Matrix geometric means based on shuffled inductive sequences *

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

October 19, 2016

Abstract

We propose a new deterministic sequence converging to the least squares mean of N symmetric positive definite (SPD) matrices. By "least squares mean", we refer to the Riemannian barycenter with respect to the natural metric (also known as the trace metric or affine-invariant metric) on the set $\mathbb{P}_n$ of $n \times n$ SPD matrices. In some papers, this mean is also referred to as the Karcher mean. The sequence we propose belongs to the family of Inductive sequences, obtained by letting a point take successive steps towards the data points, with step lengths progressively diminishing to zero. We use the word "Inductive" since each point of those sequences can be seen as the Inductive mean of a well-chosen set of points containing multiple replications of the data points. We show that visiting the data points in a cyclic order usually results in a slow convergence, and remedy this weakness by reordering repeatedly the data points using a shuffling algorithm. We prove the convergence of the resulting Inductive sequence to the least squares mean of the data in the general framework of NPC spaces (complete metric spaces with non-positive curvature), which includes the set $\mathbb{P}_n$. We also illustrate numerically on $\mathbb{P}_n$ that some points of this sequence are fairly accurate estimates of the least squares mean, while being considerably cheaper to compute, and perform well as initializers for state-of-the-art algorithms. To reduce further the computation time, we finally exploit Inductive sequences to produce a family of parallelizable algorithms for estimating the least squares mean.

Keywords: geometric means, Karcher mean, least squares mean, inductive mean, symmetric positive definite matrices, NPC spaces.

AMS subject classifications: 15A45, 47A64, 65F30

1 Introduction

We consider the problem of efficiently averaging a set of N symmetric positive definite (SPD) matrices. This task arises in many applications, e.g., in medical imaging (DTI

*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

images denoising and segmentation [CSV12]), in mechanics (elasticity tensor computation [Moa06]) and in some recently developed algorithms for video tracking and radar detection [LLS10, NB13]. The development of fast averaging methods is crucial to allow algorithms to run in real time in this last application.

A simple way to average SPD matrices is to take their arithmetic mean. However, this approach has some important shortcomings. First, some applications require the mean M to be invariant under inversion, meaning that $M(A,B) = M(A^{-1},B^{-1})^{-1}$ for any $A,B \in \mathbb{P}_n$ (with $\mathbb{P}_n$ the cone of $n \times n$ SPD matrices), which is not the case for the arithmetic mean. Secondly, the determinant of the arithmetic mean of two matrices can be considerably bigger than the determinants of the two matrices themselves, which does not make sense in some applications [PFA06].

A matrix *geometric mean* is a function $G : \mathbb{P}_n^N \to \mathbb{P}_n$ that satisfies all 10 properties of the celebrated ALM list (see [ALM04] or Appendix A). This uniquely defines the two-variable geometric mean $G(A,B)$ as

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

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

First generalizations of the two-variable geometric mean to more matrices were built recursively, starting from expression (1). These means, which include among others the ALM and BMP means [ALM04, BMP10], were shown to meet the ten ALM properties. However, their recursive definition results in a prohibitive computation time when the number of matrices is large. To remedy this situation, other computationally cheaper matrix means were proposed, such as the Cheap mean [BI11], the Circular mean [Pál11] and the Arithmetic-Harmonic mean [JV13]. However, these new means do no longer satisfy the ten ALM properties: they can be considered as quasi-geometric means.

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

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

where $\delta(A,B)$ is the affine-invariant distance: $\delta(A,B) = ||\log(A^{-\frac{1}{2}}BA^{-\frac{1}{2}})||_F$ for $A,B \in \mathbb{P}_n$. The least squares mean has been shown to meet the ten properties of the ALM list [BK12, LL11]. This mean is currently the most popular geometric mean, probably due to its interpretation as a center of mass, similarly to the arithmetic mean in Euclidean geometry, and to the fact that its computation cost increases more slowly with the number of matrices than the ALM and BMP means mentioned above.

Various methods have been proposed to compute the least squares mean, most of them resorting to optimization techniques to solve problem (2), such as gradient descent, conjugate gradient, limited-memory BFGS, Newton and trust-region methods [RA11, JVV12, BI13, YHAG16]. A majorization-minimization algorithm has also been considered in [Zha13]. The sizable computation time of Hessian evaluations results in rather slow second-order algorithms, and first-order methods are usually more efficient [JVV12]. In [YHAG16], the condition number of the Hessian of the cost function (2) is showed to be bounded by a constant related to the logarithm of the largest condition number of the data points, which explains the good performances of first order methods. Several step length choices have been investigated for the gradient descent, the most popular being probably the one provided in the Matrix Means Toolbox developed by Bini & Iannazzo (`http://bezout.dm.`

unipi.it/software/mmtoolbox/). In this implementation, the step length is dynamically chosen depending on the condition number of the matrices arising in the expression of the gradient.

Another approach for computing the least squares mean is based on the Law of Large Numbers characterizing least squares means on an arbitrary NPC space $(\mathcal{M}, \delta)$ (i.e., a complete non-positively curved metric space) as limit points of random walks, see [Stu03]. In the case where $(\mathcal{M}, \delta)$ is the set $\mathbb{P}_n$ endowed with the affine-invariant metric, Holbrook [Hol12] proved that the least squares mean can be obtained as the limit point of a related *deterministic* sequence, obtained by letting one point take successive steps towards the N data points, with decreasing step sizes. The data points are then visited according to the cyclic order $1, 2, \ldots, N, 1, 2, \ldots$. This result was extended to general NPC spaces by Lim, Pálfia and Bačák (see [LP14] and [Bac14]).

There is a direct link between the sequence proposed by Holbrook and the Inductive mean proposed on $\mathbb{P}_n$ by Sagae & Tanabe [ST94] (see Definition 2.2 below). The Inductive mean is a quasi-geometric matrix mean in the sense that it satisfies all ALM properties except invariance under permutation: usually $M_{\text{Ind}}(A_{p_1}, A_{p_2}, \ldots, A_{p_N}) \neq M_{\text{Ind}}(A_1, \ldots, A_N)$ for $(p_1, \ldots, p_N)$ an arbitrary permutation of $(1, \ldots, N)$. The link is that the k^{th} element of the sequence proposed by Holbrook corresponds to

$$M_{\text{Ind}}(\underbrace{A_1, A_2, \ldots, A_N, A_1, A_2, \ldots}_{k \text{ elements}}).$$

Hence, we refer to the sequence proposed by Holbrook as an *Inductive sequence*. On $\mathbb{P}_n$, Inductive sequences correspond to the iterates of an incremental gradient descent for problem (2). On arbitrary NPC spaces, possibly lacking a differentiable structure, Inductive sequences might be the only available tool for least squares means computation.

In this paper, we show that the Inductive mean tends to overemphasize on $\mathbb{P}_n$ the last data points, and illustrate the resulting negative impact on the convergence rate of Inductive sequences. As a remedy, we propose to replace the cyclic ordering $1, 2, \ldots, N, 1, 2, \ldots$ by other orderings obtained via shuffling algorithms. We prove the convergence of the corresponding Inductive sequences (referred to here as shuffled Inductive sequence) towards the least squares mean of the data under weak assumptions on the shuffling process. Our proof is presented in the general framework of NPC spaces. We provide a shuffling strategy and illustrate numerically the gain in convergence speed. We also use some points of our shuffled sequence as quasi-geometric matrix means and compare them numerically with other means, regarding the ALM criteria, the proximity to the least squares mean and the computation time. We illustrate that these points are efficient initializers for steepest descent algorithms when solving problem (2); none of the other initializers considered in the literature yields indeed a faster convergence. Finally, we propose a set of parallelizable algorithms for estimating the least squares mean using Inductive means, and compare them with the sequential approach.

