A Riemannian Limited-Memory BFGS Algorithm for Computing the Matrix Geometric Mean

Xinru Yuan[1], Wen Huang[2], P.-A. Absil[2], and K. A. Gallivan[1]

[1] Department of Mathematics, Florida State University, FL, U.S.A.
[2] ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, Belgium.

Abstract
Various optimization algorithms have been proposed to compute the Karcher mean (namely the Riemannian center of mass in the sense of the affine-invariant metric) of a collection of symmetric positive-definite matrices. Here we propose to handle this computational task with a recently developed limited-memory Riemannian BFGS method using an implementation tailored to the symmetric positive-definite Karcher mean problem. We also demonstrate empirically that the method is best suited for large-scale problems in terms of computation time and robustness when comparing to the existing state-of-the-art algorithms.

Keywords: Karcher mean, geometric mean, symmetric positive-definite matrices, Riemannian optimization, Limited-memory Riemannian BFGS method

1 Introduction

Symmetric positive-definite (SPD) matrices have become fundamental computational objects in many areas. For example, they appear as data points in medical imaging [8, 23], image processing [9, 22], radar signal processing [19, 5] and elasticity [21]. In these and similar applications, it is often of interest to average SPD matrices. Averaging is required, e.g., to aggregate several noisy measurements of the same object. It also appears as a subtask in interpolation methods, see for example [1].

Let $\mathcal{S}_{++}^n$ be the manifold of $n \times n$ SPD matrices. Since $\mathcal{S}_{++}^n$ is an open submanifold of the vector space of $n \times n$ symmetric matrices, its tangent space at point X—denoted by $\mathrm{T}_X\, \mathcal{S}_{++}^n$—can be identified as the set of $n \times n$ symmetric matrices. The manifold $\mathcal{S}_{++}^n$ becomes a Riemannian manifold when endowed with the affine-invariant metric, see [22], given by

$$g_X(\xi_X, \eta_X) = \mathrm{trace}(\xi_X X^{-1} \eta_X X^{-1}). \tag{1}$$

Given a collection of points $\{A_1, \ldots, A_K\}$ on the Riemannian manifold $\mathcal{S}_{++}^n$, their Riemannian center of mass—also termed *Karcher mean* in view of [18]—is the minimizer of the sum of

squared distance

$$\mu = \operatorname*{arg\,min}_{X \in \mathcal{S}_{++}^n} F(X), \quad \text{with } F : P_n^+ \to \mathbb{R},\ X \mapsto \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{2}$$

where $\delta(p, q) = \| \log(p^{-1/2} q p^{-1/2}) \|_{\mathrm{F}}$ is the geodesic distance associated with Riemannian metric (1). In [18] it is proved that function F has a unique minimizer. Hence a point $\mu \in \mathcal{S}_{++}^n$ is a Karcher mean if it is a stationary point of F, i.e.,

$$\operatorname{grad} F(\mu) = 0, \tag{3}$$

where $\operatorname{grad} F$ denotes the Riemannian gradient of F under metric (1).

The Karcher mean has been recognized as one of the most suitable means on $\mathcal{S}_{++}^n$ in the sense that it satisfies all of the geometric properties given in the ALM list [4], which are known to be important in practical applications, e.g. [20, 21, 6]. However, a closed-form solution for problem (2) or equation (3) is unknown in general, and for this reason, the Karcher mean is usually computed by iterative methods. A Richardson-like iteration is derived and evaluated empirically in [7], and is available in the Matrix Means Toolbox.[1] In fact, solving problem (2) can be considered in the framework of Riemannian optimization (see, e.g., [2]), since it requires optimizing a function on a manifold. This approach gave rise to several algorithms for the Karcher mean computation. A constant stepsize steepest descent method with local convergence is proposed in [3]. An adaptive stepsize selection rule based on the explicit expression of the Hessian of cost function F and a Newton's method are studied in [25]. Most recently, [17] presents a survey of several optimization algorithms, including Riemannian version of steepest descent, conjugate gradient, BFGS, and trust-region Newton's method. It is empirically demonstrated that the steepest descent and conjugate gradient methods are the preferred choices for problem (2) in terms of time efficiency. The benefit of fast convergence of Newton's method and BFGS is nullified by their high computational cost in each iteration, especially when the size of the matrices increases. There are however recently-developed general-purpose Riemannian optimization methods that have not yet been exploited in the context of SPD Karcher mean computation; this is the case in particular of the limited-memory BFGS that is known to be particularly efficient in various contexts [15, 26] but that has not yet been exploited in the context of the SPD Karcher mean computation.

