Approximate matrix geometric means based on the inductive mean *

Estelle M. Massart† Julien M. Hendrickx† P.-A. Absil†

October 13, 2015

Abstract

We propose a new algorithm to approximate the Karcher mean of N symmetric positive definite (SDP) matrices. By "Karcher mean", we refer to the Riemannian center of mass with respect to the natural metric (also known as the trace metric or affine-invariant metric) on the space $\mathbb{P}_n$ of $n \times n$ SDP matrices. The approximation we propose compares favorably with state-of-the-art methods according to the accuracy vs computation time criterion. It has also the advantage of relying only on the geometric mean of two matrices and requires therefore fewer tools than most of the algorithms presently used to compute the Karcher mean (e.g., optimization algorithms). We study numerically the evolution of the computation time and accuracy of the proposed approximation with respect to some key parameters of the problem such as the number of matrices, their size, and their condition number.

1 Introduction

We consider the problem of efficiently computing an approximate geometric mean of a set of N symmetric positive definite (SPD) matrices. This is motivated by the sizeable computation cost of existing methods for computing an exact geometric mean of SPD matrices.

Averaging SPD matrices is important in various applications. In medical imaging for example, image segmentation techniques applied to diffusion tensor imaging require averaging diffusion tensors, which are specific instances of SPD matrices [CSV12]. In mechanics, the elasticity tensor of a material, which can be rewritten as a SPD matrix, is usually estimated on an experimental basis; repeating the experiment and averaging the results usually yields a better estimate of the actual tensor [Moa06]. Averaging covariance matrices, which are also SDP, arises as a subtask in techniques proposed to solve problems such as video-tracking and radar detection (see e.g [NB13], [LLS10] and [PTM06]); fast averaging methods are particularly useful for these two real-time applications.

*This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office.

†ICTEAM Institute, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium

Geometric means are preferable to the arithmetic mean for averaging SPD matrices. For example, it has been shown in [PFA06] that the determinant of the arithmetic mean of two matrices can be bigger than the determinants of the two matrices themselves, which is usually not desirable. Indeed, let $A = [1\ 0; 0\ 5]$ and $B = [5\ 0; 0\ 1]$, then $\det(A) = \det(B) = 5$ while $\det(A/2 + B/2) = 9$. When the SPD matrices are represented as ellipsoids (the volume of an ellipsoid corresponding to the determinant of the matrix), this behavior can translate in a swelling effect.

We use the term *geometric mean* to refer to a function $\mathbf{G} : \mathbb{P}_n^N \to \mathbb{P}_n$ that satisfies all 10 properties of the well-known ALM list (see [ALM04] or Appendix A). When $N = 2$, $\mathbf{G}(A, B)$ is

$$A\#B = A^{\frac{1}{2}}(A^{-\frac{1}{2}}B^{\frac{1}{2}}A^{-\frac{1}{2}})^{\frac{1}{2}}A^{\frac{1}{2}}, \tag{1}$$

but several geometric means are known for $N \geq 3$.

First generalizations of the geometric mean of two matrices to many matrices consisted in building a geometric mean recursively, starting from expression (1). This approach led among others to the ALM and NBMP means, see [ALM04] and [BMP10]. These means were shown to meet all the properties of the ALM list. However, their recursive character makes them particulary expensive to compute when the number of matrices becomes large. To solve this problem, other matrix means were proposed, such as the CHEAP mean [BI11], the Circular mean [Pál11] and the Arithmetic-Harmonic mean [JV13]. These new means have the main advantage of being cheaper to evaluate that the ALM and NBMP means, but no longer satisfy the ten properties of the ALM list. They can thus be seen as approximate geometric means.

Another well-known mean on $\mathbb{P}_n$ is the Riemannian center of mass, often termed Karcher mean in view of the seminal work [Kar77], defined by

$$\mathrm{K}(A_1, \ldots, A_N) = \arg\min_{X \in \mathbb{P}_n} \sum_{i=1}^{N} \delta(X, A_i) \tag{2}$$

where $\delta(A, B)$ is the affine-invariant distance: $\delta(A, B) = ||\log(A^{-\frac{1}{2}}BA^{-\frac{1}{2}})||_F$. The Karcher mean (2) has recently been shown to meet the ten properties of the ALM list [BK12,LL11]. The interpretation of the Karcher mean as a center of mass, similarly to the arithmetic mean in Euclidian geometry, and the fact that its computation cost increases more slowly with the number of matrices than the ALM and NBMP means mentionned above, explain that this mean is currently the most popular geometric mean. The optimization problem appearing in (2) is convex, and can be addressed using, e.g., steepest descent methods. Different methods have been considered to solve (2); see, e.g., [JVV12] and [Zha13]. They produce sequences of iterates on $\mathbb{P}_n$ that converge (possibly under some conditions) to the Karcher mean.

In this paper, we propose approximate geometric means that fit in the following scheme. Given data matrices $A_1, \ldots, A_N$ in $\mathbb{P}_n$:

1. Consider a list $p = (p_1, \ldots, p_k)$ of permutations of $1, \ldots, N$.
2. Compute $B_i = \mathrm{PM}(A_1, \ldots, A_N, p_i)$, $i = 1, \ldots, k$, where PM is the inductive mean defined in Section 2.
3. Return $M(B_1, \ldots, B_k)$ where M is some matrix mean.