The paper is organized as follows. The Inductive mean and Inductive sequences are presented in Section 2. The third section contains the core of our contributions: we motivate and describe our shuffling strategy, and prove the convergence of shuffled Inductive sequences. In Section 4, we use our shuffled sequence to generate quasi-geometric matrix means and compare numerically those points with existing means. Finally, Section 5 contains a family of parallelizable algorithms for estimating the least squares mean, while conclusions are drawn in Section 6.

2 Background on Inductive means and sequences

Inductive sequences can be used to estimate the least squares mean of a set of data points belonging to an arbitrary NPC space $(\mathcal{M}, \delta)$. The least squares mean of the points $\mathbf{A} = (A_1, \dots, A_N) \in \mathcal{M}^N$, with corresponding positive weights $\omega = (\omega_1, \dots, \omega_N)$, is defined as

$$\Lambda(\omega; \mathbf{A}) = \arg\min_{X \in \mathcal{M}} \sum_{i=1}^{N} \omega_i \delta^2(X, A_i). \tag{3}$$

Inductive sequences are built by starting from one data point and making successive steps towards other data points, which can be more formally written as follows.

Definition 2.1. (Inductive sequences) *Let $\mathbf{A} = (A_1, \dots, A_N) \in \mathcal{M}^N$ be a tuple of data points belonging to a NPC space $(\mathcal{M}, \delta)$. Given a sequence of indexes $p := (p_1, p_2, \dots)$ and a sequence of step lengths $t := (t_1, t_2, \dots)$, we define an* Inductive sequence $(X_k)_{k \in \mathbb{N}_0}$ *(with $\mathbb{N}_0$ the set of strictly positive naturals) as follows:*

$$\begin{aligned} X_1 &= A_{p_1} \\ X_k &= X_{k-1} \#_{t_k} A_{p_k} \qquad k \geq 2. \end{aligned}$$

The notation $A \#_{t_k} B$, $t_k \in [0, 1]$, stands for the unique point located on the minimizing geodesic between A and B and satisfying $\delta(A, A \#_{t_k} B) = t_k \delta(A, B)$. If $\mathcal{M} = \mathbb{P}_n$, this point is the weighted two-variable geometric mean, defined as

$$A \#_t B = A^{\frac{1}{2}} (A^{-\frac{1}{2}} B A^{-\frac{1}{2}})^t A^{\frac{1}{2}}. \tag{4}$$

To the best of our knowledge, convergence of Inductive sequences to the weighted least squares mean has been proven for two specific choices of p and t:

Theorem 2.1. (Law of Large Numbers [Stu03]) *In Definition 2.1, choose $p := (p_1, p_2, \dots)$ as a sequence of independent identically distributed random variables taking their value in the set $\{1, \dots, N\}$ according to the distribution $\mu(x) = \sum_{i=1}^{N} \omega_i \delta_i^{\text{Dirac}}(x)$, where $\delta_z^{\text{Dirac}}(x)$ is the Dirac delta located at z, and choose $t_k = 1/k$ for all k. Then, the sequence $(X_k)_{k \in \mathbb{N}_0}$ converges to the least squares mean (3) almost surely.*

Theorem 2.2. (Cyclic Inductive sequence [LP14, Bac14]) *In Definition 2.1, choose $p_k = k \bmod N$, identifying the nul residual with N, i.e., $p_{mN} = N \ \forall m \in \mathbb{N}_0$, and choose $t_k = \omega_{p_k} / \left(\sum_{i=1}^{k} \omega_{p_i} \right)$ (in the case $\omega_k = 1/N$ for all k, this amounts to choosing $t_k = 1/k$). Then, the sequence $(X_k)_{k \in \mathbb{N}_0}$ converges to the least squares mean (3).*

The convergence of the Cyclic Inductive sequence (noted CI-sequence) is typically slow. We propose in the next section a strategy to improve the convergence rate. We end the current section with the definition of Inductive means on $\mathbb{P}_n$.

Definition 2.2. (Inductive mean) *The Inductive mean $M_{\text{Ind}}(A_1, \dots, A_N)$ of a tuple of data points $(A_1, \dots, A_N) \in \mathbb{P}_n^N$ with corresponding weights $\omega = (\omega_1, \dots, \omega_N)$ is defined as*

$$M_{\text{Ind}}(A_1, \dots, A_N) = X_N,$$

where X_N is generated according to Definition 2.1, for $\mathbf{A} = (A_1, \dots, A_N)$, choosing $p = (1, \dots, N)$ and $t_k = \omega_k / \left(\sum_{i=1}^{k} \omega_i \right)$ for $k = 1, \dots, N$.

The Inductive mean satisfies all the ALM criteria except invariance under permutation, and can therefore be seen as a quasi-geometric matrix mean.

3 Shuffled Inductive sequences

In this section, we illustrate on $\mathbb{P}_n$ the suboptimality of the choice $p_k = k \bmod N$ regarding the convergence rate of the CI-sequence, and we propose an alternative choice based on a shuffling procedure. We also prove the convergence of shuffled Inductive sequences on arbitrary NPC spaces to the least squares mean of the data.

3.1 Motivation on $\mathbb{P}_n$

Firstly, we motivate on $\mathbb{P}_n$ the shuffling in the specific case of three variables. Consider a tuple of data points $(A_1, A_2, A_3) \in \mathbb{P}_n^3$. Remember that the least squares mean is defined as the solution of an optimization problem:

$$\Lambda(A_1, A_2, A_3) = \underset{X \in \mathbb{P}_n}{\arg\min} \left(\delta^2(X, A_1) + \delta^2(X, A_2) + \delta^2(X, A_3) \right). \tag{5}$$

The three-variable Inductive mean $M_{\text{Ind}}(A_1, A_2, A_3)$, defined as

$$M_{\text{Ind}}(A_1, A_2, A_3) = (A_1 \#_{1/2} A_2) \#_{1/3} A_3,$$

can be characterized as the solution of another optimization problem as stated by Theorem 3.1, whose proof is given in Appendix B. In the cost function of this second optimization problem, the terms $\delta^2(A_1, X)$ and $\delta^2(A_2, X)$ of equation (5) are replaced by lower bounds, while the term $\delta^2(A_3, X)$ is unchanged. This shows that the Inductive mean tends to be localized closer to the third matrix A_3 than the least squares mean. Numerical experiments confirm this trend, which is also still visible when considering more matrices ($N \geq 3$).

Theorem 3.1. *When $N = 3$, $M_{\text{Ind}}(A_1, A_2, A_3)$ satisfies*

$$M_{\text{Ind}}(A_1, A_2, A_3) = \underset{X \in \mathbb{P}_n}{\arg\min} \ \left(d_M^2(A_1, X) + d_M^2(A_2, X) + \delta^2(A_3, X) \right) \tag{6}$$

with $M := A_1 \# A_2$ and $d_M(A, \cdot) : \mathbb{P}_n \to \mathbb{R}_+ : X \mapsto || \log(M^{\frac{-T}{2}} A M^{\frac{-1}{2}}) - \log(M^{\frac{-T}{2}} X M^{\frac{-1}{2}}) ||_F$. Moreover, for all $A, M \in \mathbb{P}_n$, the function $X \mapsto d_M(A, X)$ is upper bounded by the affine-invariant distance between A and X:

$$d_M(A, X) \leq \delta(A, X)$$

where the equality holds iff A and X commute.