In this paper, we exploit a limited-memory Riemannian BFGS (LRBFGS) method developed in [14] to tackle the SPD Karcher mean computation problem. An abstract formulation of this method is provided in [14, Section 5], while its practical implementation requires more work than one might anticipate. This paper contributes a performance-driven development that turns the abstract LRBFGS into a concrete and efficient numerical algorithm for the SPD Karcher mean computation. More specifically, the LRBFGS method involves manipulation of differential geometric objects on manifolds, such as tangent space, Riemannian metric, retraction, vector transport, etc. We present an approach to produce efficient numerical representations of those objects on the manifold $\mathcal{S}_{++}^n$. Those techniques are also applied to our implementation of Riemannian steepest descent and conjugate gradient methods. It is shown empiricially that the LRBFGS method is best suited for computing the SPD Karcher mean when the size or number of matrices increases.

This paper is organized as follows. Section 2 presents a brief analysis on the conditioning of the problem. LRBFGS method is stated in Section 3. Section 4 presents the implementation techniques for $\mathcal{S}_{++}^n$. Numerical experiments are reported in Section 5.

[1] `http://bezout.dm.unipi.it/software/mmtoolbox/`

2 Conditioning of the sum of squared distances function

Methods such as BFGS that gather second-order information are expected to dramatically outperform steepest descent methods when the Hessian of the objective function is very ill-conditioned at the minimizer. Indeed, steepest descent methods are notoriously slow in such a situation due to the hemstitching phenomenon. In this section, we show that the condition number of the Hessian of cost function F in (2) admits an upper bound that behaves like the logarithm of the largest condition number of the data points. Hence a large condition number cannot be achieved in practice for the Hessian of F. This result sheds new light on the finding in [17] that BFGS and Newton's method do not outperform the steepest descent approach. Nevertheless, we will show numerically in Section 5 that LRBFGS tends to outperform steepest descent and conjugate gradient methods in various situations.

Rentmeesters et al. [25] derived bounds on the eigenvalues of the Hessian of the squared distance function $f_A(X) = \frac{1}{2}\delta(X, A)^2$. Notice that Riemannian metric (1) is invariant under the isometry $y \mapsto L^{-1}yL^{-T}$ where $X = LL^T$, and so is function $f_A(X)$. As a result, without loss of generality we can assume $X = I$. The bounds for the Hessian of $f_A(X)$ at $X = I$ are given by, see [25],

$$1 \leq \frac{\operatorname{Hess} f_A(X)[\Delta X, \Delta X]}{\|\Delta X\|^2} \leq \frac{\log \kappa}{2} \coth\left(\frac{\log \kappa}{2}\right), \tag{4}$$

where κ is the condition number of A and log stands for the natural logarithm.

We are interested in the conditioning of problem (2) at the minimizer, i.e., the Karcher mean—denoted by μ—of data points $\{A_1, \ldots, A_K\}$. Again without loss of generally, we can assume $\mu = I$. Otherwise, we can "translate" the problem to identity using an isometry. It is straightforward to generalize the bounds for the Hessian of our cost function F from inequality (4) since $F(X) = \frac{1}{2K}\sum_{i=1}^{K} f_{A_i}(X)$. At the Karcher mean μ, we have

$$\begin{aligned} 1 \leq \frac{\operatorname{Hess} F(\mu)[\Delta\mu, \Delta\mu]}{\|\Delta\mu\|^2} &\leq \frac{1}{K}\sum_{i=1}^{K} \frac{\log \kappa_i}{2} \coth\left(\frac{\log \kappa_i}{2}\right) \\ &\leq \max_i \left[\frac{\log \kappa_i}{2} \coth\left(\frac{\log \kappa_i}{2}\right)\right] \qquad (5) \\ &\leq 1 + \frac{\log(\max_i \kappa_i)}{2}, \qquad (6) \end{aligned}$$

where κ_i is the condition number of A_i. We are using the fact that function $x \coth(x)$ is strictly increasing and bounded by $1 + x$ on $[0, \infty]$ in (5) to get (6). For example, assume that there is at least one very ill-conditioned point in set $\{A_1, \ldots, A_K\}$, with condition number 10^{10}. From (6) the upper bound on the condition number of the Hessian at the minimizer is given by $1 + \log(10^{10})/2 \approx 12.51$. Based on above discussion, we are ready to conclude that we cannot expect a very ill-conditioned Hessian at the minimizer in practice.

