$, the inequality (3.4) follows from (3.5). □

Let us define

$$K = \{\nu(1) - 1, \nu(2) - 1, \ldots\}, \tag{3.6}$$

and observe that $\nu(j) < \nu(j + 1) < \nu(j) + 2M$.

LEMMA 3.3. *In the hypotheses of Theorem 3.1 and with K as in* (3.6)*, every limit point of $\{x_k\}_{k\in K}$ is stationary.*

Proof. Since f is bounded in $\mathcal{M}^{(x_0)} = \{x \in \mathcal{M} : f(x) \leqslant f(x_0)\}$ and it is monotone in $\{x_{k+1}\}_{k\in K}$, there exists $\bar{f}$ such that $\lim_{k\in K} f(x_{k+1}) = \bar{f}$. Taking the limit of both sides of (3.4), and using the assumption (a) of Theorem 3.1, we get

$$\lim_{j\to\infty} \gamma\alpha_{\nu(j+1)-1}\langle g_{\nu(j+1)-1}, d_{\nu(j+1)-1}\rangle_{x_{\nu(j+1)-1}} = 0,$$

that is

$$\lim_{k\in K} \alpha_k \langle g_k, d_k\rangle_{x_k} = 0. \tag{3.7}$$

By contradiction, assume that there exists a nonstationary limit point, namely, there exists $K_2 \subset K$ such that $\lim_{k\in K_2} x_k = x_*$ and $g(x_*) \neq 0$.

Since d_k is gradient-related by assumption (b), there exists $K_3 \subset K_2$ such that $\lim_{k\in K_3}\langle g_k, d_k\rangle_{x_k}$ exists and is a negative number. Thus, $\lim_{k\in K_3} \alpha_k = 0$, in view of (3.7).

Since $\lambda_k \geqslant a > 0$, the condition $\lim_{k\in K_3} \alpha_k = 0$ implies that, for a sufficiently large $k \in K_3$, say for $k \in K_4 \subset K_3$, we have $h_k \geqslant 1$, and thus condition (3.3) is not verified for $\alpha'_k = \alpha_k/\sigma$. We have found a sequence $\{\alpha'_k\}_{k\in K_4}$ such that $\lim_{k\in K_4} \alpha'_k = 0$ and

$$f\big(R_{x_k}(\alpha'_k d_k)\big) > \max_{1\leqslant j\leqslant \min\{k+1,M\}}\{f(x_{k+1-j})\} + \gamma\alpha'_k\langle g_k, d_k\rangle_{x_k} \geqslant f(x_k) + \gamma\alpha'_k\langle g_k, d_k\rangle_{x_k}, \quad k \in K_4. \tag{3.8}$$

By taking a subsequence $K_5 \subset K_4$ we may assume that every point $\{x_k\}_{k\in K_5}$ and x_* belong to a single coordinate chart.

By the previous relation (3.8), the fact that $R_x(0) = x$ for $x \in \mathcal{M}$, and the mean value theorem applied to the smooth function $f \circ R_{x_k} : T_{x_k}\mathcal{M} \to \mathbb{R}$, there exists $\xi_k \in (0, \alpha'_k)$, such that

$$\begin{aligned}\frac{f\big(R_{x_k}(\alpha'_k d_k)\big) - f\big(R_{x_k}(0)\big)}{\alpha'_k} &= D(f \circ R_{x_k})(\xi_k d_k)[d_k] \\ &= Df(R_{x_k}(\xi_k d_k))[DR_{x_k}(\xi_k d_k)[d_k]] \\ &= \langle g(R_{x_k}(\xi_k d_k)), DR_{x_k}(\xi_k d_k)[d_k]\rangle_{R_{x_k}(\xi_k d_k)} > \gamma\langle g_k, d_k\rangle_{x_k}.\end{aligned} \tag{3.9}$$

Since d_k is bounded by assumption (b), there exists $K_6 \subset K_5$ such that $\lim_{k\in K_6} d_k = d \neq 0$ and thus $\lim_{k\in K_6} \xi_k d_k = 0$, and since the retraction $R_x(v)$ is smooth as a function of both x and v, we have $\lim_{k\in K_6} R_{x_k}(\xi_k d_k) = R_{x_*}(0) = x_*$ and $\lim_{k\in K_6} DR_{x_k}(\xi_k d_k)[d_k] = DR_{x_*}(0)[d] = d$.

Finally, since f and the Riemannian metric are smooth, using the chart where the sequence and x_* lie, and taking limits on the last line of (3.9) we get

$$\lim_{k\in K_6} \langle g(R_{x_k}(\xi_k d_k)), DR_{x_k}(\xi_k d_k)[d_k]\rangle_{R_{x_k}(\xi_k d_k)} = \langle g(x_*), d\rangle_{x_*} \geqslant \gamma\langle g(x_*), d\rangle_{x_*}.$$

Now we have $(1-\gamma)\langle g(x_*), d\rangle_{x_*} \geqslant 0$, with $\gamma \in (0,1)$, and on the other hand, by the choice of K_3, $\langle g(x_*), d\rangle_{x_*} < 0$ which leads to a contradiction. □

LEMMA 3.4. *In the hypotheses of Theorem 3.1, let $K(\ell)$ for $\ell \geqslant 0$ be the set $\{k+\ell, k \in K\}$. For each $\ell \geqslant 0$, every limit point of $\{x_k\}_{k\in K(\ell)}$ is stationary.*

Proof. We give a proof by induction on ℓ. Since $K(0) = K$, the case $\ell = 0$ is Lemma 3.3.

We assume that the property is true for $K(\ell)$ and we prove it for $K(\ell+1)$. Let x_* be a limit point of $\{x_k\}_{k\in K(\ell+1)}$, namely, there exists $H_1 \subset K(\ell+1)$ such that $\lim_{k\in H_1} x_k = x_*$; we want to prove that x_* is stationary. The sequence $\{x_{k-1}\}_{k\in H_1}$ belongs to the compact set $\mathcal{M}^{(x_0)}$ and thus it has a subsequence $\{x_{k-1}\}_{k\in H_2}$, with $H_2 \subset H_1$, converging to a certain limit point y. By inductive hypothesis, since $\{x_{k-1}\}_{k\in H_2}$ is a subsequence of $\{x_k\}_{k\in K(\ell)}$, we have that y is a stationary point.

By assumption (c), there exists $H_3 \subset H_2$ such that $\lim_{k\in H_3} d_{k-1} = 0$, and since $\alpha_j \leqslant \lambda_j \leqslant b$ for any j, we have

$$x_* = \lim_{k\in H_3} x_k = \lim_{k\in H_3} R_{x_{k-1}}(\alpha_{k-1}d_{k-1}) = R_y(0) = y;$$

hence, x_* is stationary. □

Proof. (of Theorem 3.1). Since $\{1,2,\ldots\} = \bigcup_{\ell=0}^{2M} K(\ell)$, one can see that for any sequence $\{x_k\}_{k\in K_0}$ converging to x_*, there exists $0 \leqslant \widehat{\ell} \leqslant 2M$ such that infinitely many terms of $\{x_k\}_{k\in K_0}$ belong to $\{x_k\}_{k\in K(\widehat{\ell})}$, that is, there exists $K_{00} \subset K_0 \cap K(\widehat{\ell})$ such that $\lim_{k\in K_{00}} x_k = x_*$. Since the sequence $\{x_k\}_{k\in K_{00}}$ belongs to $\{x_k\}_{k\in K(\widehat{\ell})}$, using Lemma 3.4, we conclude that x_* is a stationary point. □

A few remarks on the assumptions of Theorem 3.1 are in order. Provided that x_k is not stationary for all k, assumptions (a)-(c) are always fulfilled for non-finite sequences generated by a gradient-type algorithm, being $d_k = -g_k$. The requirement that $\mathcal{M}^{(x_0)}$ is a compact set is natural in nonlinear minimization. The strongest hypothesis is that the retraction R has to be defined in the whole tangent bundle. This occurs in several practical cases, but it is not always the case. On the other hand, in order to apply the algorithm and prove its global convergence, we only need that for each k, $R_{x_k}(td_k)$ is defined for $t \in [0, \lambda_k]$; this might be true also if R is not globally defined.