This bias affects the convergence of the CI-sequence $(X_k^{\text{CI}})_{k \in \mathbb{N}_0}$ defined in the previous section. Indeed, Figure 1 illustrates how the ratios $\delta(X_k^{\text{CI}}, A_i)/\delta(K, A_i)$, $i = 1, \ldots, N$, evolve as we progress in the sequence (i.e., as k increases), in the three-variable case ($N = 3$). The black points correspond to steps $k = mN$, $m \in \mathbb{N}_0$, i.e, steps at which each data point has been visited the same number of times. Intermediary steps are represented in grey for better readability. The figure indicates that the last data point (matrix A_3) is systematically overemphasized by the algorithm: the distance ratio $\delta(X_k^{\text{CI}}, A_3)/\delta(K, A_3)$ remains inferior to 1 at steps $k = mN$, $m \in \mathbb{N}_0$. Observe also that the Inductive mean $M_{\text{Ind}}(A_1, \ldots, A_N)$ corresponds to the N^{th} point of the CI-sequence, hence to step $k = 3$ on Figure 1.

Further numerical experiments confirm this behavior for other numbers, sizes and types of SPD matrices.

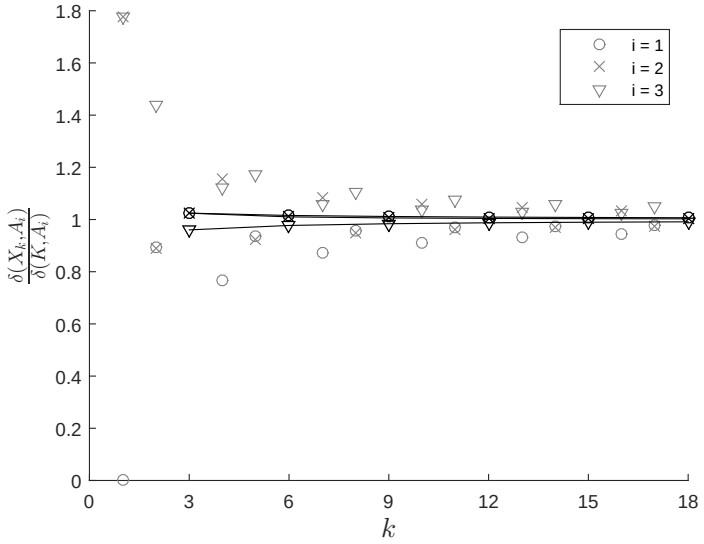

Figure 1: Average ratio between the distance from the points of the CI-sequence to the data and the distance from the least squares mean to the data, computed over 1000 sets of $N = 3$ matrices of size $n = 10$ generated according to the Wishart distribution $W_n(I, n)$.

3.2 Shuffling algorithm

To avoid the systematic bias appearing with the CI-sequence, we propose to use another sequence of indices p in Definition 2.1. Among the approaches investigated, the following shuffling algorithm was found numerically to yield good performances. We split the sequence p in frames of size N (with N the number of matrices in the data set): we define $p = (p^{(1)}, p^{(2)}, \dots)$, where each frame $p^{(k)} := (p_{(k-1)N+1}, \dots p_{kN})$, $k \in \mathbb{N}_0$, is a permutation of $(1, \dots, N)$ chosen according to Algorithm 1 below. The underlying idea consists in choosing $p^{(k)}$ to reduce the bias caused by older frames $p^{(j)}$, $j < k$.

Algorithm 1 Shuffling algorithm

Data: N, the number of data points.
1: $p^{(1)} = (1, \dots, N)$;
2: **for** $i = 1, 2, \dots$
3: $p^{(2i)} = \textbf{reverse}(p^{(2i-1)})$; see (7)
4: $p^{(2i+1)} = \textbf{in-shuffle}(p^{(2i-1)})$; see (8)
5: **end**

3.2.1 Reverse method

The **reverse** procedure invoked in line 3 of Algorithm 1 consists in reversing the sequence taken as input. Let $p := (p_1, \dots p_N)$, then :

$$\textbf{reverse}(p) = (p_N, p_{N-1}, ..., p_1). \tag{7}$$

For example, $\textbf{reverse}(1, 2, 3)$ gives the sequence $(3, 2, 1)$. In Algorithm 1, the first data points of $p^{(2i-1)}$ become the last ones in $p^{(2i)}$, and conversely. The goal is to force the data points that have been underemphasized in the previous frame to become overemphasized in the current one.

3.2.2 In-shuffle method

The **in-shuffle** procedure invoked in line 4 of Algorithm 1 refers to a specific type of riffle shuffle: a shuffling method often used for card shuffling. This method reorders a given sequence p containing $N = 2\tilde{N}$ elements as:

$$\textbf{in-shuffle}(p) = [p_{\tilde{N}+1}, p_1, p_{\tilde{N}+2}, p_2, \dots, p_{2\tilde{N}}, p_{\tilde{N}}]. \tag{8}$$

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 motivation behind this second shuffling step is to renew the set of matrices that will be either underemphasized or overemphasized by the algorithm (i.e., those corresponding to the first and last indices of the permutation).

3.3 Convergence proof

We prove the convergence of shuffled Inductive sequences on general NPC spaces towards the weighted least squares mean of the data. Our proof is valid under weak assumptions on the shuffling strategy: we only require p to be a sequence of permutations of the set $(1, \dots, N)$, which forces all data points to be regularly visited by the sequence.

Theorem 3.2. *Let $(\mathcal{M}, \delta)$ be a NPC space, $\mathbf{A} = (A_1, \dots, A_N) \in \mathcal{M}^N$ a tuple of data points and $\omega = (\omega_1, \dots, \omega_N)$ their respective weights. Let $(X_k)_{k\in\mathbb{N}_0}$ be defined as in Definition 2.1, choosing p as a sequence of permutations of $(1, \dots, N)$, i.e., $p = (p^{(1)}, p^{(2)}, \dots)$ where each $p^{(k)} := (p_{(k-1)N+1}, \dots, p_{kN})$ is a permutation of $(1, \dots, N)$. Choose also $t_k = \omega_{p_k} / \left(\sum_{i=1}^{k} \omega_{p_i}\right)$. Then,*

$$\lim_{k\to\infty} X_k = \Lambda(\omega; \mathbf{A})$$

where $\Lambda(\omega; \mathbf{A})$ is the weighted least squares mean defined by equation (3).

Proof. Our proof consists of two steps. We first prove the convergence of the subsequence $(X_{kN})_{k\in\mathbb{N}_0}$ to the least squares mean. In the second step, we prove that, for $j = 1, \dots, N-1$, $\delta(X_{kN+j}, X_{kN}) \to 0$. The conclusion directly follows.