The various algorithms are named according to the pattern <p>-PM-<M>, where <p> indicates the permutation generation mechanism and <M> refers to the choice of M. We

show that, for various simple choices of p and M, most of the ALM properties are preserved. We compare <p>-PM-<M> algorithms with state-of-the-art methods according to two criteria: distance to the Karcher mean and computation time. It turns out that some of the new algorithms are nondominated, i.e., none of the other algorithms achieve both higher precision and lower computation time. We also observe that the advantage of the new algorithms becomes stronger when the size or the condition number of the data matrices gets large. It is also interesting to note that the <p>-PM-PM methods solely rely on the two-variable weighted geometric mean, a favorable situation towards an extension to sets other than $\mathbb{P}_n$.

The paper is organized as follows. In Section 3, we investigate algorithms of type F-PM-<M>, where the list p of permutations consists of the N! permutations of $1, \ldots, N$. These algorithms are practical only when N is very small. In Section 4, the list of permutations is restricted to a small subset generated by a shuffle method. Numerical experiments are presented in Section 5 and conclusions are drawn in Section 6.

2 The Progressive Merging algorithm

All the approximate geometric means introduced in this paper rely on the Progressive Merging method described in Algorithm 1. This method is also known in the literature as the inductive mean. It gives a basic process to generalize the geometric mean of two matrices to an approximate geometric mean of several matrices. This process computes successively the weighted geometric mean of two matrices, defined for $A, B \in \mathbb{S}_+^n$ and $t \in [0,1]$ by

$$A\#_t B = A^{\frac{1}{2}}(A^{-\frac{1}{2}}BA^{-\frac{1}{2}})^t A^{\frac{1}{2}}. \tag{3}$$

Algorithm 1 PM

Data: $A_1, ..., A_N \in \mathbb{S}_+^n$ and a permutation p of the sequence $1, 2, \ldots, N$.
Let $X_1 = A_1$;
for $i = 2, ..., N$
    $X_i = X_{i-1}\#_{1/i}A_{p(i)}$, defined as in (3)
end
return $X_N =: \mathrm{PM}(A_1, \ldots, A_N, p)$

The PM algorithm consists in taking successively one step towards each input matrix, with a progressively decreasing step-length. An illustration of this process is given in Figure 1. A variant of the sequence of matrices generated in Algorithm 1 has already been used in the literature to prove the monotonicity of the Karcher mean [LL11].

The PM algorithm returns a matrix containing information from all the initial matrices. Moreover, it lacks only one of the ten ALM properties:

Theorem 2.1. *PM($A_1, \ldots, A_N, p$) fulfills nine of the ten criteria of the ALM list (see Appendix A), the violated criterion being the invariance under permutation.*

Proof. The weighted mean of two matrices, defined by (3), fulfills the weighted version of the ten ALM properties [ALM04]. Applying these properties by recurrence to the sequence of matrices $(X_j)_{j=1,\ldots,N}$ generated by Algorithm 1, we obtain that the ALM properties are satisfied by $\mathrm{PM}(A_1, \ldots, A_N, p)$ except for P3 (invariance under permutation). We give here the detail of the proof for P7 (joint concavity), which is a bit less direct.

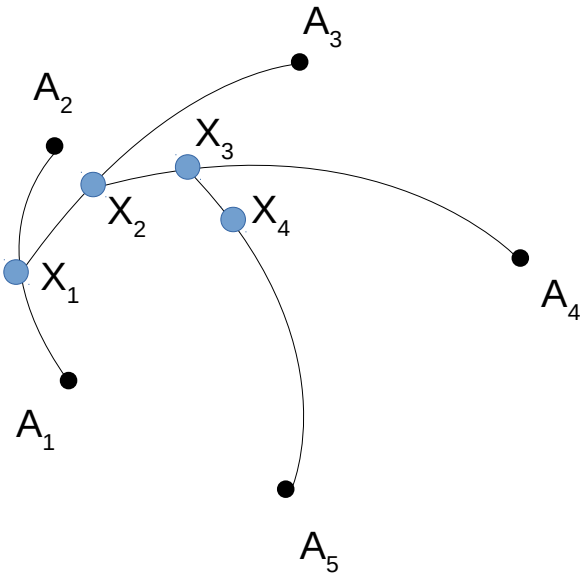

Figure 1: Illustration of the PM algorithm

Let $\lambda \in [0,1]$, let $A_j = \lambda \tilde{A}_j + (1-\lambda)\tilde{\tilde{A}}_j,\ j = 1,\dots,N$ and let p be a permutation of $1,\dots,N$. Let also $(X_j)_{j=1,\dots,N}$, $(\tilde{X}_j)_{j=1,\dots,N}$ and $(\tilde{\tilde{X}}_j)_{j=1,\dots,N}$ be the sequences of matrices generated by respectively $\mathrm{PM}(A_1,\dots,A_N,p)$, $\mathrm{PM}(\tilde{A}_1,\dots,\tilde{A}_N,p)$, and $\mathrm{PM}(\tilde{\tilde{A}}_1,\dots,\tilde{\tilde{A}}_N,p)$. We want to show that P7 (Appendix A) is satisfied for $\mathbf{G}(A_1,\dots,A_N) = \mathrm{PM}(A_1,\dots,A_N,p)$. The proof is by recurrence on the position j in the sequences of matrices $(X_j)_{j=1,\dots,N}$, $(\tilde{X}_j)_{j=1,\dots,N}$ and $(\tilde{\tilde{X}}_j)_{j=1,\dots,N}$. We first observe that P7 is trivially satisfied for $X_1, \tilde{X}_1$ and $\tilde{\tilde{X}}_1$:

$$X_1 = \lambda\tilde{A}_1 + (1-\lambda)\tilde{\tilde{A}}_1 = \lambda\tilde{X}_1 + (1-\lambda)\tilde{\tilde{X}}_1$$

We now suppose that P7 is satisfied for $X_j, \tilde{X}_j$ and $\tilde{\tilde{X}}_j$, and we show that it is then also satisfied for $X_{j+1}, \tilde{X}_{j+1}$ and $\tilde{\tilde{X}}_{j+1}$. Using the recurrence assumption and the monotonicity property (which is also part of the ALM list and can be proved in a similar way) we get:

$$X_{j+1} = X_j \#_{1/(j+1)} A_{j+1} \geq (\lambda\tilde{X}_j + (1-\lambda)\tilde{\tilde{X}}_j)\#_{1/(j+1)}A_{j+1}.$$

We conclude by using the joint concavity property for the weighted geometric mean of two matrices:

$$X_{j+1} \geq \lambda\tilde{X}_j\#_{1/(j+1)}\tilde{A}_{j+1} + (1-\lambda)\tilde{\tilde{X}}_j\#_{1/(j+1)}\tilde{\tilde{A}}_{j+1} = \lambda\tilde{X}_{j+1} + (1-\lambda)\tilde{\tilde{X}}_{j+1}.$$

□

Because of the (usual) non commutativity of the matrices $A_1, ..., A_N$, the result of Algorithm 1 depends on the order p in which the matrices are merged, hence the PM algorithm does not satisfy the property of invariance under permutation.

Numerical simulations reported on Figure 2 indicate that, on average, $\mathrm{PM}(A_1,\dots,A_N,p)$ is further than $\mathrm{K}(A_1,\dots,A_N)$ from the first matrices $(A_{p(1)}, A_{p(2)}, ...)$ and closer to the last ones $(A_{p(N)}, A_{p(N-1)}, ...)$. This finding will motivate Algorithm 3.

Algorithm 1 has the advantage of being really cheap to run: its computation time is driven by the $N-1$ computations of the two-variable weighted geometric means. Because the result of the PM algorithm satisfies nine of the ten properties of the ALM list, we see $\mathrm{PM}(A_1, A_2, \dots, A_N, p)$ as approximate geometric mean.

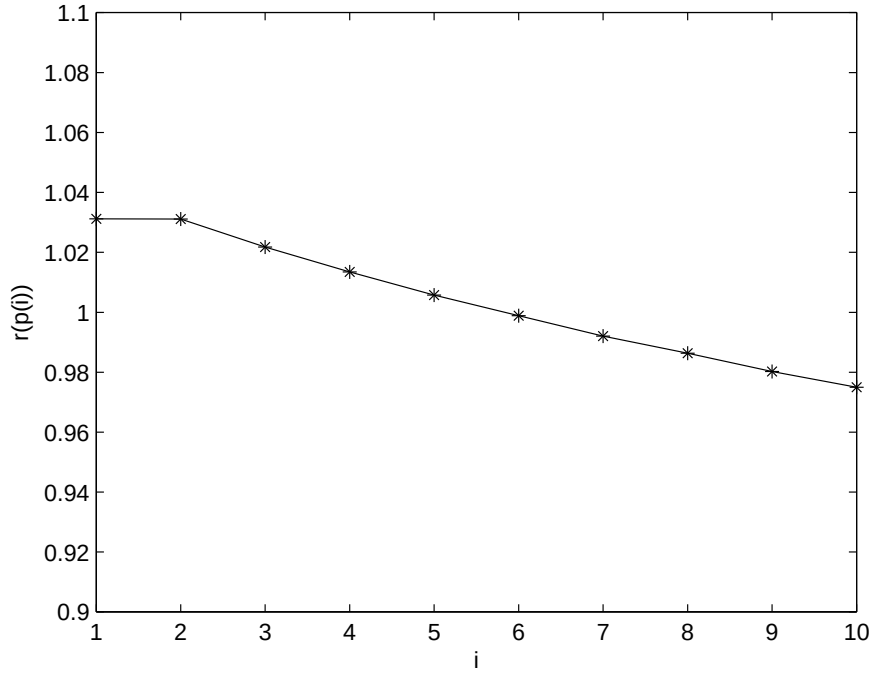

Figure 2: Evolution of the average ratio $r(p(i)) = \dfrac{\delta(\mathrm{PM}(A_1, \ldots, A_N, p), A_{p(i)})}{\delta(\mathrm{K}(A_1, \ldots, A_N), A_{p(i)})}$ as a function of i. The average is over 1000 sets of $N = 10$ matrices of size $n = 10$. The input matrices were generated according to a Wishart distribution $W_n(I, n)$. The figure is zoomed around the value $r(p(i)) = 1$.