4. The Riemannian Barzilai-Borwein method. In this section we propose an adaptation of the Euclidean BB method to the solution of the Riemannian manifold optimization problem (1.1). The BB method belongs to the class of first-order (or gradient-type) optimization algorithms, namely algorithms that only use the gradient of the cost function, while disregarding the Hessian (second-order information). Given the problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a given smooth cost function, the simplest gradient-type method is a line-search method based on the steepest descent direction: given $x_0 \in \mathbb{R}^n$, the algorithm generates a sequence $\{x_k\}_k$ with

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{4.1}$$

where α_k is a suitable step-length. The majority of gradient-type methods are also descent methods, that is, $\{x_k\}_k$ is built so that the cost function decreases monotonically on $\{x_k\}_k$, i.e., $f(x_{k+1}) < f(x_k)$ for all k, unless the sequence converges in a finite number of steps. The ideal step-length α_k satisfies

$$\alpha_k = \operatorname*{argmin}_{\alpha} f(x_k - \alpha \nabla f(x_k)) \tag{4.2}$$

('exact line-search'); however this value is usually unnecessarily expensive to compute for a general nonlinear cost function f. More practical strategies perform an 'inexact' line-search to identify a step-length that achieves adequate reductions in f at minimal cost.

The steepest-descent method with exact line-search is commonly considered ineffective because of its slow convergence rate and its oscillatory behaviour, that has been completely understood and studied in the convex quadratic case, see [27].

The BB method provides an alternative strategy for the choice of the step-length; while it does not guarantee the cost function decrease at each step, in some problems, it yields impressive good practical performance. The basic idea of the BB method is to solve, for $k \geqslant 1$, the least-squares problem

$$\min_t \|s_k t - y_k\|_2, \tag{4.3}$$

with $s_k := x_{k+1} - x_k$ and $y_k := \nabla f(x_{k+1}) - \nabla f(x_k)$, which, assuming that $x_{k+1} \neq x_k$, has the

unique solution $t = \frac{s_k^T y_k}{s_k^T s_k}$. When $s_k^T y_k > 0$, the BB step-length is chosen to be

$$\alpha_{k+1}^{BB} = \frac{s_k^T s_k}{s_k^T y_k}. \tag{4.4}$$

The overdetermined system $s_k t = y_k$ is in fact a Quasi-Newton secant equation of the type

$$A_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k),$$

where we require the matrix $A_{k+1} \in \mathbb{R}^{n\times n}$ to be a scalar multiple of the identity matrix, and then treat it as a linear least-squares problem. The Quasi-Newton method thus results in a gradient-type method where second-order information is implicitly embedded in the step-length through a cheap approximation of the Hessian.

As in the Euclidean case, the idea of the Riemannian BB method is to approximate the action of the Riemannian Hessian of f at a certain point by a multiple of the identity.

At the $(k+1)$-st step, the Hessian is a linear map from $T_{x_{k+1}}\mathcal{M}$ to $T_{x_{k+1}}\mathcal{M}$. Instead of the increment $x_{k+1} - x_k$, we can consider the vector $\eta_k = -\alpha_k g_k$ belonging to $T_{x_k}\mathcal{M}$, and transport it to $T_{x_{k+1}}\mathcal{M}$, yielding

$$s_k := \mathcal{T}_{\eta_k}(\eta_k) = \mathcal{T}_{x_k \to x_{k+1}}(-\alpha_k g_k) = -\alpha_k \mathcal{T}_{x_k \to x_{k+1}}(g_k). \tag{4.5}$$

We assume here that a vector transport between x_k and x_{k+1} exists.

To obtain y_k we need to subtract two gradients that lie in two different tangent spaces. To be coherent with the manifold structure and to work on $T_{x_{k+1}}\mathcal{M}$, this difference should be made after g_k is transported to $T_{x_{k+1}}\mathcal{M}$ so that

$$y_k := g_{k+1} - \mathcal{T}_{x_k \to x_{k+1}}(g_k) = g_{k+1} + \frac{1}{\alpha_k}\mathcal{T}_{\eta_k}(\eta_k). \tag{4.6}$$

Imposing a scalar multiple of the identity as the approximation of the Hessian, the Riemannian counterpart of the secant equation (4.3) can be written again as

$$s_k t = y_k,$$

in the unknown $t \in \mathbb{R}$. The least-squares approximation with respect to the metric of $T_{x_{k+1}}\mathcal{M}$ yields $t = \frac{\langle s_k, y_k\rangle_{x_{k+1}}}{\langle s_k, s_k\rangle_{x_{k+1}}}$. Therefore the Riemannian BB step-length has the form

$$\alpha_{k+1}^{BB} = \frac{\langle s_k, s_k\rangle_{x_{k+1}}}{\langle s_k, y_k\rangle_{x_{k+1}}}, \tag{4.7}$$

provided that $\langle s_k, y_k\rangle_{x_{k+1}} > 0$.

REMARK 4.1. Alternative strategies for the BB step-length have been proposed in Euclidean optimization, see e.g. [2, 11, 30, 15]. A standard choice is the step-length obtained by considering the least-square problem $\min_t \|s_k - y_k t\|_2$ ([2]) that, by symmetry with respect to (4.3), yields the Riemannian BB step-length

$$\alpha_{k+1}^{BB\,\prime} = \frac{\langle s_k, y_k\rangle_{x_{k+1}}}{\langle y_k, y_k\rangle_{x_{k+1}}}, \tag{4.8}$$

or the one obtained by alternating α_{k+1}^{BB} and $\alpha_{k+1}^{BB\,\prime}$.

Remark 4.2. As opposed to the Euclidean case, different alternatives are available for y_k. For instance, we can do the subtraction at $T_{x_k}\mathcal{M}$ after g_{k+1} is transported over there. Or, we can transport the two vectors to the tangent space at any point of the geodesic joining x_k and x_{k+1}, or even at any point of the manifold reachable from x_k and x_{k+1} through a vector transport. All these alternatives for y_k lead to different versions of the Riemannian BB method.

Yet another possibility is to avoid completely the vector transport, identifying $T_{x_k}\mathcal{M}$ with $T_{x_{k+1}}\mathcal{M}$ and then approximating the transport with the identity. This approximation is justified by the property $\mathcal{T}_{0_x}(v) = v$, where $v, 0_x \in T_x\mathcal{M}$ (see [1]), but it is not recommended if one wants to preserve the structure.

4.1. The globalized Riemannian Barzilai-Borwein algorithm. Algorithm 1 summarizes the main steps of the Riemannian BB method with a nonmonotone line-search strategy. Here α_k^{BB} plays the role of λ_k used in Section 3 and its updating at Step 5 follows [8]. Moreover, we assume that a retraction R and a transport vector $\mathcal{T}$ mapping are supplied for the manifold under consideration.