Step 1: We prove the following inequality for all $k \in \mathbb{N}_0$:

$$\delta^2(\Lambda(\omega; \mathbf{A}), X_{kN}) \leq \frac{1}{k}\left[3\Delta(\mathbf{A})^2 + \sum_{i=1}^{N} \omega_i \delta^2(\Lambda(\omega; \mathbf{A}), A_i)\right], \tag{9}$$

where $\Delta(\mathbf{A})^2$ stands for the squared diameter of the data points: $\Delta(\mathbf{A})^2 = \max_{i,j} \delta(A_i, A_j)^2$. Since the interior of the brackets is constant, this implies convergence of $(X_{kN})_{k\in\mathbb{N}_0}$ to the least squares mean $\Lambda(\omega; \mathbf{A})$. Inequality (9) can be seen as a particular case of the convergence theorem in [LP14], asserting that the elements of the CI-sequence $(X_j^{\mathrm{CI}})_{j\in\mathbb{N}_0}$ (obtained by choosing $t_j = \omega_{p_j} / \left(\sum_{i=1}^{j} \omega_{p_i}\right)$ and $p_j = j \bmod N$, see Theorem (2.2)) satisfy the inequality

$$\delta^2(\Lambda(\omega; \mathbf{A}), X_j^{\mathrm{CI}}) \leq \frac{1}{\sum_{i=1}^{j} \omega_{p_i}}\left[3\Delta(\mathbf{A})^2 + \sum_{i=1}^{N} \omega_i \delta^2(\Lambda(\omega; \mathbf{A}), A_i)\right] \tag{10}$$

for any $j \in \mathbb{N}_0$. Indeed, if $j = kN$, $\sum_{i=1}^{j} \omega_{p_i} = k$ since p is chosen as a sequence of permutations of $(1, \dots, N)$. To prove inequality (9), we show that the tools required in the proof of inequality (10) are still valid in the case of shuffled sequences when $j = kN$.

The proof of inequality (10) presented in [LP14] works by induction on j. The inequality is verified for $j \leq N$, since

$$\delta^2(\Lambda(\omega; \mathbf{A}), X_j^{\mathrm{CI}}) \leq \Delta(\mathbf{A})^2 \leq 3\Delta(\mathbf{A})^2 + \sum_{i=1}^{N} \omega_i \delta^2(\Lambda(\omega; \mathbf{A}), A_i), \qquad j \leq N. \tag{11}$$

Indeed, on an NPC space, Inductive sequences stay inside the geodesic convex hull of the data points $(A_1, \dots, A_N)$, which is a subset of a metric ball of diameter $\Delta(\mathbf{A})$. Assuming that inequality (10) is verified for j, its validity is proven for $j + N$, which completes the proof. The main tools required by the induction are the semi-parallelogram law and the variance inequality.

To prove inequality (9), we observe that it is satisfied for $j = N$:

$$\delta^2(\Lambda(\omega; \mathbf{A}), X_N) \leq \Delta(\mathbf{A})^2 \leq 3\Delta(\mathbf{A})^2 + \sum_{i=1}^{N} \omega_i \delta^2(\Lambda(\omega; \mathbf{A}), A_i)$$

(for the same reason as inequality (11) above) and we prove that it remains true for $j = kN$ by the same arguments as in [LP14], the two main tools used in the proof (the semi-parallelogram law and the variance inequality) being still available since all the data points are visited when computing X_{kN} from $X_{(k-1)N}$ (i.e., $\{A_{p_{k(N-1)+1}}, \dots, A_{p_{kN}}\} = \{A_1, \dots, A_N\}$), for all $k \in \mathbb{N}_0$.

Step 2: We prove here that the following bound holds for all k:

$$\delta(X_{kN+j}, X_{kN}) \leq \frac{1}{k}\Delta(\mathbf{A}), \qquad j = 1, \dots, N-1. \tag{12}$$

Applying the triangle inequality yields:

$$\delta(X_{kN+j}, X_{kN}) \leq \sum_{i=1}^{j} \delta(X_{kN+i}, X_{kN+i-1}), \qquad j = 1, \dots, N-1. \tag{13}$$

Since

$$t_{kN+i} = \frac{\omega_{p_{kN+i}}}{\sum_{l=1}^{kN+i} \omega_{p_l}} = \frac{\omega_{p_{kN+i}}}{k + \sum_{l=1}^{i} \omega_{p_{kN+l}}} \leq \frac{\omega_{p_{kN+i}}}{k},$$

inequality (13) becomes, by definition of X_{kN+i}:

$$\begin{aligned}
\delta(X_{kN+j}, X_{kN}) &\leq \sum_{i=1}^{j} \delta(X_{kN+i}, X_{kN+i-1}) \\
&= \sum_{i=1}^{j} \delta(X_{kN+i-1} \#_{t_{kN+i}} A_{p_{kN+i}}, X_{kN+i-1}) \\
&= \sum_{i=1}^{j} t_{kN+i} \delta(A_{p_{kN+i}}, X_{kN+i-1}) \\
&\leq \frac{1}{k}\Delta(\mathbf{A}).
\end{aligned}$$

Now, because of inequalities (9) and (12), the whole sequence converges to the least squares mean as k increases:

$$\delta(\Lambda(\omega; \mathbf{A}), X_{kN+j}) \leq \delta(\Lambda(\omega; \mathbf{A}), X_{kN}) + \delta(X_{kN}, X_{kN+j}) \to_{k\to\infty} 0.$$

□

This proof can be easily generalized to shuffled sequences in which all data points are visited the same number of times every j frames, with $1 \leq j < \infty$, i.e., if:

$$(p_{(k-1)N+1}, \ldots, p_{(k-1+j)N}) \text{ is a permutation of } \underbrace{(1, \ldots, N, 1, \ldots, N, \ldots, 1, \ldots, N)}_{jN \text{ elements}}$$

for all $k \in \mathbb{N}_0$ and for some $j \in [1, \infty)$, with j fixed, independent on k.

3.4 Convergence speed comparison on $\mathbb{P}_n$

We illustrate on the cone of positive definite matrices the benefit of the shuffling method regarding the convergence speed of the sequence. We note $(X_k^{\mathrm{S}})_{k \in \mathbb{N}_0}$ the Inductive sequence (Definition 2.1) associated to the sequence of indices p built according to the proposed shuffling algorithm (Algorithm 1). We compare on Figure 2 the sequence $(X_k^{\mathrm{S}})_{k \in \mathbb{N}_0}$ with other Inductive sequences, including the CI-sequence of Theorem 2.2 and two random sequences. In the first one, each frame $p^{(j)}$ is made of N numbers p_k chosen randomly with replacement in the set $\{1, \ldots, N\}$ while in the second one the N indices p_k of each frame $p^{(j)}$ are chosen randomly among $\{1, \ldots, N\}$, without replacement. In our experiments, we use the following error definition:

$$\mathrm{E}_{\mathrm{rel}}(X, \mathbf{A}) = \frac{\delta(X, \Lambda(\mathbf{A}))}{\frac{1}{N}\sum_{i=1}^{N} \delta(A_i, \Lambda(\mathbf{A}))} \tag{14}$$

with $\delta(A, B)$ the affine-invariant distance, $X \in \mathbb{P}_n$ the estimate of the least squares mean and $\Lambda(\mathbf{A})$ the least squares mean, estimated here by a Steepest Descent algorithm taking as initial guess the arithmetic mean of the matrices and stopped when the following stopping criterion is reached:

$$\frac{||X^{(k+1)} - X^{(k)}||_F}{||X^{(k)}||_F} \leq 10^{-12}.$$

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/`). Given parameters (N, n, κ), we generate a dataset of N matrices of size n and of condition number approximately κ using the following lines of code:

```
for i = 1,..., N
    [Q,~] = qr(randn(n));
    D = diag([rand(1,floor(n/2))+1,(rand(1,n-floor(n/2))+1)/kappa]);
    A{i} = Q*D*Q';
end
```

All the error measures given in this paper were averaged over a large number of datasets.

The first row of Figure 2 illustrates the convergence of the sequences towards the least squares mean, for several values of (N, n, κ): the horizontal axis corresponds to the position in the sequence while the vertical axis gives the estimation error (see equation (14)), averaged over the datasets. For each combination of parameters considered, the shuffled Inductive sequence $(X_k^S)_{k \in \mathbb{N}_0}$ converges faster. The second row gives the average evolution of the error after passing 10 times on the data points (i.e., $\mathrm{E}_{\mathrm{rel}}(X_{10N}, \mathbf{A})$) with the parameters N, n and κ. For all the sets of parameters considered, the shuffled Inductive sequence $(X_k^{\mathrm{S}})_{k \in \mathbb{N}_0}$ achieves the best performance. Conversely, choosing the indices of each frame $p^{(j)}$ randomly with replacement among the set $\{1, \ldots, N\}$ leads to a particularly slow convergence compared to the other shuffling methods investigated.