3 Brief description of limited-memory Riemannian BFGS

We start the description of the LRBFGS method—which we adapt to problem (2) in the next section—by recalling its foundation, the RBFGS method. More details can be found in [14]. The methods proposed in [14] are retraction-based line-search methods, namely, the iterate x_k on the manifold $\mathcal{M}$ is updated by

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k), \tag{7}$$

where R is a retraction on $\mathcal{M}$, $\eta_k \in \mathrm{T}_{x_k}\mathcal{M}$ is the search direction and $\alpha_k \in \mathbb{R}$ denotes the stepsize. A definition of retraction appears in [2, Definition 4.1.1]. The search direction in (7) is $\eta_k = -\mathcal{B}_k^{-1} \operatorname{grad} f(x_k)$, where $\mathcal{B}_k$ is a linear operator that approximates the action of the Hessian on $\mathrm{T}_{x_k}\mathcal{M}$. More specifically, $\mathcal{B}_k$ incurs a rank-two update at each iteration, see [14, Algorithm 1] for the update formula. In addition, the RBFGS method requires the Riemannian manifold $\mathcal{M}$ to be equipped with a vector transport $\mathcal{T}$. A vector transport $\mathcal{T} : \mathrm{T}\mathcal{M} \oplus \mathrm{T}\mathcal{M} \to \mathrm{T}\mathcal{M}$, $(\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}\xi_x$ with associated retraction R is a smooth mapping such that, for all (x, η_x) in the domain of R and all ξ_x, $\zeta_x \in \mathrm{T}_x\mathcal{M}$, it holds that (i) $\mathcal{T}_{\eta_x}\xi_x \in \mathrm{T}_{R(\eta_x)}\mathcal{M}$, (ii) $\mathcal{T}_{0_x}\xi_x = \xi_x$, (iii) $\mathcal{T}_{\eta_x}$ is a linear map. Additionally, the RBFGS method in [14, Algorithm 1] requires the vector transport, denoted by $\mathcal{T}_S$, to be isometric, i.e., $g_{R(\eta_x)}(\mathcal{T}_{S_{\eta_x}}\xi_x, \mathcal{T}_{S_{\eta_x}}\zeta_x) = g_x(\xi_x, \zeta_x)$. Throughout the paper, we use the notation $\mathcal{T}_S$ for isometric vector transport.

However, for large-scale problems, the RBFGS method may require excessive computation time and storage space, since it stores and transports linear operator $\mathcal{B}_k^{-1}$ as a dense matrix. A well-known way to remedy this drawback is the limited-memory version of BFGS, which stores only the m most recent rank-two updates of $\mathcal{B}$. A Riemannian version of the limited-memory BFGS was proposed in [14, Algorithm 2]. Our actual instantiation depends on this algorithm. For the SPD Karcher mean problem, we modify this version to use an alternate update in [13] that allows the line search using the Riemannian Wolfe conditions to be replaced by the Armijo line search addressed in [2, Algorithm 1].

4 Implementation

This section presents the implementation details of all the required objects for LRBFGS on the SPD Karcher mean computation problem, including a tangent vector, Riemannian metric, vector transport, retraction, and Riemannian gradient of the objective function.

4.1 Representations of a tangent vector and the Riemannian metric

As noted in the introduction, the tangent space to $\mathcal{S}_{++}^n$ at X can be identified as the set of symmetric matrices. Thus, $\mathcal{S}_{++}^n$ is a d-dimensional submanifold of an n^2-dimensional Euclidean space $\mathcal{E}$, where $d = n(n+1)/2$. A tangent vector in $\mathrm{T}_X\mathcal{S}_{++}^n$ can be represented either by an n^2-dimensional vector in $\mathcal{E}$, or a d-dimensional vector of coordinates in a given basis B_X of $\mathrm{T}_X\mathcal{S}_{++}^n$. The n^2-dimensional representation is called the extrinsic approach, and the d-dimensional one is called the intrinsic approach. The computational benefits of using intrinsic representation are addressed in [11, 12]: (i) The Riemannian metric is reduced to the Euclidean metric. (ii) There exists an isometric vector transport, called vector transport by parallelization, whose intrinsic implementation is simply the identity. (iii) Working in d-dimensions reduces the computational complexity of linear operations on the tangent space. However, the intrinsic representation requires a basis of the tangent space, and in order to obtain those computational benefits, it must be orthonormal. Hence, if a manifold admits a smooth field of orthonormal tangent space bases with acceptable computationally complexity, the intrinsic representation often leads to a very efficient implementation. This property holds for manifold $\mathcal{S}_{++}^n$ as shown next.