3 The Full - Progressive Merging (F-PM) algorithm

In the proposed F-PM-<M> class of algorithms described in Algorithm 2, the PM algorithm is run on all possible permutations of $1, \ldots, N$, resulting in $N!$ estimates $B_1, \ldots, B_{N!}$ of geometric means. An approximate geometric mean of $B_1, \ldots, B_{N!}$ is then returned. We expect these estimates $B_1, \ldots, B_{N!}$ to be located closer to each other than the initial matrices, allowing approximate geometric means to be used to combine the estimates without introducing large errors (see lines 2 to 14 in Algorithm 2). The choice of the algorithm <M> to evaluate the approximate geometric mean of $B_1, \ldots, B_{N!}$ leads to different variants of F-PM-<M>.

3.1 The F-PM-PM algorithm

In this first variant, the averaging of $B_1, \ldots, B_{N!}$ is performed using the PM algorithm itself. We show the following theorem:

Theorem 3.1. *The F-PM-PM algorithm returns a matrix satisfying the same ALM properties as the PM algorithm (Algorithm 1), namely, the ten properties except invariance under permutation.*

Proof. The proof is similar to the one proposed for Theorem 2.1. The F-PM-PM algorithm is not invariant under permutation because the order in which the estimates B_i are computed and then merged in Algorithm 2 has an impact on the solution, as explained in Section 2. □

Algorithm 2 F-PM-<M> with <M> = PM, Cr or Ar

Data: $A_1, ..., A_N \in \mathbb{P}^n$

1: Generate a list $p = (p_1, \dots, p_{N!})$ of the permutations of $1, \dots, N$; $k = N!$
2: $M = 0$, if <M> = Cr: $M_a = 0 = M_h$
3: **for** $i = 1, ..., k$;
4: $B_i = \mathbf{PM}(A_1, ..., A_N, p_i)$;
5: **if** <M> = PM
6: $M \leftarrow M \#_{\frac{i-1}{i}} B_i$;
7: **else if** <M> = Cr
8: $M_a \leftarrow \frac{i-1}{i} M_a + \frac{1}{i} B_i$;
9: $M_h \leftarrow \frac{i-1}{i} M_h + \frac{1}{i} B_i^{-1}$;
10: $M = M_a \# M_h^{-1}$;
11: **else if** <M> = Ar
12: $M \leftarrow \frac{i-1}{i} M + \frac{1}{i} B_i$;
13: **end**
14: **end**
15: **return** M;

3.2 The F-PM-Cr algorithm

In this second variant, we use the Crude algorithm [JV13], approximating a geometric mean by combining arithmetic and harmonic means, to average the estimates $B_1, \dots, B_{N!}$. The Crude mean of matrices $B_1, \dots, B_{N!}$ is defined as

$$G_{Cr}(B_1, \dots, B_{N!}) = \left(\frac{\sum_{i=1}^{N!} B_i}{N!} \right) \#_{1/2} \left(\frac{\sum_{i=1}^{N!} B_i^{-1}}{N!} \right)^{-1}$$

Theorem 3.2. *The F-PM-Cr algorithm returns a matrix satisfying nine of the ten ALM properties, the violated property being the determinant equality.*

Proof. We showed in Section 2 that the PM algorithm satisfies all the ALM properties except the invariance under permutation. We show that most of these properties are conserved when using the Crude mean, and that the invariance under permutation is recovered as soon as the PM algorithm is run on the $N!$ permutations of $1, \dots, N$. We give here the detail of the proof for properties P2, P3, P4 and P7.

P2 (Joint homogenity): Because the PM algorithm satisfies the joint homogeneity property, we know that for each permutation p, the following equality holds:

$$\mathrm{PM}(\alpha_1 A_1, \dots, \alpha_N A_N, p) = (\alpha_1 \dots \alpha_N)^{\frac{1}{N}} \mathrm{PM}(A_1, \dots, A_N, p).$$

Observing then that the Crude means fulfills the following equality ends the proof for P2:

$$\begin{aligned} &G_{Cr}\left((\alpha_1 \dots \alpha_N)^{\frac{1}{N}} \mathrm{PM}(A_1, \dots, A_N, p_1), \dots, (\alpha_1 \dots \alpha_N)^{\frac{1}{N}} \mathrm{PM}(A_1, \dots, A_N, p_{N!}) \right) \\ &= (\alpha_1 \dots \alpha_N)^{\frac{1}{N}} G_{Cr}\left(\mathrm{PM}(A_1, \dots, A_N, p_1), \dots, \mathrm{PM}(A_1, \dots, A_N, p_{N!}) \right). \end{aligned}$$

P3 (Invariance under permutation): The estimates B_i, i = 1 ..., N!, are obtained by running the PM algorithm on all possible permutations. Therefore, the set of matrices B_i appearing in line 4 of Algorithm 2 will be the same regardless the initial order of the matrices. We can conclude by observing that the Crude mean is itself invariant under permutation.