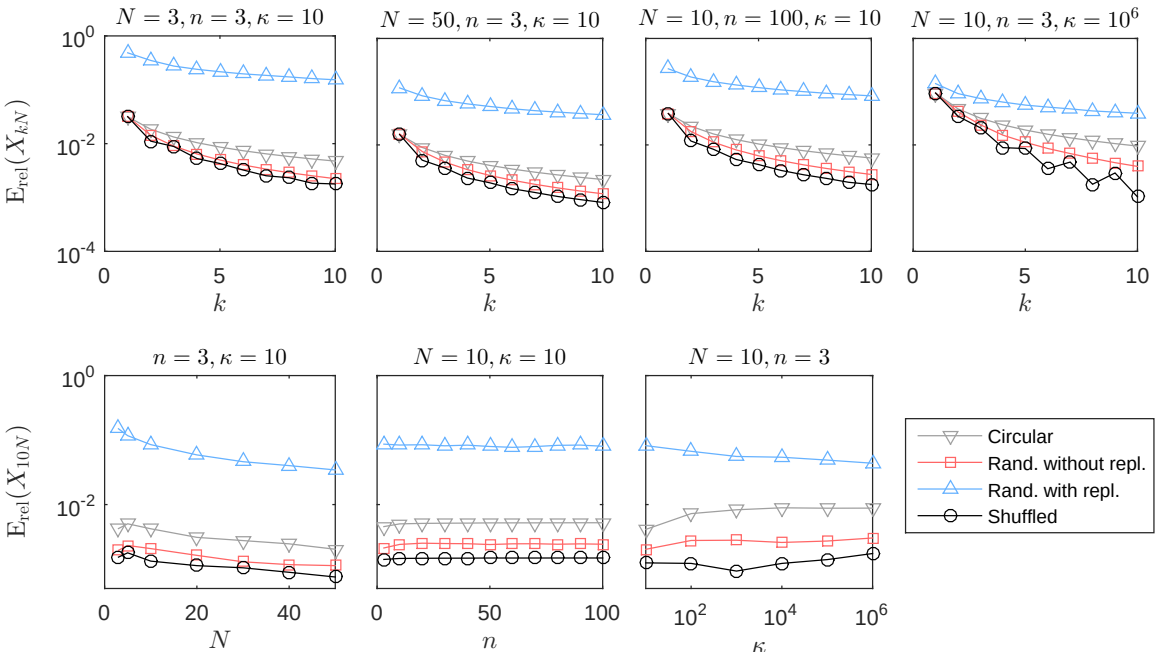

Figure 2: Comparison of the convergence rates of different Inductive sequences.

4 Quasi-geometric matrix means

In this section, we motivate the use of the points X^{S}_{kN}, with k finite, as means on $\mathbb{P}_n$. We first prove that those points are quasi-geometric matrix means: they satisfy most of the properties expected from geometric means.

Theorem 4.1. *For $k < \infty$, X^{S}_{kN} satisfies all the ALM properties except invariance under permutation.*

Proof. This result is well known for $k = 1$ since X^{S}_N is the Inductive mean [AKL07]. The case $k > 1$ can be obtained by induction on k, for example, for the monotonicity property, see the proof of Lemma 1 in [LL11]. The proof is similar for the other properties. □

Secondly, we illustrate on Figures 3 and 4 that the points X^{S}_{kN}, with k small, are already good estimates of the least squares mean: they outperform most other quasi-geometric matrix means regarding the proximity to the least squares mean versus computation time criterion. The means considered here include the Arithmetic, Cheap, Log-Euclidean and Arithmetic-Harmonic means (see the definitions below and [BI11, JV13]). All the experiments are performed with Matlab on a Windows 7 platform with 8 cores at 3.60 GHz and 16 GB ram.

The Cheap mean is a quasi-geometric matrix mean computed iteratively. The initial iterates are the N input matrices: $\tilde{A}^0_i = A_i, i = 1, \ldots, N$, and subsequent iterates are defined as:

$$\tilde{A}^{k+1}_i = \tilde{A}^k_i \exp\left(\frac{1}{N}\sum_{l=1, l\neq i}^{N} \log((\tilde{A}^k_i)^{-1}\tilde{A}^k_l)\right). \tag{15}$$

Each iteration of the Cheap mean has the cost of N gradient steps, which makes this method rather costly. However, it usually converges within numerical precision after only a few iterations (typically less than five). In our experiments, we used the implementation available in the Matrix Mean Toolbox, and we interrupted it after a fixed number of

iterations k_{Ch}, with $k_{\mathrm{Ch}} \in \{1, 2, \ldots, 5\}$. After k_{Ch} iteration, we aggregate the iterates $\tilde{A}_i^{k_{\mathrm{Ch}}}$, $i = 1, \ldots, N$, according to:

$$M_{\mathrm{Cheap}}(\mathbf{A}, k_{\mathrm{Ch}}) = \frac{1}{N} \sum_{i=1}^{N} \tilde{A}_i^{k_{\mathrm{Ch}}}. \tag{16}$$

The Arithmetic-Harmonic mean is defined as the geometric mean of the arithmetic and harmonic means:

$$M_{\mathrm{AH}}(\mathbf{A}) = G\left(\frac{1}{N} \sum_{i=1}^{N} A_i, \left(\frac{1}{N} \sum_{i=1}^{N} A_i^{-1} \right)^{-1} \right). \tag{17}$$

Finally, the Log-Euclidean mean is the result of a gradient step, taking as initial point I:

$$M_{\mathrm{LogEucl}}(\mathbf{A}) = \exp\left(\frac{1}{N} \sum_{i=1}^{N} \log(A_i) \right). \tag{18}$$

As illustrated on Figure 3, the Arithmetic, Arithmetic-Harmonic and Log-Euclidean means are rather far away from the least squares mean. Better performance can be achieved by running one iteration of the Cheap algorithm. However, Figure 4 indicates that this mean becomes particularly costly when the number of data points increases and seems to be strongly sensitive to the condition number of the data. The points X_{kN}^{S} of the Shuffled Inductive sequence reach a comparable accuracy without suffering from this drawback.

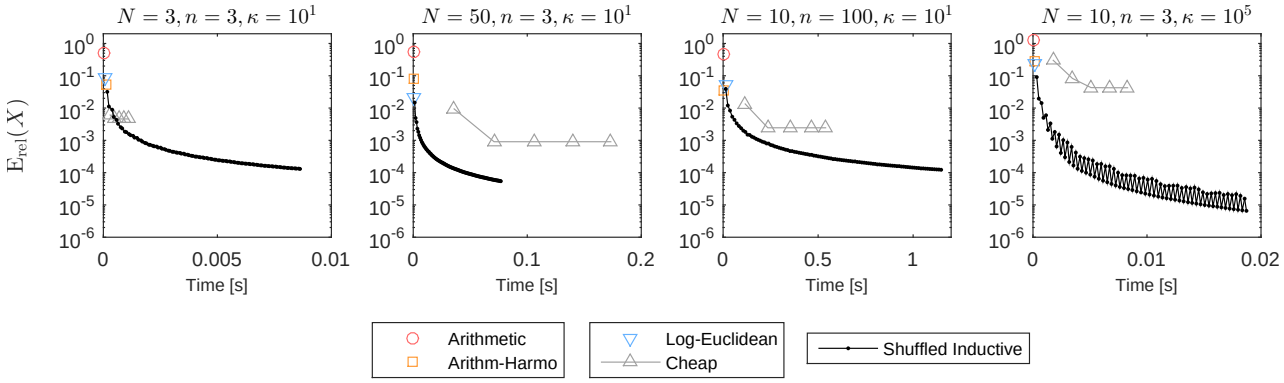

Figure 3: Comparison of several means as estimates of the least squares mean. For the Cheap mean, the triangles correspond to values of k_{Ch} ranging from 1 to 5, and for the Shuffled Inductive sequence the dots correspond to the points X_{kN}^{S} with $k = 1, \ldots, 100$ (the case $k = 1$ corresponding to M_{Ind}).