The orthonormal basis of $\mathrm{T}_X\mathcal{S}_{++}^n$ that we select is given by $B_X = \{Le_ie_i^TL^T : i = 1, \ldots, n\} \cup \{\frac{1}{\sqrt{2}}L(e_ie_j^T + e_je_i^T)L^T, i < j, i = 1, \ldots, n, j = 1, \ldots, n\}$, where $X = LL^T$ is the Cholesky decomposition, and $\{e_i, \ldots, e_n\}$ is the standard basis of n-dimensional Euclidean space. It is easy to verify the orthonormality of B_X under Riemannian metric (1), i.e., $B_X^\flat B_X = I_{d\times d}$ for all X. (The notation $a^\flat$ denotes the function $a^\flat : \mathrm{T}_X\mathcal{M} \to \mathbb{R} : v \mapsto g_X(a, v)$, where g stands for the affine-invariant metric (1).) We assume throughout the paper that B_X stands for our

selected orthonormal basis of $\mathrm{T}_X\,\mathcal{S}_{++}^n$. Given $\xi_X, \eta_X \in \mathrm{T}_X\,\mathcal{S}_{++}^n$, their intrinsic representations are $u_X = B_X^\flat \xi_X$ and $v_X = B_X^\flat \eta_X$ respectively. In practice, the d-dimensional representation of tangent vector $\xi_X \in \mathrm{T}_X\,\mathcal{S}_{++}^n$ is obtained by taking the diagonal elements of $L^{-1}\xi_X L^{-T}$, and its upper triangular elements row-wise and multiplied by $\sqrt{2}$. The intrinsic representation of the Riemannian metric (1) is then given by

$$\tilde{g}(u_X, v_X) := g_X(B_X u_X, B_X v_X) = u_X^T v_X.$$

A detailed proof can be found in [11, Section 9.2.1].

4.2 Retraction and vector transport

The choice of retraction and vector transport is a key step in the design of efficient Riemannian optimization algorithms. The exponential mapping is a natural choice for retraction. When $\mathcal{S}_{++}^n$ is endowed with the affine-invariant Riemannian metric (1), the exponential mapping is given by, see [10],

$$\mathrm{Exp}_X(\xi_X) = X^{1/2}\exp(X^{-1/2}\xi_X X^{-1/2})X^{1/2}, \tag{8}$$

for all $X \in \mathcal{S}_{++}^n$ and $\xi_X \in \mathrm{T}_X\,\mathcal{S}_{++}^n$. In practice, the exponential mapping (8) is expensive to compute. More importantly, when computing the matrix exponential $\exp(B)$, eigenvalues of matrix B with large magnitude can lead to numerical difficulties, such as overflow. Jeuris et al. [17] proposed a retraction

$$R_X(\xi_X) = X + \xi_X + \frac{1}{2}\xi_X X^{-1}\xi_X, \tag{9}$$

which is a second order approximation to (8). Retraction (9) is cheaper to compute and tends to avoid numerical overflow. Additionally, $R_X(\xi_X)$ remains symmetric positive-definite for all $X \in \mathcal{S}_{++}^n$ and $\xi_X \in \mathrm{T}_X\,\mathcal{S}_{++}^n$.

Parallel translation is a particular instance of vector transport. The parallel translation on $\mathcal{S}_{++}^n$ is given by, see [10],

$$\mathcal{T}_{p_{\eta_X}}(\xi_X) = X^{1/2}\exp(\frac{X^{-1/2}\eta_X X^{-1/2}}{2})X^{-1/2}\xi_X X^{-1/2}\exp(\frac{X^{-1/2}\eta_X X^{-1/2}}{2})X^{1/2}. \tag{10}$$

Unfortunately, the implementation of parallel translation (4.2) is computationally expensive. We will thus resort to another vector transport, as follows.

Recently, Huang et al. [12] proposed a novel way to construct an isometric vector transport, called vector transport by parallelization. For all $X \in \mathcal{S}_{++}^n$ and $\xi_X,\ \eta_X \in \mathrm{T}_X\,\mathcal{S}_{++}^n$, the vector transport by parallelization is given by

$$\mathcal{T}_{S_{\eta_X}}\xi_X = B_Y B_X^\flat \xi_X, \tag{11}$$

where $Y = R_X(\eta_X)$. In fact, the intrinsic representation of this vector transport is simply the identity, i.e.,

$$\mathcal{T}_S^d u_X = u_X, \tag{12}$$

where $\mathcal{T}_S^d$ denotes the intrinsic approach of $\mathcal{T}_S$ and u_X is the intrinsic representation of ξ_X. The derivation of (12) can be found in [11, Section 9.5.2].

Another possible choice for the vector transport is the identity: $\mathcal{T}_{id_{\eta_X}}(\xi_X) = \xi_X$. However, vector transport $\mathcal{T}_{id}$ is not applicable to the LRBFGS method since it is not isometric under Riemannian metric (1).

Our implementation of LRBFGS for the SPD Karcher mean computation uses retraction (9) and the intrinsic approach of vector transport (12).