Algorithm 1 (The Riemannian Barzilai-Borwein with nonmonotone line-search (RBB-NMLS) algorithm). *Set the line-search parameters: the step-length reduction factor $\sigma \in (0,1)$; the sufficient decrease parameter $\gamma \in (0,1)$; the integer parameter for the nonmonotone line-search $M > 0$; the upper and lower bounds for the step-length $\alpha_{max} > \alpha_{min} > 0$.*
Set the initial values: the starting point $x_0 \in \mathcal{M}$; the starting step-length $\alpha_0^{BB} \in [\alpha_{min}, \alpha_{max}]$.

1. *Compute $f_0 = f(x_0)$ and $g_0 = \nabla^{(\mathcal{R})} f(x_0)$.*
2. *For $k = 0, 1, 2, \ldots$*
   find the smallest $h = 0, 1, 2, \ldots$ such that
$$f(R_{x_k}(-\sigma^h \alpha_k^{BB} g_k)) \leqslant \max_{1 \leqslant j \leqslant \min\{k+1, M\}} f_{k+1-j} - \gamma \sigma^h \alpha_k^{BB} \langle g_k, g_k \rangle_{x_k}$$
   and set $\alpha_k := \sigma^h \alpha_k^{BB}$.
3. *Compute $x_{k+1} = R_{x_k}(-\alpha_k g_k)$, $f_{k+1} = f(x_{k+1})$ and $g_{k+1} = \nabla^{(\mathcal{R})} f(x_{k+1})$.*
4. *Compute y_k and s_k as in (4.6) and (4.5), respectively.*
5. *Set $\tau_{k+1} = \frac{\langle s_k, s_k \rangle_{x_{k+1}}}{\langle s_k, y_k \rangle_{x_{k+1}}}$.*
6. *Set*
$$\alpha_{k+1}^{BB} = \begin{cases} \min\{\alpha_{max}, \max\{\alpha_{min}, \tau_{k+1}\}\} & \text{if } \langle s_k, y_k \rangle_{x_{k+1}} > 0, \\ \alpha_{max} & \text{otherwise.} \end{cases}$$

We point out that τ_{k+1} can be choose using alternative strategies, as mentioned in in Remark 4.1. Moreover, the backtracking strategy considered in Step 2 of Algorithm 1 could be slightly generalized without affecting the global convergence. In particular, the contraction factor σ could vary at each iteration of the line-search and, for example, could be chosen by safeguarded interpolation, see Sec. 3.5 in [27].

If x_k is not a stationary point, then assuming that $R_{x_k}(-\alpha g_k)$ is defined for $\alpha > 0$, by the mean value theorem (compare equation (3.9)), there exists $\xi \in (0, \alpha)$, such that

$$\begin{aligned} &\frac{f(R_{x_k}(-\alpha g_k)) - f(x_k)}{\alpha} + \gamma \langle g_k, g_k \rangle_{x_k} \\ &\quad = \big\langle g(R_{x_k}(-\xi g_k)), DR_{x_k}(-\xi g_k)[-g_k] \big\rangle_{R_{x_k}(-\xi g_k)} + \gamma \langle g_k, g_k \rangle_{x_k} \overset{\alpha \to 0}{\to} (\gamma - 1) \langle g_k, g_k \rangle_{x_k} < 0, \end{aligned}$$

since $\gamma < 1$. Thus, for α sufficiently small, we have

$$f(R_{x_k}(-\alpha g_k)) \leqslant f(x_k) - \gamma\, \alpha \langle g_k, g_k \rangle_{x_k} \leqslant \max_{1 \leqslant j \leqslant \min\{k+1, M\}} \{f(x_{k+1-j})\} - \gamma\, \alpha \langle g_k, g_k \rangle_{x_k},$$

that is, the nonmonotone Armijo's condition at Step 2 of Algorithm 1 is verified after a finite number of step reductions.

The practical implementation of the various steps strictly depends on the structure of the cost function f and of the underlying manifold $\mathcal{M}$; further details will be discussed in the next section for the computation of the matrix geometric mean. Other implementation issues are postponed to Section 6.

We remark that the use of the nonmonotone strategy requires the additional cost of evaluating the cost function at each trial point $\tilde{x}_k$ (Step 3). Indeed, a 'pure' BB algorithm (i.e., without line-search) consists in setting $x_{k+1} = R_{x_k}(-\alpha_k^{BB} g_k)$ without computing α_k at Step 2. We will refer to RBB-NMLS and to RBB to indicate the Riemannian BB algorithm with and without nonmonotone line-search, respectively. Finally, note that if $M = 1$ then Algorithm 1 corresponds to the standard (monotone) line-search with Armijo's rule.

5. Computing the Karcher mean of positive definite matrices. In this section we recall the Riemannian optimization problem whose solution is the Karcher mean of positive definite matrices, and propose the adaptation of the RBB method for this problem.

First, we describe the set of positive definite matrices as a Riemannian manifold. Let $\mathcal{P}_n$ denote the set of positive definite matrices of size n and $\mathbb{H}_n$ denote the real vector space of Hermitian matrices. The set $\mathcal{P}_n$ is an open subset of the Euclidean space $\mathbb{H}_n$ (with the scalar product $\langle E, F\rangle = \operatorname{trace}(EF)$ for $E, F \in \mathbb{H}_n$) and thus it is a differentiable manifold with $T(\mathcal{P}_n)_X$ isomorphic to $\mathbb{H}_n$ for each $X \in \mathcal{P}_n$.

Two main Riemannian structures on $\mathcal{P}_n$ are commonly used in the literature: the first one is obtained by the Euclidean scalar product of $\mathbb{H}_n$ at each point $X \in \mathcal{P}_n$, the second is the one induced by the scalar product

$$\langle E, F\rangle_X = \operatorname{trace}(X^{-1}EX^{-1}F), \qquad E, F \in \mathbb{H}_n, \qquad X \in \mathcal{P}_n \tag{5.1}$$

see Ch. XII in [22] and Sec. 6.3 in [25]. Since the first structure is trivial and the second one does not have an established name, in the following we will refer to the second geometry as the *Riemannian structure of* $\mathcal{P}_n$.

The scalar product (5.1) induces a metric on $\mathcal{P}_n$ such that the distance between $A, B \in \mathcal{P}_n$ is

$$\delta(A, B) = \|\log(A^{-1/2}BA^{-1/2})\|_F, \tag{5.2}$$

where $\|\cdot\|_F$ is the Frobenius norm. The resulting metric space is complete [3], simply connected and with nonpositive curvature, thus it is an example of Cartan-Hadamard manifold.

There exists a unique geodesic joining A and B in $\mathcal{P}_n$ and whose natural parametrization is

$$\gamma(t) = A(A^{-1}B)^t = A^{1/2}(A^{-1/2}BA^{-1/2})^tA^{1/2} =: A\#_tB, \qquad t \in [0, 1],$$

see [23]. Notice that the geodesic can be indefinitely extended for $t \in \mathbb{R}$.

A nice feature of this geometry is that for any set of matrices $A_1, \ldots, A_m \in \mathcal{P}_n$, there exists unique $X \in \mathcal{P}_n$ that minimizes the function

$$f(X) := f(X; A_1, \ldots, A_m) = \sum_{k=1}^{m} \delta^2(X, A_k). \tag{5.3}$$

This point is called *matrix geometric mean* or *Karcher mean* of $A_1, \ldots, A_m$ and it is established as the geometric mean of matrices. Therefore, we state the problem of computing the Karcher mean as the problem of minimizing the above function f over the manifold $\mathcal{P}_n$. A class of algorithms developed for this problem can be found in [5] and in [20].

Since the cost function f in (5.3) is smooth, the corresponding derivative can be computed and two different gradients can be defined, one for each considered geometry, i.e. the Euclidean and Riemannian geometry.