In the rest of this section, we use the points X_{kN}^{S} as initializers for optimization algorithms to compute the least squares mean. Since second-order algorithms have been showed to result in poor performance due to the sizable computation time of Hessian evaluations, we focus on lower order methods. We consider the Steepest Descent (SD) implementation provided in the Matrix Mean Toolbox, in which the step length is dynamically chosen depending on the condition number of the matrices arising in the expression of the gradient. We also consider the limited memory BFGS (LRBFGS) method: a Quasi-Newton algorithm in which an estimate of the Hessian is maintained in memory in the form of rank-two updates. In the limited memory variant, the spatial complexity is reduced by maintaining only the m most recent updates in memory. In [YHAG16], the authors showed that this algorithm outperforms the SD algorithm mentioned above for several problem settings

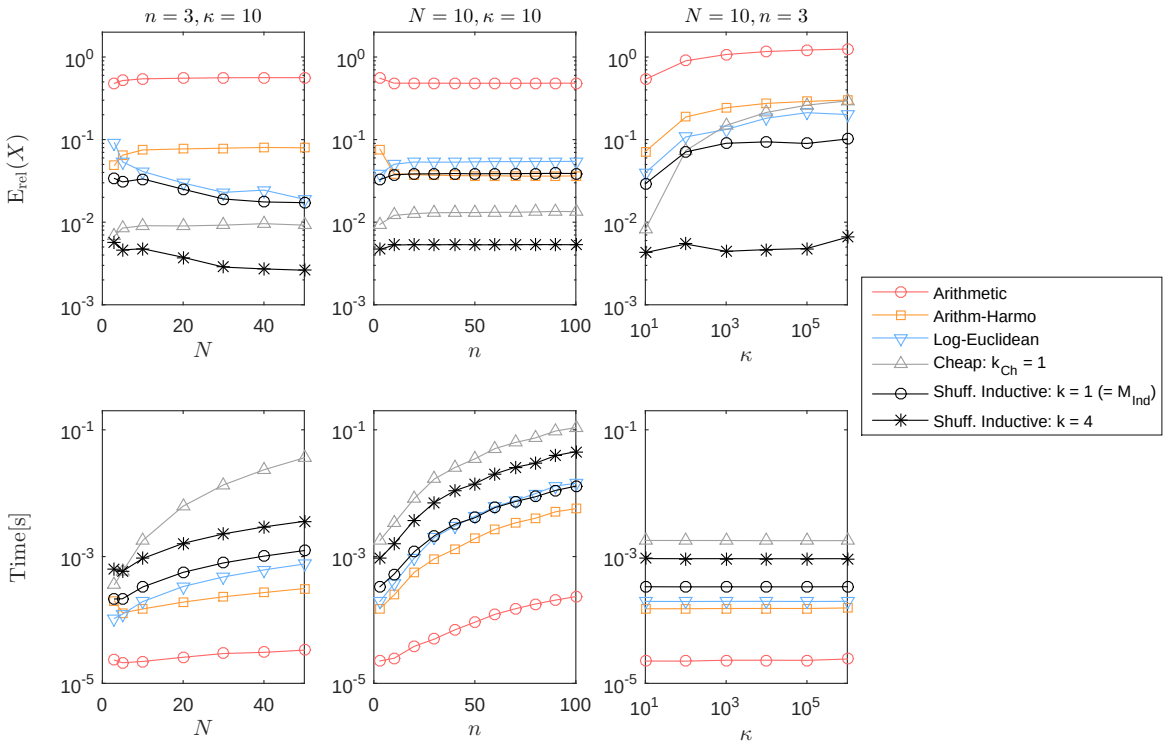

Figure 4: Comparison of several means as estimates of the least squares mean: evolution with the parameters (N, n, κ).

(e.g., when working with a large number of ill-conditioned matrices). We resort to their Matlab implementation of the LRBFGS method.

We investigate several sets of parameters, resulting respectively in Figures 5, 6 and 7. The first column of these figures illustrates the convergence of the SD and LRBFGS methods and compares it with the (sublinear) convergence of the Shuffled Inductive sequence X^{S}_{kN}. The graphs in this column also illustrate the gain obtained by choosing the Inductive mean as initial point for the methods. The second column is devoted to the SD algorithm. It illustrates the first iterates of the algorithm, for different initial points, chosen here as the points X^{S}_{kN}, for several choices of k. Finally, the third column presents the analogous graph for the LRBFGS method.

The three figures illustrate the nice behavior of the LRBFGS method when working with big or ill-conditioned matrices. Observe also that when the data are badly conditioned, it becomes less efficient to use the Inductive mean as initial point for optimization algorithms, and that points located further in the sequence (e.g., X^{S}_{2N} or X^{S}_{4N}) are better choices.

Finally, we illustrate on Figures 8 and 9 the evolution with the parameters of the problem of the CPU time required by respectively the SD and LRBFGS algorithms to reach a given accuracy. Inductive sequences provide good initial points for those algorithms. For example, the point X^{S}_{4N} results in a faster convergence of those optimization algorithms than other quasi-geometric means, in particular when the data matrices are ill-conditioned.

We end this section with a word of caution. When the matrix size n is large, the computation times are dominated by the BLAS calls, but for small values of n, the timings should be taken with a pinch of salt, since another programming language may lead to very different timings and possibly alter the conclusions.