4.3 Riemannian gradient of the sum of squared distances function

The Riemannian gradient of cost function F in (2) is given by, see [18],

$$\operatorname{grad} F(X) = -\frac{1}{K}\sum_{i=1}^{K} \operatorname{Exp}_X^{-1}(A_i), \tag{13}$$

where $\operatorname{Exp}_x^{-1}(y)$ is the log-mapping, i.e., the inverse exponential mapping. On $\mathcal{S}_{++}^n$, the log-mapping is computed as

$$\operatorname{Exp}_X^{-1}(\xi_X) = X^{1/2}\log(X^{-1/2}\xi_X X^{-1/2})X^{1/2} = \log(\xi_X X^{-1})X. \tag{14}$$

Note that the computational complexity of the Riemannian gradient is less than that conveyed in formula (14) since the most expensive logarithm computation is already available from the evaluation of the cost function at X. Specifically, each term in (13) is computed as $-\operatorname{Exp}_X^{-1}(A_i) = -\log(A_i X^{-1})X = \log(XA_i^{-1})X = A_i^{-1/2}\log(A_i^{-1/2}XA_i^{-1/2})A_i^{1/2}X^{-1}$, and the term $\log(A_i^{-1/2}XA_i^{-1/2})$ is available from the evaluation of the cost function $F(X)$.

We have thus gathered all the necessary ingredients to implement the LRBFGS method of [13] for the SPD Karcher mean problem (2).

5 Experiments

We compare the performance of the LRBFGS method described earlier and existing state-of-the-art methods, including the Riemannian steepest descent (RSD) [2, Page 62], the Riemannian conjugate gradient (RCG) [2, Algorithm 13], and the Richardson-like iteration (RL) [7]. In particular, we use the implementation of the RL iteration in Bini et al.'s Matrix Means Toolbox.[1]

In our practical implementation, the stepsizes in LRBFGS and RSD are selected with a line search method that satisfies the Armijo condition [2, Algorithm 1], with Armijo parameter $c_1 = 10^{-4}$, backtracking reduction factor $\gamma = 0.25$, and initial stepsize $\alpha = 2/(1+L)$ in [24] where L is the upper bound at the initial iterate defined in inequality (6). The line search procedure in RCG enforces the strong Wolfe condition [27, Algorithm 3.5] with constant $c_1 = 10^{-4}$ and $c_2 = 0.999$. In LRBFGS, we set the memory size m to 2. The d-dimensional intrinsic representation is used for tangent vectors. Retraction (9) and the vector transport by parallelization (12) are used. When the iterate is close enough to the minimizer (in our experiments, when the norm of the Riemannian gradient is less than 10^{-4}), in order to achieve sufficient accuracy, we switch the line search procedure to another stepsize selection strategy. For RSD and RCG, the choice of stepsize follows the rule introduced in [24, Section 3.6]. For LRBFGS, the stepsize is set to 1. Our choice of the initial iterate is the Arithmetic-Harmonic mean [16] of data points. We run the algorithms until they reach their highest accuracy.

For simplicity of notation, throughout this section we denote the number, dimension, and condition number of the matrices by K, n, and κ respectively. For each choice of (K, n) and the range of conditioning desired, a single experiment comprises generating 5 sets of K random $n \times n$ matrices with appropriate condition numbers, and running all 4 algorithms on each set with the initial iterate described earlier. The result of the experiment is the distance to the true Karcher mean averaged over the 5 sets as a function of iteration and time. All experiments are performed using MATLAB R2014b in standard double precision arithmetic on a MAC OS X platform with a 1.8 GHz CPU.

5.1 Experiment design

When defining each set of experiments, we choose a desired (true) Karcher mean μ, and construct data matrices A_i's such that their Karcher mean is exactly μ, i.e., equation $\sum_{i=1}^{K} \mathrm{Exp}_{\mu}^{-1}(A_i) = 0$ holds. The benefits of this scheme are: (i) We can control the conditioning of μ and A_i's, and observe the influence of the conditioning on the performance of algorithms. (ii) Since the true Karcher mean μ is known, we can monitor the distance δ between μ and the iterates produced by various algorithms, thereby removing the need to consider the effects of a terminating criteria.

Given a Karcher mean μ, the A_i's are constructed as follows: (i) Generate W_i in Matlab as follows, with n being the size of matrix, f the order of magnitude of the condition number, and p some number less than n,

```
[O, ~] = qr(randn(n));
D = diag([rand(1,p)+1, (rand(1,n-p)+1)*10^(-f)]);
W = O * D * O'; W = W/norm(W,2);
```

(ii) Compute $\eta_i = \mathrm{Exp}_{\mu}^{-1}(W_i)$. (iii) Enforce the condition $\sum_{i=1}^{K} \eta_i = 0$ on η_i's. Specifically, we test on data sets with $K = 3,\ 30,\ 100$. In the case of $K = 3$, we enforce $\eta_3 = -\eta_1 - \eta_2$. When $K = 30$ or $K = 100$, let $k_i = 5(k-1) + i$ for $1 \le k \le K/5$ and $1 \le i \le 5$. We enforce $\eta_{k_4} = -\eta_{k_1} - 0.5\eta_{k_3}$ and $\eta_{k_5} = -\eta_{k_2} - 0.5\eta_{k_3}$, which gives $\sum_{i=1}^{5} \eta_{k_i} = 0$, and thus $\sum_{k=1}^{K/5} \sum_{i=1}^{5} \eta_{k_i} = 0$. (iv) Compute $A_i = \mathrm{Exp}_{\mu}(\eta_i)$.