P4 (Monotonicity): Let $A_i \leq \tilde{A}_i \ \forall i = 1, \ldots, N$, let $B_j = \mathrm{PM}(A_1, \ldots, A_N, p_j),\ j = 1, \ldots, N!$ and finally let $\tilde{B}_j = \mathrm{PM}(\tilde{A}_1, \ldots, \tilde{A}_N, p_j),\ j = 1, \ldots, N!$. Theorem 2.1 tells us:

$$B_j \leq \tilde{B}_j \qquad \forall j = 1, \ldots, N!.$$

Using the monotonicity property for the arithmetic, harmonic and two-variable geometric means [Cha12], we finally obtain

$$G_{Cr}(B_1, \ldots, B_{N!}) \leq G_{Cr}(\tilde{B}_1, \ldots, \tilde{B}_{N!}).$$

P7 (Joint concavity) Let $\lambda \in [0, 1]$, let $A_i = \lambda \tilde{A}_i + (1 - \lambda)\tilde{\tilde{A}}_i,\ i = 1, \ldots, N$ and let $B_j = \mathrm{PM}(A_1, \ldots, A_N, p_j),\ j = 1, \ldots, N!,\ \tilde{B}_j = \mathrm{PM}(\tilde{A}_1, \ldots, \tilde{A}_N, p_j),\ j = 1, \ldots, N!$ and finally let $\tilde{\tilde{B}}_j = \mathrm{PM}(\tilde{\tilde{A}}_1, \ldots, \tilde{\tilde{A}}_N, p_j),\ j = 1, \ldots, N!$. Theorem 2.1 gives:

$$B_j \geq \lambda \tilde{B}_j + (1 - \lambda)\tilde{\tilde{B}}_j \qquad \forall j = 1, \ldots, N.$$

Summing the two sides of this inequality, we obtain:

$$\frac{1}{N!}\sum_{j=1}^{N!} B_j \geq \frac{\lambda}{N!}\sum_{j=1}^{N!} \tilde{B}_j + \frac{(1-\lambda)}{N!}\sum_{j=1}^{N!} \tilde{\tilde{B}}_j$$

and because of the monotonicity and joint concavity properties for the harmonic mean [Cha12], we get successively:

$$\begin{aligned}\left(\frac{1}{N!}\sum_{j=1}^{N!} B_j^{-1}\right)^{-1} &\geq \left[\frac{1}{N!}\sum_{j=1}^{N!}\left(\lambda\tilde{B}_j + (1-\lambda)\tilde{\tilde{B}}_j\right)^{-1}\right]^{-1} \\ &\geq \lambda\left(\frac{1}{N!}\sum_{j=1}^{N!}\tilde{B}_j^{-1}\right)^{-1} + (1-\lambda)\left(\frac{1}{N!}\sum_{j=1}^{N!}\tilde{\tilde{B}}_j^{-1}\right)^{-1}\end{aligned}$$

We finally conclude by using the monotonicity and joint concavity properties for the two-variable geometric mean:

$$\begin{aligned}G_{Cr}(B_1, \ldots, B_{N!}) &= \left(\frac{1}{N!}\sum_{j=1}^{N!} B_j\right) \#_{1/2} \left(\frac{1}{N!}\sum_{j=1}^{N!} B_j^{-1}\right)^{-1} \\ &\geq \lambda G_{Cr}(\tilde{B}_1, \ldots, \tilde{B}_{N!}) + (1-\lambda) G_{Cr}(\tilde{\tilde{B}}_1, \ldots, \tilde{\tilde{B}}_{N!})\end{aligned}$$

□

3.3 The F-PM-Ar algorithm

Finally, we consider a third variant: the F-PM-Ar algorithm, in which the average of the estimates is defined as their arithmetic mean. We show the following result:

Theorem 3.3. *The F-PM-Ar algorithm returns a matrix satisfying eight of the ten ALM properties, the violated properties being the invariance under inversion and the determinant equality.*

Proof. The proof is similar to the one for the F-PM-Cr algorithm. □

4 The In Shuffle - Progressive Merging (IS-PM) algorithm

The F-PM-<M> algorithms require computing $N!$ matrices $B_1, \ldots, B_{N!}$, which is impractical unless N is very small. In this section, we propose a class of methods, termed IS-PM-<M> (Algorithm 3), where the PM algorithm is run on a few well-chosen permutations.

Algorithm 3 IS-PM-<M>, with <M> = PM, Cr or Ar

1: Replace line 1 of Algorithm 2 by the following:
2: $\bar{k} = \lceil \log_2(N) \rceil - 1$
3: **for** $i = 2 : \bar{k}$
4: $p_{2i-1} = \mathbf{in-shuffle}(p_{2i-3})$; see (5)
5: $p_{2i} = \mathbf{reverse}(p_{2i-1})$; see (4)
6: **end**
7: Continue as in Algorithm 2, from line 2.