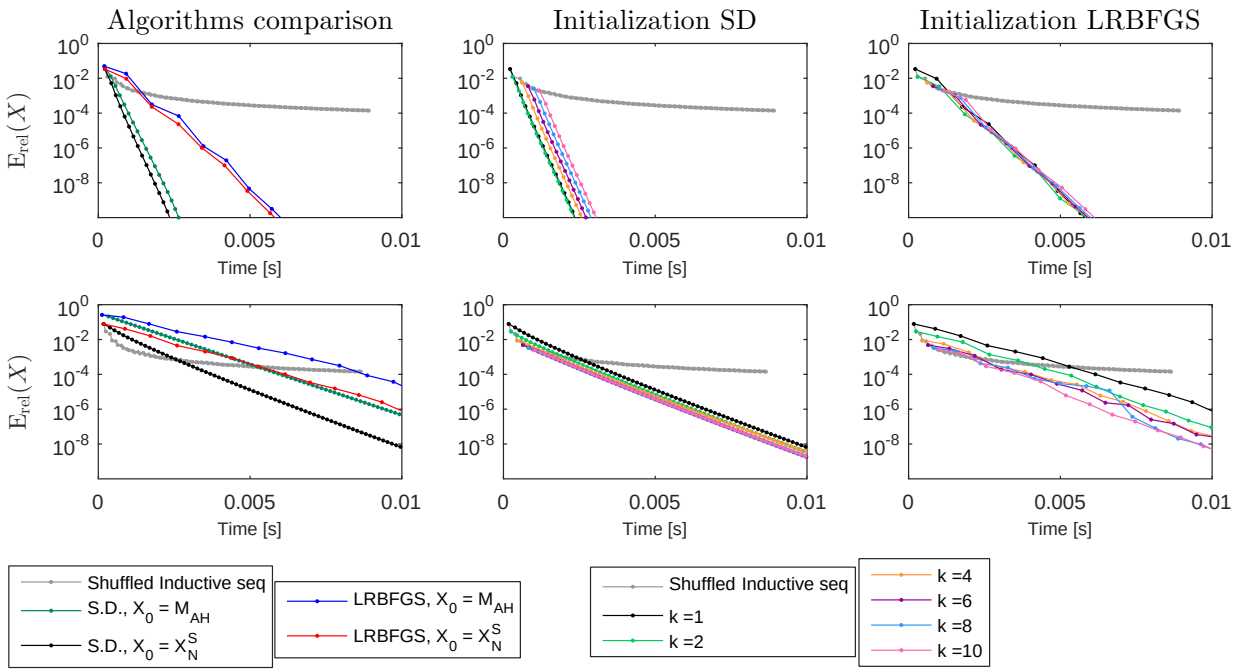

Figure 5: Algorithm comparison for approximating the least squares mean, for the parameters sets $(N, n, \kappa) = (3, 3, 10)$ (top row) and $(N, n, \kappa) = (3, 3, 10^5)$ (bottom row).

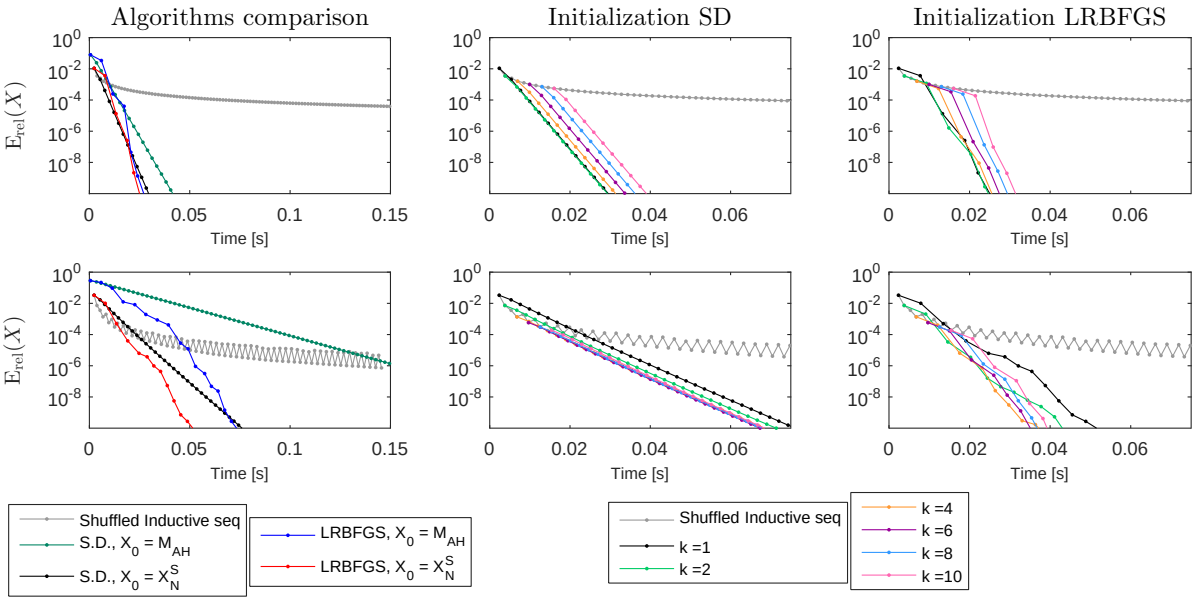

Figure 6: Algorithm comparison for approximating the least squares mean, for the parameters sets $(N, n, \kappa) = (100, 3, 10)$ (top row) and $(N, n, \kappa) = (100, 3, 10^5)$ (bottom row).

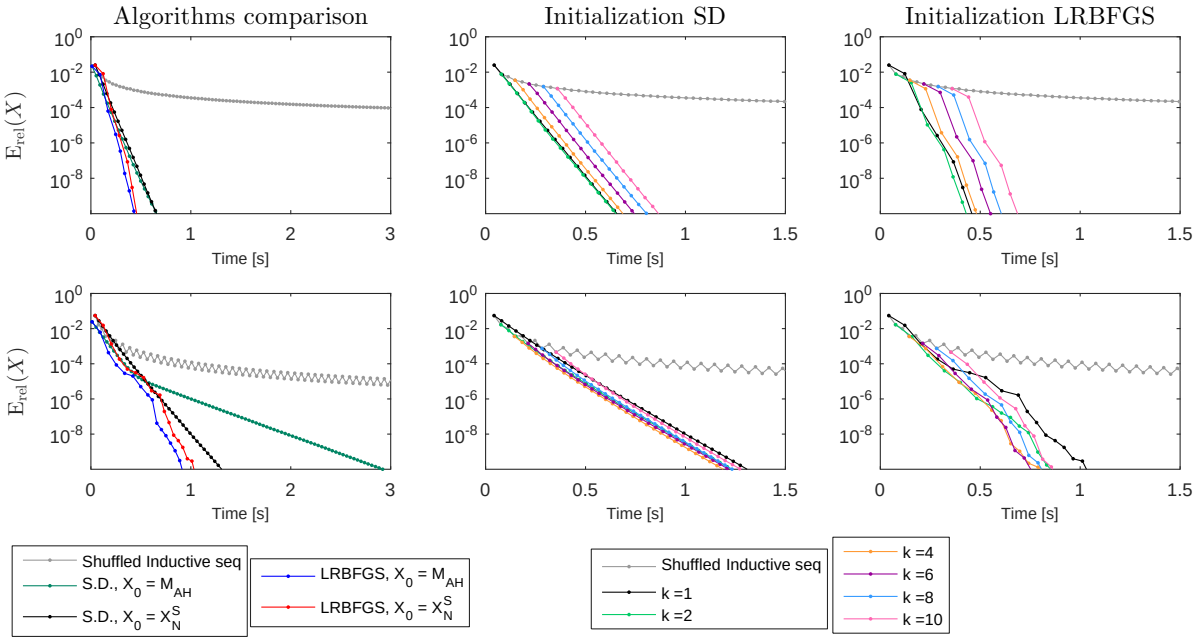

Figure 7: Algorithm comparison for approximating the least squares mean, for the parameters sets $(N, n, \kappa) = (30, 100, 10)$ (top row) and $(N, n, \kappa) = (30, 100, 10^5)$ (bottom row).

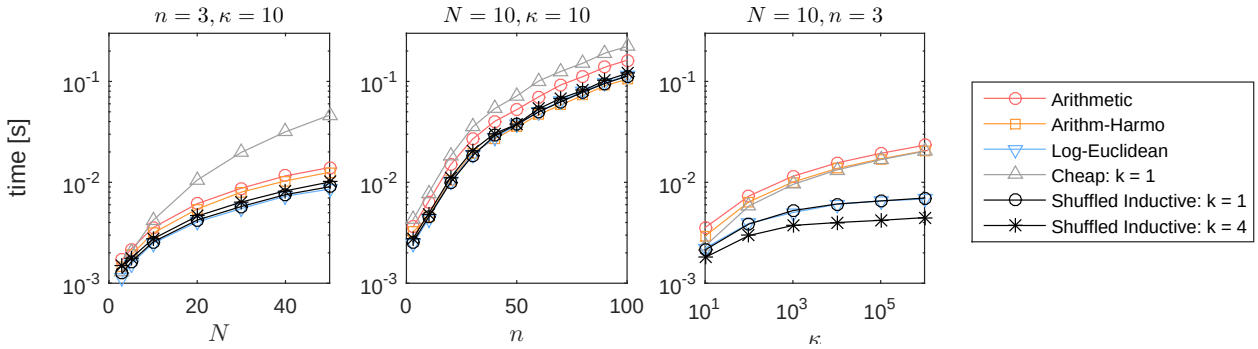

Figure 8: Computation time required by the SD algorithm to achieve a relative error $E_{\text{rel}} = 10^{-6}$, for various initial points.