It should be pointed out that instead of producing η_i directly, we produce W_i first and obtain η_i from the log-mapping, since this procedure gives us greater control over the conditioning of data points. As discussed in Section 2, we can take the Karcher mean μ as the identity matrix in numerical experiments, since we can "translate" the data set $\{\mu, A_1, \ldots, A_K\}$ to $\{I, L^{-1}A_1L^{-T}, \ldots, L^{-1}A_KL^{-T}\}$ using an isometry, where $\mu = LL^T$.

5.2 Numerical results

We now compare the performances of all 4 algorithms on various data sets by examining performance results from representative experiments for various choices of (K, n) and conditioning. The Matlab code that generated the figures is available from `http://www.math.fsu.edu/~whuang2/papers/ARLBACMGM.htm`.

Figure 1 displays the performance results of different algorithms running on well-conditioned data set with small K and n. Specifically, we take $K = 3$, $n = 3$, and $1 \le \kappa(A_i) \le 10$. It is seen that LRBFGS converges the fastest in terms of number of iterations, but is outperformed by RL in terms of computation time. Note that our LRBFGS Matlab library is designed to be user friendly, hence for small-scale problems, the Matlab library machinery dominates the computation time. Thus the findings on small-scale problems may be quite different if Matlab is replaced by a compiled language such as C++. This is no longer the case for the large-scale problems considered next, as it is then the BLAS calls that dominate the computation time.

Figure 2 and Figure 3 report the results of tests conducted on data sets with large K ($K = 100,\ n = 3$) and large n ($K = 30,\ n = 100$) respectively. Note that when $n = 100$, the dimension of manifold $\mathcal{S}_{++}^{n}$ is $d = n(n+1)/2 = 5050$. In each case, both well-conditioned and ill-conditioned data sets are tested. From Figure 2 and Figure 3, we observe that LRBFGS outperforms the state-of-the-art algorithms in term of computation time and number of iterations per unit of accuracy required. On a well-conditioned data set, RSD, RCG and RL require similar numbers of iterations to achieve the same accuracy, while on an ill-conditioned data set, RL requires significantly more iterations.

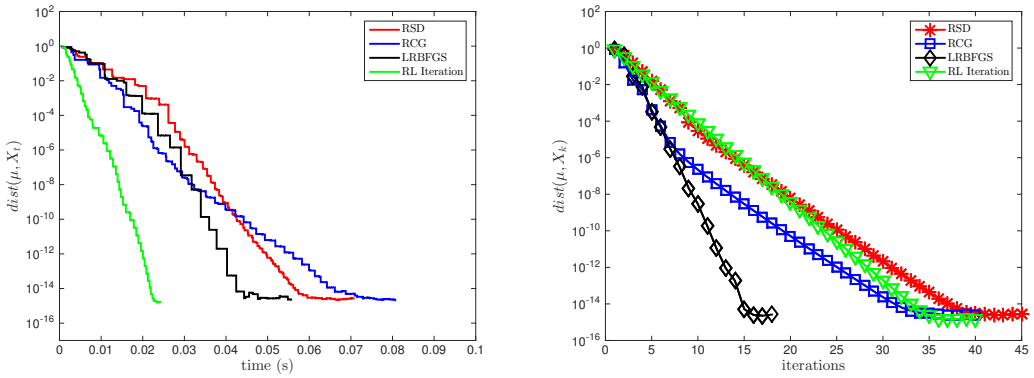

Figure 1: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations with $K = 3$, $n = 3$, and $1 \leq \kappa(A_i) \leq 20$.

References

[1] P.-A. Absil and Pierre-Yves Gousenbourger. Differentiable piecewise-Bezier surfaces on Riemannian manifolds. Technical report, ICTEAM Institute, Universite catholique de Louvain, 2015.

[2] P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[3] Bijan Afsari, Roberto Tron, and René Vidal. On the convergence of gradient descent for finding the Riemannian center of mass. *SIAM Journal on Control and Optimization*, 51(3):2230–2260, 2013.

[4] T Ando, Chi-Kwong Li, and Roy Mathias. Geometric means. *Linear algebra and its applications*, 385:305–334, 2004.