The IS-PM-<M> algorithms have the same ALM properties as their F-PM-<M> counterpart, except for the invariance by permutation which is lost. In view of Theorem 3.1, the next results follow:

Theorem 4.1. *The IS-PM-PM algorithm returns a matrix satisfying the same ALM properties as the PM algorithm (Algorithm 1), namely, the ten properties except invariance under permutation.*

Theorem 4.2. *The IS-PM-Cr algorithm returns a matrix satisfying eight of the ten ALM properties, the violated properties being the invariance under permutation and the determinant equality.*

Theorem 4.3. *The IS-PM-PM algorithm returns a matrix satisfying seven of the ten ALM properties, the violated properties being the invariance under permutation, under inversion and the determinant equality.*

4.1 Reverse method

Let $p = [p(1), p(2), \ldots, p(N)]$ be a permutation of $1,\ 2, \ldots, N$. The **reverse** procedure invoked in line 5 of Algorithm 3 consists in reversing the sequence taken as input:

$$\mathbf{reverse}(p) = [p(N), p(N-1), ..., p(1)] \tag{4}$$

For example, $\mathbf{reverse}(1, 2, 3)$ gives the sequence $3, 2, 1$. The motivation behind line 1 of Algorithm 3 is to address the bias documented in Figure 2.

4.2 In Shuffle method

The *in shuffle* (IS) is a specific type of *riffle shuffle*, a shuffling method often used for card shuffling. This method reorders a given sequence containing $N = 2\tilde{N}$ elements as:

$$\mathbf{in-shuffle}(p) = [p(\tilde{N}+1), p(1), p(\tilde{N}+2), p(2), \ldots, p(2\tilde{N}), p(\tilde{N})] \tag{5}$$

In card game terms, the in-shuffle consists in firstly separating the cards in two decks of equal size: the first half of the cards are placed in the first deck and the second half in

the other one. Then, the cards of the two decks are perfectly interleaved, meaning that the resulting deck contains a perfect alternance of cards coming from each deck. In the in shuffle method, the first card becomes the second one after the shuffling.

When the number N of elements is odd, $N = 2\tilde{N} + 1$, we alternate between $\tilde{N}$ and $\tilde{N} + 1$ elements in the first deck.

The choice of $\bar{k}$ in line 2 of Algorithm 3 ensures that all the generated permutations are distinct. This follows from [DGK83].

5 Numerical results

We compare here the performances achieved by our algorithms to those obtained with two other ways of approximating the Karcher mean, namely, the Cheap mean and a prematurely stopped iterative optimization algorithm.

The Cheap mean [BI11] is an approximate geometric mean computed according to an iterative scheme. The initial iterates are the N input matrices: $\tilde{A}_i^0 = A_i, i = 1, \ldots, N$, and subsequent iterates are defined by:

$$\tilde{A}_i^{k+1} = \tilde{A}_i^k \exp\left(\frac{1}{N} \sum_{l=1, l\neq i}^{N} \log((\tilde{A}_i^k)^{-1} \tilde{A}_l^k)\right) \tag{6}$$

In the numerical experiments described here, we stopped the algorithm when $\max_{i,j} d(\tilde{A}_i^k, \tilde{A}_j^k) \leq 10^{-8}$ with $d(A, B) = ||A - B||_F$. The Cheap mean is then defined as the arithmetic mean of the iterates $A_1^k, \ldots, A_N^k$.

A steepest descent algorithm with automatic step-length choice (SD-Auto) has been proposed in [BI13] to solve (2). We resort to the state-of-the-art implementation available as the `karcher.m` function in the Matrix Means Toolbox (`http://bezout.dm.unipi.it/software/mmtoolbox/`). Running only a few iterations of this algorithm gives us another estimate of the Karcher mean.

We compare here the accuracy of the estimates and the computation time required by the methods with some key parameters of the problem. We measure a normalized approximation error, defined as:

$$E_{rel} = \frac{\delta(\tilde{G}(A_1, \ldots, A_N), K(A_1, \ldots, A_N))}{\delta(A_1, K(A_1, \ldots, A_N))}$$

with $\delta(A, B)$ the affine-invariant distance, $\tilde{G}(A_1, \ldots, A_N)$ the estimate of the Karcher mean and $K(A_1, \ldots, A_N)$ a high precision value for the Karcher mean, computed with the SD-Auto algorithm mentioned above taking as initial guess the arithmetic mean of the matrices and stopped after a large number of iterations. Each column in Figure 3 gives the evolution of the performances with respect to a specific parameter of the problem generator.

The first column of Figure 3 illustrates the performances as a function of the number of matrices N. The test was run on 100 sets of N matrices of size $n = 20$ and distributed according to the Wishart distribution $W_n(n, I)$. The three algorithms IS-PM-PM, IS-PM-Cr and IS-PM-Ar are compared to the Cheap algorithm (iteration (6)) and the SD-Auto algorithm stopped after 10 or 15 iterations (taking as initial guess the arithmetic mean of the matrices $A_1, \ldots, A_N$). The Cheap mean is globally a little more costly than the other methods considered.