A tedious computation proves that the gradient with respect to the Euclidean geometry, that is the unique matrix $\nabla^{(\mathcal{E})} f(X)$ such that $Df(X)[H] = \operatorname{trace}(\nabla^{(\mathcal{E})} f(X) H)$, is

$$\nabla^{(\mathcal{E})} f(X) = 2 \sum_{k=1}^{m} X^{-1} \log(X A_k^{-1}).$$

While gradient with respect to the Riemannian structure, namely the Riemannian gradient, defined as the unique Hermitian matrix $\nabla^{(\mathcal{R})} f(X)$ such that $Df(X)[H] = \langle \nabla^{(\mathcal{R})} f(X), H \rangle_X$, for $H \in \mathbb{H}_n$, turns out to be

$$\nabla^{(\mathcal{R})} f(X) = -2 \sum_{k=1}^{m} X \log(X^{-1} A_k) \tag{5.4}$$

see [24]. Notice that $\nabla^{(\mathcal{E})} f(X) = 0$ if and only if $\nabla^{(\mathcal{R})} f(X) = 0$.

An interesting property of the Riemannian geometry of $\mathcal{P}_n$ is that the function f in (5.3) is geodesically strictly convex with respect to the Riemannian geometry of $\mathcal{P}_n$, that is for any $A, B \in \mathcal{P}_n$, with $A \neq B$, we have (see [3])

$$f(A \#_t B) < (1-t) f(A) + t f(B), \qquad t \in (0,1);$$

on the contrary, it can be shown that f is not convex in the Euclidean sense.

The application of Algorithm 1 for the Karcher mean computation requires the definition of the retraction R and transport vector $\mathcal{T}$ mappings for the manifold $\mathcal{P}_n$.

Since the final point of a geodesic starting at A and with tangent vector V is $A \exp(A^{-1} V)$, the retraction at point A has the form $R_A(V) = A \exp(A^{-1} V)$. It follows that the iterate update of RBB is

$$X_{k+1} = R_{X_k}(-\alpha_k^{BB} g_k) = X_k \exp(-\alpha_k^{BB} X_k^{-1} g_k), \tag{5.5}$$

where α_k^{BB} is defined by (4.7) (or in an alternative way as discussed in Remark 4.1) and s_k and y_k are given below.

Notice that the retraction (5.5) is the exponential map with respect to the Riemannian geometry of $\mathcal{P}_n$, and thus the vector transport is given by the parallel transport. In the Riemannian geometry of $\mathcal{P}_n$, the parallel transport of a vector $V \in \mathbb{H}^n$ from the tangent space at A to the tangent space at B is given by (see [32])

$$\mathcal{T}_{A \to B}(V) = (BA^{-1})^{1/2} V (A^{-1} B)^{1/2}.$$

Notice that the parallel transport is a congruence of the type MVM^*, with $M = (BA^{-1})^{1/2}$.

Using the $\operatorname{vec}(\cdot)$ operator which stacks the columns of a matrix into a long vector and the Kronecker product, we get the simple expression

$$\mathcal{T}_{A \to B} \operatorname{vec}(V) = (\overline{M} \otimes M) \operatorname{vec}(V), \qquad M = (BA^{-1})^{1/2}.$$

Alternatively, we can also write $\mathcal{T}_{A \to B} = (A \#_{1/2} B) A^{-1} V A^{-1} (A \#_{1/2} B)$.

In view of (4.5) and (4.6), it is sufficient to transport a vector along itself, or, equivalently, along the geodesic tangent to it. We now derive the formula for this case that is simpler than those above.

Let $V_A \in T_A\mathcal{P}_n$ and let $B := R_A(V_A) = A\exp(A^{-1}V_A)$, then $A^{-1}B = \exp(A^{-1}V_A)$ and

$$\begin{aligned}\mathcal{T}_{V_A}(V_A) = \mathcal{T}_{A\to B}(V_A) &= A(A^{-1}B)^{1/2}A^{-1}V_A(A^{-1}B)^{1/2} \\ &= A\exp(A^{-1}V_A)^{1/2}A^{-1}V_A\exp(A^{-1}V_A)^{1/2} = V_AA^{-1}R_A(V_A) = V_A\exp(A^{-1}V_A), \quad (5.6)\end{aligned}$$

where we have used standard properties of matrix functions (compare Theorem 1.13 in [18]).

The formula (5.6) yields a simple expressions for s_k and y_k of equations (4.5) and (4.6), respectively,

$$s_k = -\alpha_k g_k \exp(-\alpha_k X_k^{-1} g_k), \qquad y_k = g_{k+1} - g_k \exp(-\alpha_k X_k^{-1} g_k),$$

that characterize the Riemannian BB method for the Karcher mean computation.

As a final remark we observe that the retraction used in the Riemannian geometry of the positive definite matrices is defined in the whole tangent bundle and that, for any $X_0 \in \mathcal{P}_n$, the level set $\mathcal{M}^{(X_0)} = \{X \in \mathcal{P}_n : f(X) \leqslant f(X_0)\}$ is a compact set (compare the proof of Theorem 2.1 in [7]). Thus, this problem fits the hypotheses of Theorem 3.1.

5.1. Implementation of the RBB method for the Karcher mean. The RBB method requires the computation of several quantities of the type $A\varphi(A^{-1}B)$, where A is positive definite, B is Hermitian and φ is a real function. This is indeed the case for the Riemannian gradient (5.4) and the retraction (5.5), while formula (5.6) is fairly similar.

Following [19], these quantities can be efficiently computed by forming the Cholesky factorization of $A = R_A^*R_A$. Using the similarity invariance of matrix functions, we write

$$A\varphi(A^{-1}B) = R_A^*R_A\varphi(R_A^{-1}(R_A^*)^{-1}B) = R_A^*\varphi((R_A^*)^{-1}BR_A^{-1})R_A,$$

and observe that $(R_A^*)^{-1}BR_A^{-1} = UDU^*$, where $D = (d_{ij})$ is diagonal real and U is unitary, so that we obtain

$$A\varphi(A^{-1}B) = R_A^*U\,\mathrm{diag}(\varphi(d_{11}),\dots,\varphi(d_{nn}))U^*R_A.$$

The cost of this basic operation is approximately $\gamma_1 n^3$, where γ_1 is a moderate constant.

Since one step of the RBB algorithm requires $m+1$ evaluations of the type $A\varphi(A^{-1}B)$ (m are needed to compute the gradient and one for the retraction), the cost of the algorithm is about $\gamma_1(m+1)n^3k$ arithmetic operations (ops), where k is the number of steps needed to achieve convergence, while m is the number of matrices and n is their size. For comparison, the Richardson algorithm in [5], the steepest descent with fixed step-length and the Majorization-Minimization algorithm in [34] have essentially the same cost.

Regarding the cost of the version of the RBB algorithm with the nonmonotone line-search, we need to take into account the number of ops needed to compute the distance (5.2) during the cost function evaluation. By a similar use of Cholesky factorizations and an eigenvalue computation, the distance can be computed with $\gamma_2 n^3$ ops, where γ_2 is smaller than γ_1. Overall, the cost function evaluation requires $\gamma_2 mn^3$ ops. This is not negligible with respect to the cost of a single step of the RBB algorithm. Hence algorithms without line-search are usually more effective, when the performed number of steps is the same.

6. Experiments. In this section we explore the behaviour of the Riemannian BB method (RBB) and the Riemannian BB method with nonmonotone line-search (RBB-NMLS) for the solution of various optimization problems on Riemannian manifolds.