[5] F Barbaresco. Innovative tools for radar signal processing based on Cartans geometry of SPD matrices & information geometry. In *Radar Conference, 2008. RADAR'08. IEEE*, pages 1–6. IEEE, 2008.

[6] Rajendra Bhatia and Rajeeva L Karandikar. Monotonicity of the matrix geometric mean. *Mathematische Annalen*, 353(4):1453–1467, 2012.

[7] Dario A Bini and Bruno Iannazzo. Computing the Karcher mean of symmetric positive definite matrices. *Linear Algebra and its Applications*, 438(4):1700–1710, 2013.

[8] Guang Cheng, Hesamoddin Salehian, and Baba C Vemuri. Efficient recursive algorithms for computing the mean diffusion tensor and applications to DTI segmentation. In *Computer Vision–ECCV 2012*, pages 390–401. Springer, 2012.

[9] P Thomas Fletcher and Sarang Joshi. Riemannian geometry for the statistical analysis of diffusion tensor data. *Signal Processing*, 87(2):250–262, 2007.

[10] P Thomas Fletcher, Conglin Lu, Stephen M Pizer, and Sarang Joshi. Principal geodesic analysis on symmetric spaces: statistics of diffusion tensors. *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis*, 3117:87–98, 2004.

[11] Wen Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, Dept. of Mathematics, Florida State University, 2012.

[12] Wen Huang, P.-A. Absil, and K A Gallivan. A Riemannian symmetric rank-one trust-region method. *Mathematical Programming*, 2014.

[13] Wen Huang, P.-A. Absil, and K. A. Gallivan. A Riemannian BFGS method for nonconvex optimization problems. Technical Report UCL-INMA-2015.11, U.C.Louvain, 2015.

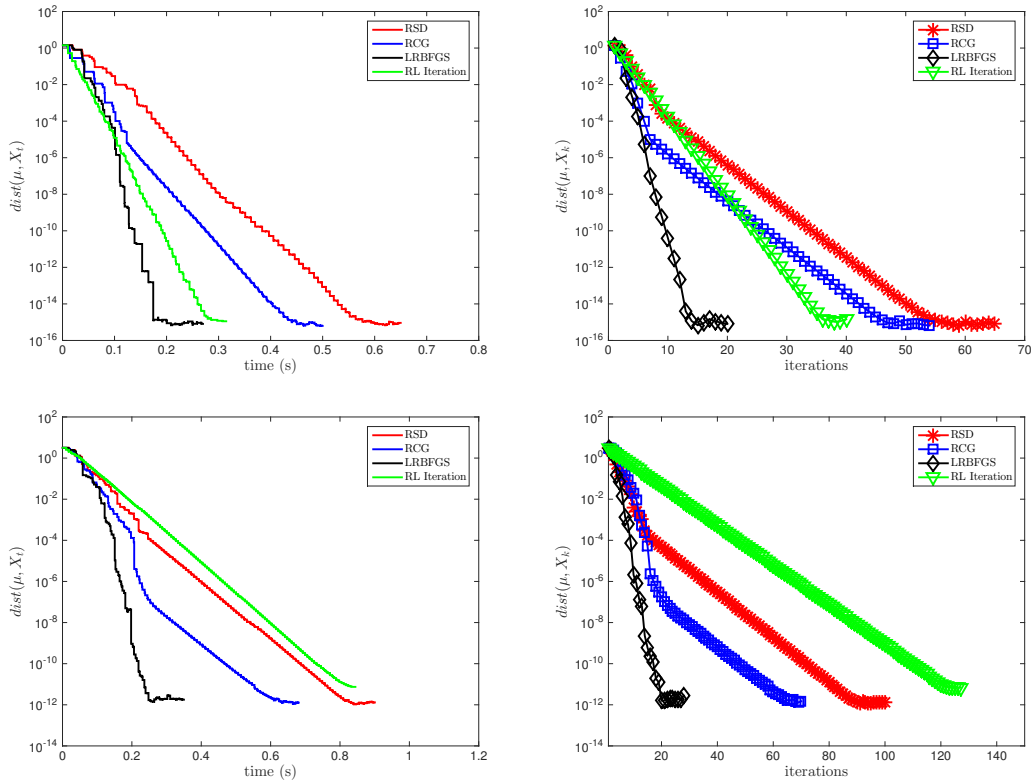

Figure 2: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations with $K = 100$ and $n = 3$; Top: $1 \leq \kappa(A_i) \leq 200$; Bottom: $10^3 \leq \kappa(A_i) \leq 2 \cdot 10^6$

[14] Wen Huang, K. A. Gallivan, and P.-A. Absil. A Broyden class of quasi-Newton methods for Riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[15] Wen Huang, Kyle A Gallivan, Anuj Srivastava, Pierre-Antoine Absil, et al. Riemannian optimization for elastic shape analysis. In *Proceedings of the 21st Internaltional Symposium on Mathematical Theory of Networks and Systems (MTNS 2014)*, 2014.