The second column gives the performances achived by the different methods, run on sets of $N = 10$ matrices of varying size n, again distributed according to the $W_n(n, I)$

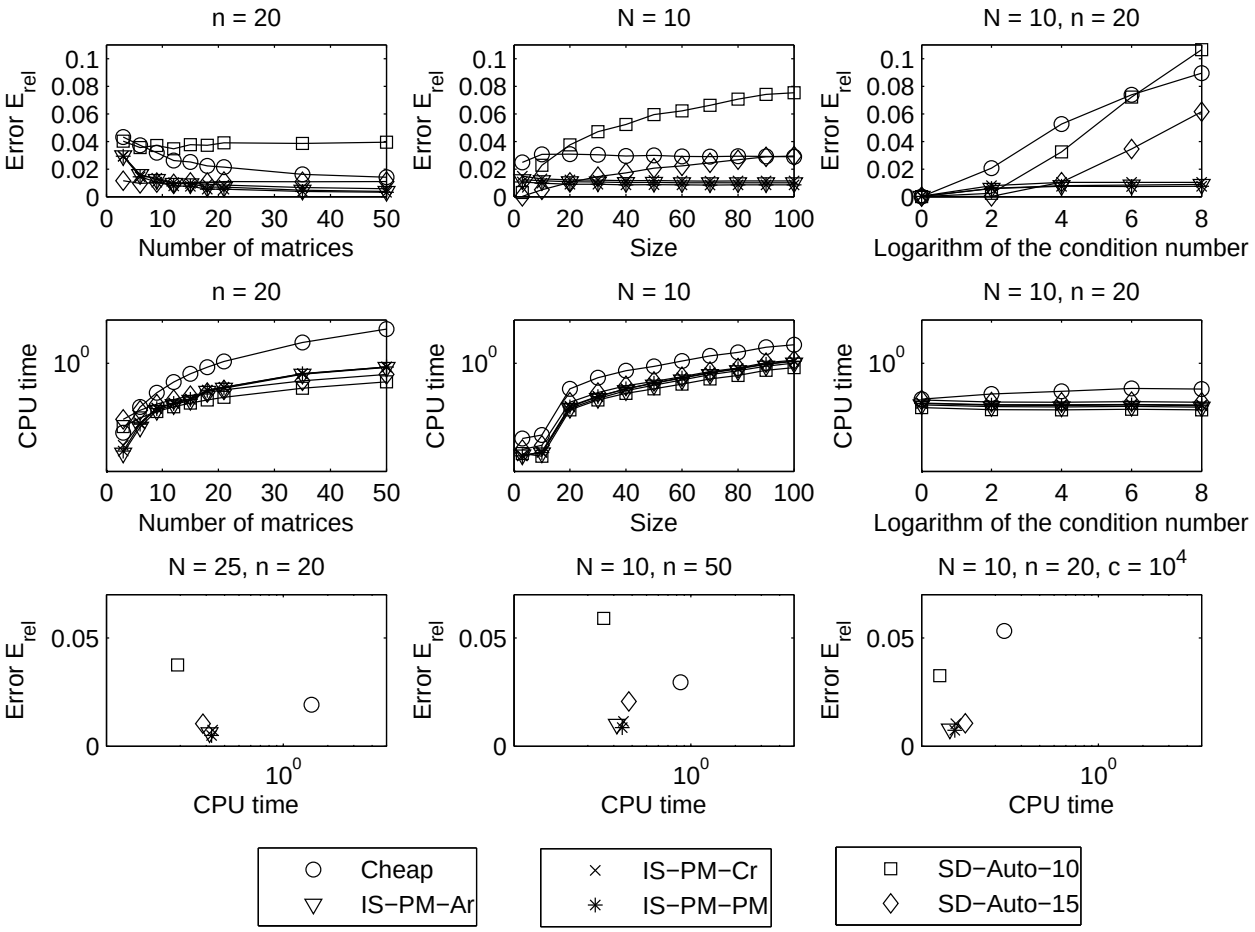

Figure 3: Evolution of the (normalized) approximation error and of the computation cost of the methods with several key parameters of the problem

distribution. As indicated by the first graph of this columns, the accuracy of our methods does not seem to be sensitive to the size of the matrices, unlike the prematurely stopped SD-Auto methods.

Finally, the third column illustrates the performances achieved by the methods on sets of $N = 10$ matrices of size $n = 20$ and with a varying condition number. To impose the condition number of our matrices, we used the following lines of code:

```
[Q,~] = qr(rand(n));
D = diag([rand(1,n-1)+1,10^(-n)]);
A = Q*D*Q';
```

Figure 3 indicates that our methods are not sensitive to the condition number of the input matrices, unlike the Cheap and prematurely stopped SD-Auto methods.

6 Conclusion

We have proposed and analyzed a class of approximate geometric means on the space of SPD matrices. All the methods of the class rely on the inductive mean (PM) combined upstream with a permutation generator <P> and downstream with a matrix average <M> that aggregates (preferably on the fly, as in the proposed algorithms) the outcomes of the

inductive mean. We have proposed a <P> inspired from card-shuffling techniques, for which numerical experiments indicate that the proposed method compares favorably with the state of the art according to the accuracy-vs-time criterion (where "accuracy" refers to the distance to the Karcher mean, i.e., the Riemannian center of mass with respect to the trace metric). Further work may aim at obtaining even better performances with other choice of <P> and <M>.