Section 6.1 is devoted to the computation of the Karcher mean of a set of positive definite matrices. To this purpose, we consider several test matrices constructed with the random function

of the Matrix Means Toolbox ([6]). These tests have been performed using MATLAB R2011b on a machine with a Double Intel Xeon X5680 Processor with 24 GB of ram.

Experiments in Section 6.2 are carried out using Manopt 1.0.7 ([10]), a toolbox which provides a large library of manifolds and ready-to-use Riemannian optimization algorithms together with the flexibility of including user-defined solvers. The experiments are run using MATLAB R2015a on a machine with Double Intel Xeon E5-2640 v2 Processor and 128 GB of DDR3 ram.

6.1. Tests on the Karcher Mean computation. We investigate the performance of RBB (`RBB`) in comparison with four other first-order optimization algorithms for computing the Karcher mean of positive definite matrices: the Richardson iteration (`Richardson`) proposed in [5], the Majorization-Minimization algorithm (`MajMin`) of [34] and the standard Steepest-Descent (`SD`) and the Fletcher-Reeves variant of the Conjugate Gradients (`CG FR`) algorithms described in [10, 20]. Comparisons with the (Euclidean) Barzilai-Borwein algorithm (`BB NoRiem`) are also included.

For the `SD` and the `CG FR` algorithms we implemented the Riemannian Armijo's line search (3.1), where the parameters $\lambda_k = 1$, $\sigma = 0.5$ and $\gamma = 0.5$, are chosen in accordance with [20]; the maximum number of reductions has been set to 10. For the Conjugate Gradient algorithm we have chosen the Fletcher-Reeves variant because in our experience it showed lightly better results. The implementation of the standard Barzilai-Borwein algorithm (`BB NoRiem`) consists in using the Euclidean gradient as a search direction, avoiding the parallel transport and using the Euclidean scalar product in the definition of α_k; moreover, in order to ensure that all iterates are positive definite, the iteration update has been performed using the retraction, i.e. setting $X_{k+1} = R_{x_k}\exp(-\alpha_k g_k)$.

We compare the number of iterations needed for the convergence since all algorithms require roughly the same computational cost at each step, with the exception of `SD` and `CG FR` where some extra cost function evaluations are needed.

`RBB` has been implemented following Algorithm 1. We have considered only the RBB without nonmonotone line-search, since we have noticed that for this problem the line-search is not required. The first step-length α_0^{BB} for the `RBB` has been chosen using the strategy of the Richardson iteration in [5]. The step-length has been computed using (4.7) in both `RBB` and `RBB NoRiem`. The updating (4.8) and the alternating strategy for the step-length described in Remark 4.1 have been tested as well, giving essentially the same results (not report here).

As a first test, we consider the three matrices

$$A_1 = \begin{bmatrix} 1.0 & 0.2 & -0.6 \\ 0.2 & 3.1 & -0.7 \\ -0.6 & -0.7 & 1.7 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1.8 & 0.05 & 0.2 \\ 0.05 & 0.5 & -0.6 \\ 0.2 & -0.6 & 1.5 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0.8 & 0.5 & -0.5 \\ 0.5 & 1.5 & 0.2 \\ -0.5 & 0.2 & 1.4 \end{bmatrix}, \tag{6.1}$$

and we run the algorithms for the Karcher mean for a fixed number of steps and with no stopping criterion. At each step of each algorithm, we compute the relative error of the current value X_k, that is

$$\varepsilon_k = \frac{\|X_k - K\|_2}{\|K\|_2},$$

where K is a reference value of the Karcher mean obtained using variable precision arithmetic in MATLAB and rounded to 16 significant digits. The results are presented in Figure 6.1 and show that `RBB` works very well in this example and outperforms the other first-order algorithms. Notice that `BB NoRiem` does not exploit the geometric structure of the problem and then gives worse results, converging in a nonmonotone way and in a larger number of steps.

As a second test, we investigate how convergence depends on the initial value. We consider the matrices (6.1) with four different initial values: the arithmetic mean of the data, the Cheap

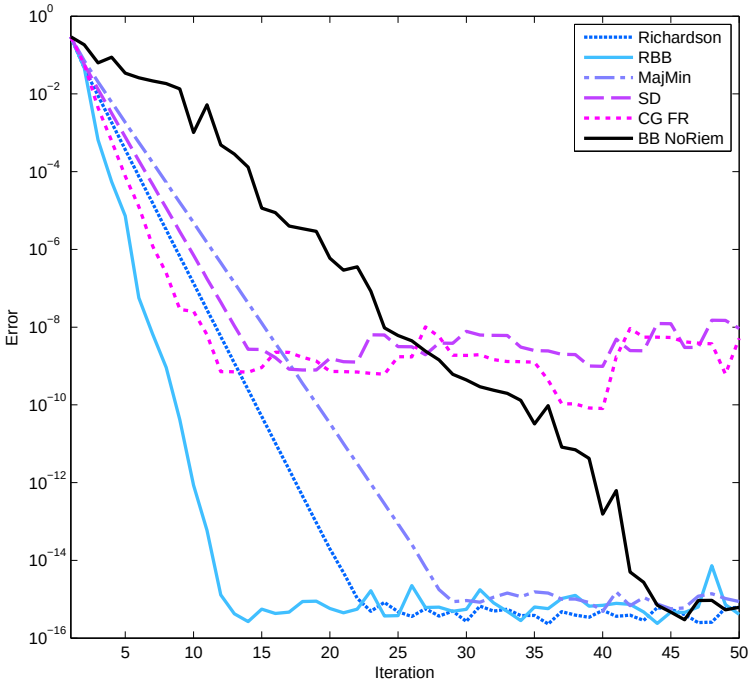

FIG. 6.1. *Comparison of different first-order Riemannian optimization methods and the (Euclidean) Barzilai-Borwein algorithm for the Karcher mean of the matrices in (6.1).*

mean of the data, known to be a good approximation of the Karcher mean ([4]), one of the data matrices and an ill-conditioned matrix of norm 1. The results are presented in Figure 6.2. The relative behaviour of the methods is mostly independent of the initial value in the later phase, while there can be some difference in the early steps. In all cases `RBB` performs better than the other methods. The step-length choice of the Richardson method is based on optimizing the rate of convergence near the fixed point (see [5]) and therefore the method is penalized whenever the initial value is far from the Karcher mean.

As a third test, we run the algorithms on 4 different data sets:

1. 100 random 10×10 matrices, obtained with the command `random(10)` of the Matrix Means Toolbox;
2. 10 random 100×100 matrices, obtained with the command `random(100)`;
3. 10 ill-conditioned 10×10 matrices, with condition number 10^5, obtained with the command `random(10,2,1e5)`;
4. 10 ill-conditioned 10×10 matrices clustered far from the identity, obtained with the command `random(10,2,1e5)*0.01+A`, where A is a fixed matrix generated as in data set 3.

In Figure 6.3, we show the relative error with respect to a reference approximation of the Karcher mean computed with variable precision arithmetic.

We observe that `RBB` convergence behaviour is always the best, while performance of `Richardson, MajMin, SD` and `CG FR` depends on the test case. In particular, failures of `SD` and `CG FR` are due to the computation of tiny steps that prevent progress in the iteration.

We remark that, since the data are randomly generated, each test has been repeated several times and we have always obtained the qualitative behavior shown in Figure 6.3.