[16] Ben Jeuris and Raf Vandebril. Geometric mean algorithms based on harmonic and arithmetic iterations. In *Geometric Science of Information*, pages 785–793. Springer, 2013.

[17] Ben Jeuris, Raf Vandebril, and Bart Vandereycken. A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *Electronic Transactions on Numerical Analysis*, 39(EPFL-ARTICLE-197637):379–402, 2012.

[18] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 1977.

[19] J Lapuyade-Lahorgue and F Barbaresco. Radar detection using Siegel distance between autoregressive processes, application to HF and X-band radar. In *Radar Conference, 2008. RADAR'08. IEEE*, pages 1–6. IEEE, 2008.

[20] Jimmie Lawson and Yongdo Lim. Monotonic properties of the least squares mean. *Mathematische Annalen*, 351(2):267–279, 2011.

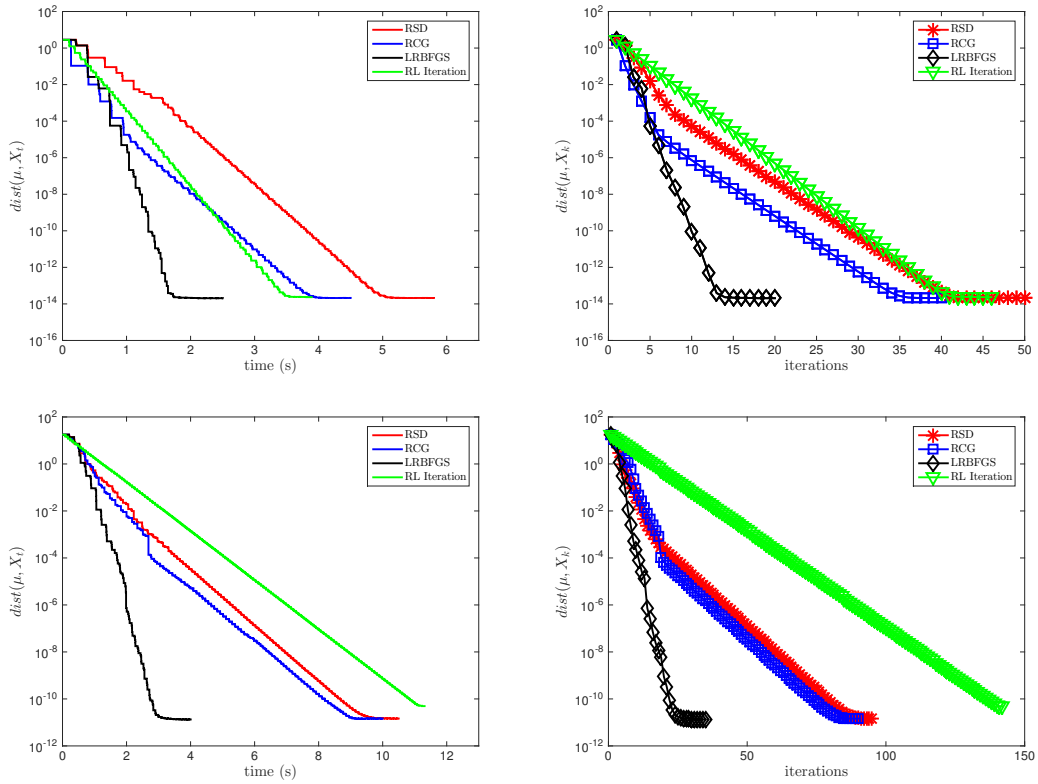

Figure 3: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations with $K = 30$ and $n = 100$; Top: $1 \leq \kappa(A_i) \leq 20$; Bottom: $10^4 \leq \kappa(A_i) \leq 2 \cdot 10^6$

[21] Maher Moakher. On the averaging of symmetric positive-definite tensors. *Journal of Elasticity*, 82(3):273–296, 2006.

[22] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[23] Yogesh Rathi, Allen Tannenbaum, and Oleg Michailovich. Segmenting images on the tensor manifold. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[24] Quentin Rentmeesters. *Algorithms for data fitting on some common homogeneous spaces*. PhD thesis, UCL, 2013.

[25] Quentin Rentmeesters and P.-A. Absil. Algorithm comprison for Karcher mean computation of rotation matrices and diffusion tensors. *19th European Signal Processing Conference (EUSIPCO 2011)*, 2011.

[26] V Schulz, Martin Siebenborn, and Kathrin Welker. Structured inverse modeling in parabolic diffusion problems. *arXiv preprint arXiv:1409.3464*, 2014.

[27] Stephen J Wright and Jorge Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.