Appendices

A ALM list of criteria for geometric means

In [ALM04], Ando, Li and Mathias have collected a list of criteria that a mean has to satisfy to be considered as a geometric mean. These criteria are the followings (where $\mathbf{G}(A_1, ..., A_n)$ stands for a candidate geometric mean):

P1. Consistency: if the matrices $A_1, ..., A_n$ commute, then $\mathbf{G}(A_1, ..., A_n) = (A_1 \cdots A_n)^{\frac{1}{n}}$

P2. Joint homogeneity: $\mathbf{G}(\alpha_1 A_1, ..., \alpha_n A_n) = (\alpha_1 \cdots \alpha_n)^{\frac{1}{n}} \mathbf{G}(A_1, ..., A_n),\ \forall \alpha_1, ..., \alpha_n \in \mathbb{R}+$

P3. Invariance under permutation: $\mathbf{G}(A_{\pi_1}, ..., A_{\pi_n}) = \mathbf{G}(A_1, ..., A_n)$ with π a permutation of $(1, ..., n)$.

P4. Monotonicity: if $B_i \leq A_i \forall\ i = 1, ..., n$ then $\mathbf{G}(B_1, ..., B_n) \leq \mathbf{G}(A_1, ..., A_n)$

P5. Continuity from above: if $A_i^{(j)}$ denotes a monotonically decreasing sequence that converges towards A_i^* for $j \to \infty$, $\forall i = 1, ..., n$, then $\mathbf{G}(A_1^{(j)}, ..., A_n^{(j)})$ converges towards $\mathbf{G}(A_1^*, ..., A_n^*)$ as $j \to \infty$.

P6. Congruence invariance: $\forall S \in \mathbb{R}^{m \times m}$ invertible, $\mathbf{G}(SA_1S^T, ..., SA_nS^T) = S\mathbf{G}(A_1, ..., A_n)S^T$

P7. Joint concavity: $\mathbf{G}(\lambda A_1 + (1-\lambda)B_1, ..., \lambda A_n + (1-\lambda)B_n) \geq \lambda \mathbf{G}(A_1, ..., A_n) + (1-\lambda)\mathbf{G}(B_1, ..., B_n)$ for $0 \leq \lambda \leq 1$.

P8. Invariance under inversion: $\mathbf{G}(A_1, ..., A_n) = (\mathbf{G}(A_1^{-1}, ..., A_n^{-1}))^{-1}$

P9. Determinant equality: $\det \mathbf{G}(A_1, ..., A_n) = (\det A_1 \cdots \det A_n)^{\frac{1}{n}}$

P10. Arithmetic-geometric-harmonic inequality: $\frac{1}{n} \sum_{i=1}^n A_i \geq \mathbf{G}(A_1, ..., A_n) \geq (\frac{1}{n} \sum_{i=1}^n A_i^{-1})^{-1}$

References

[ALM04] T Ando, Chi-Kwong Li, and Roy Mathias. Geometric means. *Linear algebra and its applications*, 385:305–334, 2004.

[BI11] Dario Andrea Bini and Bruno Iannazzo. A note on computing matrix geometric means. *Advances in Computational Mathematics*, 35(2-4):175–192, 2011.

[BI13] 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.

[BK12] Rajendra Bhatia and Rajeeva L Karandikar. Monotonicity of the matrix geometric mean. *Mathematische Annalen*, 353(4):1453–1467, 2012.

[BMP10] Dario Bini, Beatrice Meini, and Federico Poloni. An effective matrix geometric mean satisfying the ando-li-mathias properties. *Mathematics of Computation*, 79(269):437–452, 2010.

[Cha12] Pattrawut Chansangiam. Operator means and applications. *LINEAR ALGEBRA–THEOREMS AND APPLICATIONS*, page 163, 2012.

[CSV12] 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.

[DGK83] Persi Diaconis, RL Graham, and William M Kantor. The mathematics of perfect shuffles. *Advances in Applied Mathematics*, 4(2):175–196, 1983.

[JV13] Ben Jeuris and Raf Vandebril. Geometric mean algorithms based on harmonic and arithmetic iterations. In *Geometric Science of Information*, pages 785–793. Springer, 2013.

[JVV12] 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*, 39:379–402, 2012.

[Kar77] Hermann Karcher. Riemannian center of mass and mollifier smoothing. *Communications on pure and applied mathematics*, 30(5):509–541, 1977.

[LL11] Jimmie Lawson and Yongdo Lim. Monotonic properties of the least squares mean. *Mathematische Annalen*, 351(2):267–279, 2011.

[LLS10] Yunpeng Liu, Guangwei Li, and Zelin Shi. Covariance tracking via geometric particle filtering. *EURASIP Journal on Advances in Signal Processing*, 2010(1):583918, 2010. URL: `http://asp.eurasipjournals.com/content/2010/1/583918`, `doi:10.1155/2010/583918`.

[Moa06] Maher Moakher. On the averaging of symmetric positive-definite tensors. *Journal of Elasticity*, 82(3):273–296, 2006.

[NB13] Frank Nielsen and Rajendra Bhatia. *Matrix information geometry*. Springer, 2013.

[Pál11] Miklós Pálfia. A multivariable extension of two-variable matrix means. *SIAM Journal on Matrix Analysis and Applications*, 32(2):385–393, 2011.

[PFA06] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[PTM06] Fatih Porikli, Oncel Tuzel, and Peter Meer. Covariance tracking using model update based on lie algebra. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 728–735. IEEE, 2006.

[Zha13] Teng Zhang. A majorization-minimization algorithm for the karcher mean of positive definite matrices, 2013. `arXiv:1312.4654`.