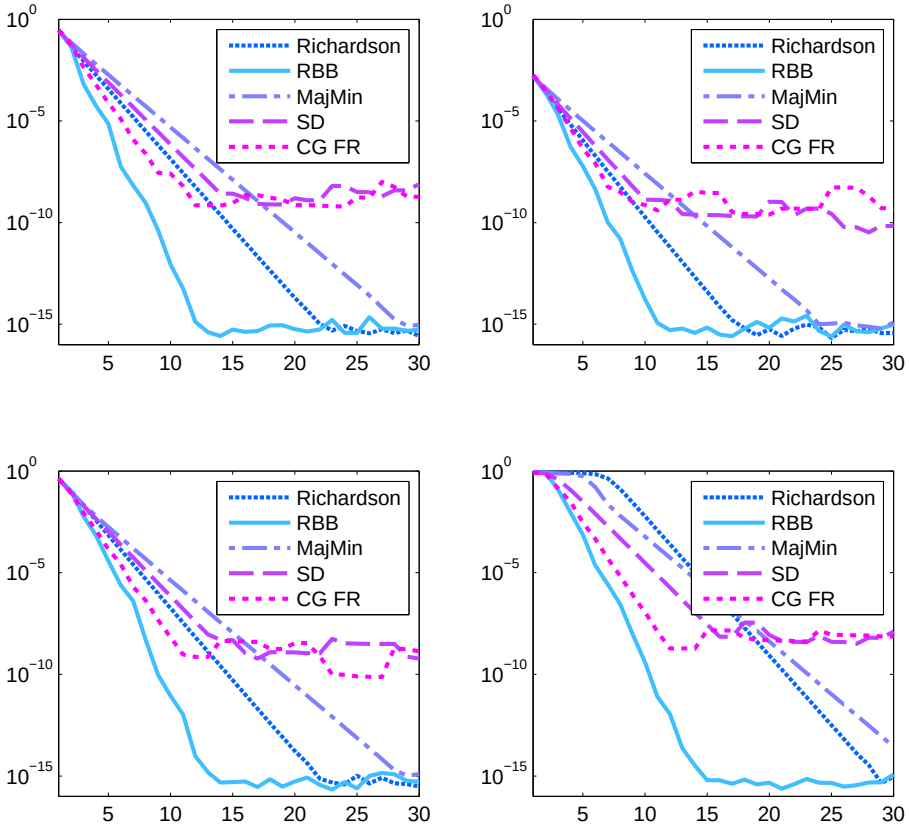

FIG. 6.2. *Comparison of different first-order optimization methods for the Karcher mean of the matrices in (6.1), using different initial values: the arithmetic mean (top-left), the Cheap mean (top-right), one of the data matrices (bottom-left) and a matrix with large condition number and unit norm (bottom-right).*

6.2. Testing RBB using Manopt. We consider the trust-region (`TR`) and steepest-descent (with monotone line-search and Armijo step-length) (`SD`) algorithms available in Manopt 1.0.7 ([10]) and, for comparison, we consider an implementation of our RBB algorithm obtained by modifying the steepest descent implementation of Manopt. This allows us to use the execution time as a fair measure of algorithmic comparison. In our tests, we use the RBB method without line-search (`RBB`) and with nonmonotone line-search (`RBB-NMLS`) as described in Algorithm 1 both using the step-length (4.7). Default parameters, stopping criteria and starting guess choices provided in Manopt have been used.

Regarding the parameter setting of `RBB-NMLS`, we set $M = 10$, $\gamma = 10^{-4}$, $\alpha_{min} = 10^{-3}$, $\alpha_{max} = 10^{3}$ and $\sigma = 0.5$ in Algorithm 1, as indicated in [8]. The vectors s_k and y_k at Step 6 are computed using the vector transport functions provided by Manopt for both `RBB` and `RBB-NMLS`. Moreover, the initial α_0^{BB} is chosen so that the first trial point $\tilde{x}_1$ is the same as the one computed by `SD`.

We consider all test problems available in [10] and follow the formulations and implementations provided therein. In particular, Table 6.1 summarizes the test problems together with a brief description and the various tested sizes.

Table 6.2 collects the results in terms of average number of iterations 'Av-it', average execution time in seconds 'Av-cpu' and number of failures 'F' computed over 10 repeated runs. In

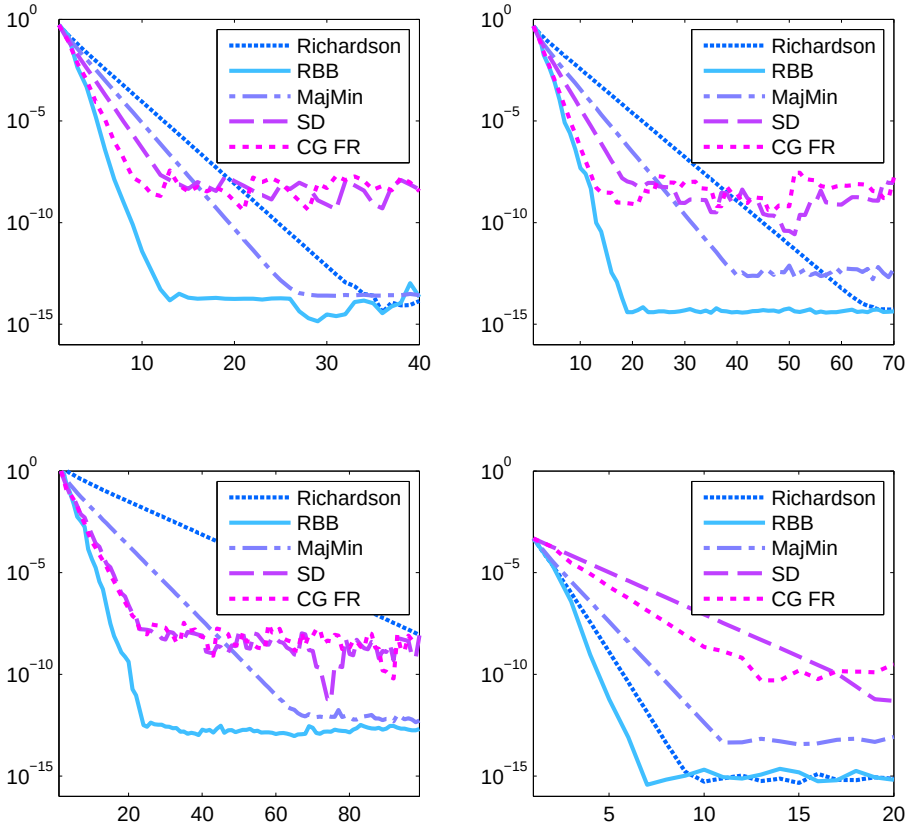

FIG. 6.3. *Comparison of different first-order optimization methods for the Karcher mean using different data sets:* 100 *random* 10×10 *matrices (top-left),* 10 *random* 100×100 *matrices (top-right),* 10 *ill-conditioned* 10×10 *matrices (bottom-left) and* 10 *ill-conditioned* 10×10 *matrices clustered far from the identity (bottom-right).*

the presence of failures within the 10 runs, the average values are computed taking into account successful runs for all solvers only. The overall number of runs is 240.

The comments below follow from Table 6.2:

- As expected, the use of the nonmonotone strategy makes the `RBB` procedure much more robust: overall `RBB` and `RBB-NMLS` fail 33 and 3 times, respectively. Clearly, when `RBB` is successful, that is, the nonmonotone strategy is not needed and therefore not activated in `RBB-NMLS`, `RBB` is faster than `RBB-NMLS` since there is no need to compute and store the cost function at each iteration.
- Focusing on first-order procedures, the use of the BB step-length significantly enhances the speed of convergence of the gradient-type algorithms: on average `SD` takes much more iterations than `RBB` and `RBB-NMLS` resulting in a much higher computational time. This behaviour is especially evident in the solution of `KM` where Av-cpu of `SD` is at least a factor 6 of the same value of `RBB`.
- Regarding the comparison of `RBB` with the second-order procedures, we note that the behaviour of `TR` and `RBB-NMLS` is comparable. For `KM` and increasing values of n, RBB is faster than `TR` since, as expected, the computation of the Hessian of the problem cost function becomes increasingly costly.

Name	Size	Description	Manifold
KM	$n = 50$ $n = 100$ $n = 200$	Computing the Karcher Mean of a set of $n \times n$ positive definite matrices.	SPD
SPCA	$n = 100, p = 10, m = 2$ $n = 500, p = 15, m = 5$ $n = 1000, p = 30, m = 10$	Computing the m Sparse Principal Components of a $p \times n$ matrix encoding p samples of n variables.	Stiefel
SP	$n = 8$ $n = 12$ $n = 24$	Finding the largest diameter of n equal circles that can be placed on the sphere without overlap (the Sphere Packing problem).	Product of Spheres
DIS	$n = 128$ $n = 500$ $n = 1000$	Finding an orthonormal basis of the Dominant Invariant 3-Subspace of an $n \times n$ matrix.	Grassmann
LRMC	$n = 100, p = 10, m = 2$ $n = 500, p = 15, m = 5$ $n = 1000, p = 30, m = 10$	Low-Rank Matrix Completion: given partial observation of an $m \times n$ matrix of rank p, attempting to complete it.	fixed-rank matrices
MC	$n = 20$ $n = 100$ $n = 300$	Max-Cut: given an $n \times n$ Laplacian matrix of a graph, finding the max-cut, or an upper-bound on the maximum-cut value.	SPD matrices of rank 2
TSVD	$n = 60, m = 42, p = 5$ $n = 100, m = 60, p = 7$ $n = 200, m = 70, p = 8$	Computing the SVD decomposition of an $m \times n$ matrix truncated to rank p.	Grassmann
GP	$m = 10, N = 50$ $m = 50, N = 100$ $m = 50, N = 500$	Generalized Procrustes: rotationally align clouds of points. Data: matrix $A \in \mathbb{R}^{3\times m\times N}$, each slice is a cloud of m points in $\mathbb{R}^3$.	Product of rotations with the Euclidean space for A

TABLE 6.1
Test problems and manifold structure provided in Manopt.

	TR			SD			RBB			RBB-NMLS		
	Av-it	Av-cpu	F	Av-it	Av-cpu	F	Av-it	Av-cpu	F	Av-it	Av-cpu	F
KM	2.0	3.3	0	11.1	28.5	0	3.0	2.3	0	3.0	3.6	0
	2.9	66.7	0	12.8	252.1	0	3.0	41.7	0	3.0	55.5	0
	4.0	1649.7	0	14.3	5875.5	0	4.0	747.7	0	4.0	1017.0	0
SPCA	13.6	0.2	0	62.6	0.4	0	57.0	0.2	0	41.3	0.3	0
	22.8	0.5	0	-	-	10	92.4	0.4	0	89.1	0.5	0
	37.1	1.2	0	-	-	10	211.7	0.9	0	210.1	1.4	0
SP	416.14	4.7	0	1175.7	5.9	0	479.8	1.6	0	641.8	3.4	0
	408.6	5.1	0	1160.2	5.9	0	738.1	2.5	0	828.6	4.1	0
	314.3	4.9	0	1615.6	8.5	0	1461.3	5.1	0	1083.6	5.7	0
DIS	10.0	0.2	0	3601.0	0.6	1	96.7	0.1	0	100.4	0.2	0
	12.2	0.4	0	-	-	10	204.2	0.3	0	290.1	0.7	0
	12.9	0.9	0	-	-	10	280.6	0.7	0	398.1	1.4	0
LRMC	6.6	0.4	0	79.9	0.7	0	44.1	0.3	0	44.1	0.4	0
	7.0	0.8	0	76.9	2.8	0	45.5	0.7	0	45.5	0.9	0
	11.0	1.8	0	68.7	2.7	0	39.8	1.0	0	39.8	1.3	0
MC	14.5	0.9	0	195.9	1.4	0	83.8	0.7	0	75.6	1.1	0
	28.8	0.8	0	747.4	4.3	5	336.8	1.4	0	235.6	1.4	0
	54.6	2.8	0	-	-	10	607.6	3.1	5	780.6	5.6	3
TSVD	15.1	1.3	0	-	-	10	177.8	1.9	4	202.5	1.4	0
	16.4	1.7	0	-	-	10	149.0	1.0	5	182.2	1.5	0
	18.0	2.02	0	-	-	10	183.0	1.3	9	225.0	1.8	0
GP	12.5	0.8	0	94.6	1.8	0	71.9	0.9	0	44.9	0.8	0
	14.4	1.0	0	45.5	1.2	0	103.4	1.6	1	30.8	0.7	0
	14.0	1.9	0	140.0	6.1	0	93.5	2.6	8	49.0	1.8	0

TABLE 6.2
Results on problems in Table 6.1.

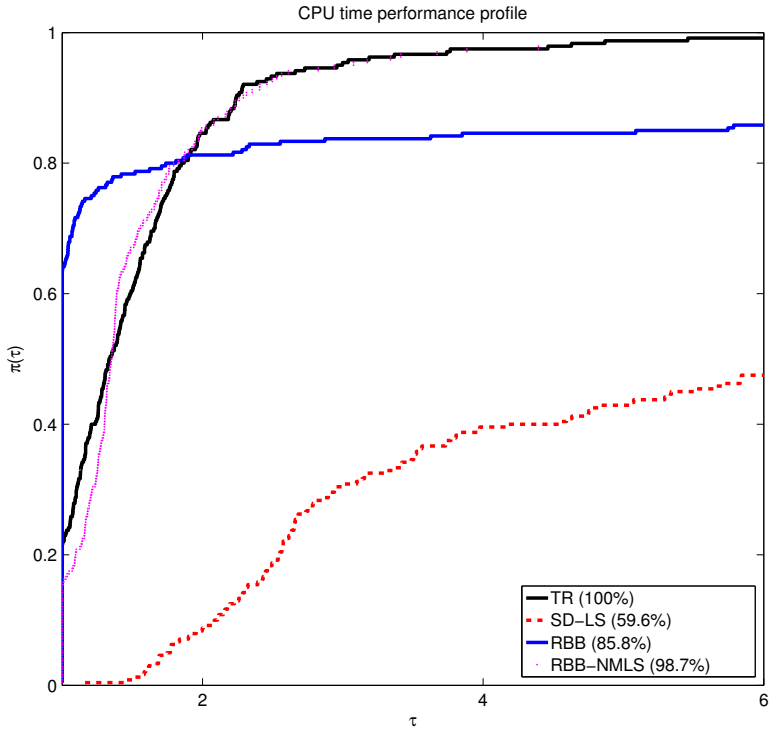

FIG. 6.4. *CPU time performance profiles on problems in Table 6.1.*

We also summarize the results plotting the performance profile function π_S defined as

$$\pi_S(\tau) = \frac{\text{number of problems s.t. } q_{P,S} \leqslant \tau\, q_P}{\text{number of problems}}, \quad \tau \geqslant 1,$$

that is the probability for solver S that a performance ratio $q_{P,S}/q_P$ is within a factor τ of the best possible ratio ([12]). Here $q_{P,S}$ denotes computational effort of the solver S to solve problem P and q_P is computational effort of the best solver to solve problem P. Note that $\pi_S(1)$ is the fraction of problems for which solver S performs the best, $\pi_S(2)$ gives the fraction of problems for which the algorithm is within a factor of 2 of the best algorithm, and that for t sufficiently large, $\pi_S(t)$ is the fraction of problems solved by S.

In Figure 6.4 we consider the total CPU time as measure of computational effort for the compared solvers and plot $\pi_S(\tau)$ for $\tau \leqslant 6$ in order to zoom the behaviour of the solver for small value of τ. Therefore, we include in the legend the percentage of tests solved by each solver (retrievable in the plot for large values of τ). We observe that `RBB` is the most efficient for the 60% of the runs, `TR` and `RBB-NMLS` are comparable in both robustness and efficiency and `SD` shows the worst performance.

7. Conclusions. We have adapted the Barzilai-Borwein method to the framework of Riemannian optimization. The resulting algorithm is competitive in several cases, since it requires just first-order information, while it converges usually faster than other first-order algorithms such as the steepest-descent method.

We have provided also a general convergence result for gradient-related Riemannian optimization algorithms, when a nonmonotone line-search is considered. This strategy, can be applied, in

particular, to guarantee global convergence of the Riemannian Barzilai-Borwein method, without spoiling, in practice, its local convergence properties.

We have observed that when the Riemannian Barzilai-Borwein algorithm is used to compute the matrix geometric mean, that is the Karcher mean of positive definite matrices, it works surprisingly well, being superior to all other first-order optimization algorithms considered in the literature. In particular, from all performed numerical tests, it seems that the algorithm converges to the matrix geometric mean with a monotonic decrease of both the error and the cost function, while the usual behaviour of the Barzilai-Borwein method is nonmonotone. In other words, the nonmonotone line-search strategy has never been required in our tests on the matrix geometric mean. At this time we do not have a theoretical justification of this phenomenon, but, in our opinion, this behaviour hides a strong convergence property which needs to be further investigated and could be the topic of a future work.

Acknowledgments. We wish to thank two anonymous referees for their comments and suggestions that greatly improved the paper. We would like to thank Valeria Simoncini for precious suggestions on a preliminary draft of this paper. The first author is indebted with Nicola Ciccoli for many useful discussions about differential geometry topics.

Funding. This work was partially supported by the Italian Istituto Nazionale di Alta Matematica (INdAM) through GNCS Projects.

REFERENCES

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*, Princeton University Press, Princeton, NJ, 2008.
[2] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
[3] R. Bhatia, *Positive Definite Matrices*, Princeton University Press, Princeton, NJ, 2007.
[4] D. Bini and B. Iannazzo, *A note on computing matrix geometric means*, Adv. Comput. Math., 35 (2011), pp. 175–192.
[5] ———, *Computing the Karcher mean of symmetric positive definite matrices*, Linear Algebra Appl., 438 (2013), pp. 1700–1710.
[6] ———, *The Matrix Means Toolbox*, Version 2.3. Retrieved on October 28, 2015 at `http://bezout.dm.unipi.it/software/mmtoolbox/`, (2015).
[7] D. A. Bini, B. Iannazzo, B. Jeuris, and R. Vandebril, *Geometric means of structured matrices*, BIT, 54 (2014), pp. 55–83.
[8] E. Birgin, J. Martínez, and M. Raydan, *Inexact spectral projected gradient methods on convex sets*, IMA J. Numer. Anal., 23 (2003), pp. 539–559.
[9] ———, *Spectral projected gradient methods: review and perspectives*, J. Stat. Softw., 60 (2014).
[10] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, *Manopt, a Matlab toolbox for optimization on manifolds*, J. Mach. Learn. Res., 15 (2014), pp. 1455–1459.
[11] Y.-H. Dai, W. W. Hager, K. Schittkowski, and H. Zhang, *The cyclic Barzilai-Borwein method for unconstrained optimization*, IMA J. Numer. Anal., 26 (2006), pp. 604–627.
[12] E. Dolan and J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[13] M. Figueiredo, R. Nowak, and S. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE J. Select. Topics Signal Process., 1 (2007), pp. 586–597.
[14] J. B. Francisco and F. S. V. Bazán, *Nonmonotone algorithm for minimization on closed sets with applications to minimization on Stiefel manifolds*, J. Comput. Appl. Math., 236 (2012), pp. 2717–2727.
[15] G. Frassoldati, L. Zanni, and G. Zanghirati, *New adaptive stepsize selections in gradient methods*, J. Ind. Manag. Optim., 4 (2008), p. 299.
[16] L. Grippo, F. Lampariello, and S. Lucidi., *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707–716.
[17] L. Grippo and M. Sciandrone, *Nonmonotone globalization techniques for the Barzilai-Borwein gradient method*, Comput. Optim. Appl., 23 (2002), pp. 143–169.

[18] N. J. HIGHAM, *Functions of matrices: Theory and computation*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008.
[19] B. IANNAZZO, *The geometric mean of two matrices from a computational viewpoint*, Numer. Linear Alg. Appl., 23 (2016), pp. 208–229.
[20] B. JEURIS, R. VANDEBRIL, AND B. VANDEREYCKEN, *A survey and comparison of contemporary algorithms for computing the matrix geometric mean*, Electron. Trans. Numer. Anal., 39 (2012), pp. 379–402.
[21] B. JIANG AND Y.-H. DAI, *A framework of constraint preserving update schemes for optimization on Stiefel manifold*, Math. Program., 153 (2015), pp. 535–575.
[22] S. LANG, *Fundamentals of differential geometry*, vol. 191 of Graduate Texts in Mathematics 191, Springer-Verlag, New York, NJ, 1999.
[23] J. D. LAWSON AND Y. LIM, *The geometric mean, matrices, metrics, and more*, Amer. Math. Monthly, 108 (2001), pp. 797–812.
[24] M. MOAKHER, *A differential geometric approach to the geometric mean of symmetric positive-definite matrices*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 735–747.
[25] Y. E. NESTEROV AND M. J. TODD, *On the Riemannian geometry defined by self-concordant barriers and interior-point methods*, Found. Comput. Math.s, 2 (2002), pp. 333–361.
[26] F. NIELSEN AND R. BHATIA, eds., *Matrix information geometry*, Springer, Heidelberg, 2013.
[27] J. NOCEDAL AND S. WRIGHT, *Numerical optimization*, Springer Science & Business Media, New York, NJ, 2nd ed., 2006.
[28] M. RAYDAN, *On the Barzilai and Borwein Choice of Steplength for the Gradient Method*, IMA J. Numer. Anal., 13 (1993), pp. 321–326.
[29] ———, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim., 7 (1997), pp. 26–33.
[30] M. RAYDAN AND B. SVAITER, *Relaxed steepest descent and cauchy-barzilai-borwein method*, Comput. Optim. Appl., 21 (2002), pp. 155–167.
[31] Q. RENTMEESTERS AND P.-A. ABSIL, *Algorithm comparison for Karcher mean computation of rotation matrices and diffusion tensors*, in Signal Processing Conference, 2011 19th European, IEEE, 2011, pp. 2229–2233.
[32] S. SRA AND R. HOSSEINI, *Conic geometric optimization on the manifold of positive definite matrices*, SIAM J. Optim., 25 (2015), pp. 713–739.
[33] Z. WEN AND W. YIN, *A feasible method for optimization with orthogonality constraints*, Math. Program., 142 (2013), pp. 397–434.
[34] T. ZHANG, *A majorization-minimization algorithm for the Karcher mean of positive definite matrices*, Preprint available at arXiv:1312.4654, (2013).