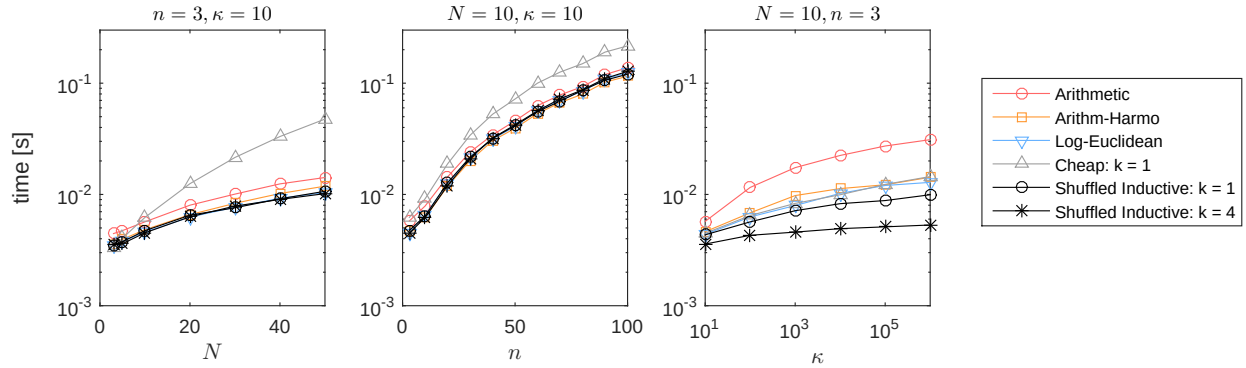

Figure 9: Computation time required by the LRBFGS algorithm to achieve a relative error $E_{\text{rel}} = 10^{-6}$, for various initial points.

5 Parallelizable variants

A commonly used approach for reducing the computation time of a program consists in resorting to parallelization techniques: the program is split in a collection of tasks that can be performed simultaneously by different processors. The values computed on each processor are then aggregated to deduce the final solution of the program. We propose here a parallelizable algorithm to estimate the least squares mean using Inductive means. Instead of taking sequentially steps towards data points, as required by Inductive sequences (Definition 2.1), we compute independently k inductive means (Definition 2.2), each one along a different permutation, and we aggregate the results to obtain an estimate of the least squares mean. These k inductive means can be computed simultaneously if a sufficient number of processors are available. Aggregating the means returned by each processor can be performed using a cheap matrix mean, e.g., the Arithmetic, Arithmetic-Harmonic or Inductive means defined above. We refer to the algorithm described here as the Aggregate Inductive mean, and note it Agg-Ind-<M>, where <M> stands for the mean used to average the k inductive means computed, i.e., <M> is the Arithmetic (M_{Ar}), Arithmetic-harmonic (M_{AH}) or Inductive (M_{Ind}) mean. For a detailed presentation, see Algorithm 2.

Algorithm 2 Aggregated Inductive means (Agg-Ind-<M>).

Data: $A_1, ..., A_N \in \mathbb{P}_n$, $k \in \mathbb{N}_0$ and <M> a cheap quasi-geometric mean: <M> $\in \{M_{\mathrm{Ar}}, M_{\mathrm{AH}}, M_{\mathrm{Ind}}\}$.

1: Generate a list $p = (p^{(1)}, \ldots, p^{(k)})$ of permutations of $(1, \ldots, N)$ according to Algorithm 1;
2: $M = 0$, if <M> $= M_{\mathrm{AH}}$: $Agg_a = 0 = Agg_h$;
3: **for** $i = 1, ..., k$;
4: $\quad B_i = M_{\mathrm{Ind}}(A_{p_1}, ..., A_{p_N})$;
5: $\quad$ **if** <M> $= M_{\mathrm{Ind}}$
6: $\quad\quad M \leftarrow M \# \frac{1}{i} B_i$;
7: $\quad$ **else if** <M> $= M_{\mathrm{AH}}$
8: $\quad\quad Agg_a \leftarrow \frac{i-1}{i} Agg_a + \frac{1}{i} B_i$;
9: $\quad\quad Agg_h \leftarrow \frac{i-1}{i} Agg_h + \frac{1}{i} B_i^{-1}$;
10: $\quad\quad M = Agg_a \# Agg_h^{-1}$;
11: $\quad$ **else if** <M> $= M_{\mathrm{Ar}}$
12: $\quad\quad M \leftarrow \frac{i-1}{i} M + \frac{1}{i} B_i$;
13: $\quad$ **end**
14: **end**
15: **return** M;

We suggest to choose $k = 2(\lceil \log_2 N \rceil - 1)$, which ensures that all the permutations generated in line 1 of Algorithm 2 are distinct. This follows from [DGK83]. The Agg-Ind-<M> algorithm satisfies part of the ALM properties:

Theorem 5.1.
1. The Agg-Ind-M_{Ar} 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.
2. The Agg-Ind-M_{AH} algorithm returns a matrix satisfying eight of the ten ALM properties, the violated properties being the invariance under permutation and the determinant equality.
3. The Agg-Ind-M_{Ind} algorithm returns a matrix satisfying the same ALM properties as

the Inductive mean, namely, the ten properties except invariance under permutation.
4. If in line 1 of Algorithm 2, the list p is chosen to be all the permutations of $(1, \ldots, N)$ *(hence* $k = N!$*), the Agg-Ind-<M> algorithm recovers invariance under permutation if* $<M> \in \{M_{\mathrm{Ar}}, M_{\mathrm{AH}}\}$.

Proof. The proof is similar to the one of Theorem 4.1, combined with the invariance properties of the Arithmetic and Arithmetic-Harmonic means. □

We compare on Figure 10 the aggregated means, for various choices of k and <M>, with the points X_{kN}^{S}, as estimates of the least squares mean. The parallelizable variants are slightly less efficient than the sequential approach. However, if enough (at least k) processors are available, the inductive means evaluated in Algorithm 2 (line 4) can be computed simultaneously, and the computation time of the Agg-Ind-<M> means would become comparable with that of $X_N^{\mathrm{S}} (= M_{\mathrm{Ind}})$. It is also interesting to note that conversely to the Shuffled Inductive sequences, the aggregated means do not converge to the least squares mean as k tends towards infinity in Algorithm 2. Indeed, for large values of k, the aggregate means are weighted <M>-means of the (at most $N!$) inductive means corresponding to the permutations $p^{(j)}$ considered, where <M> is the arithmetic, arithmetic-harmonic or inductive mean. Finally, the choice of the mean <M> does not seem to have a significant impact on the performance of the method when the matrices are well-conditioned. However, the choice $<M> = M_{\mathrm{Ar}}$ seems to be less efficient than the two other variants (i.e., $<M> \in \{M_{\mathrm{AH}}, M_{\mathrm{Ind}}\}$) when the condition number of the data is big.

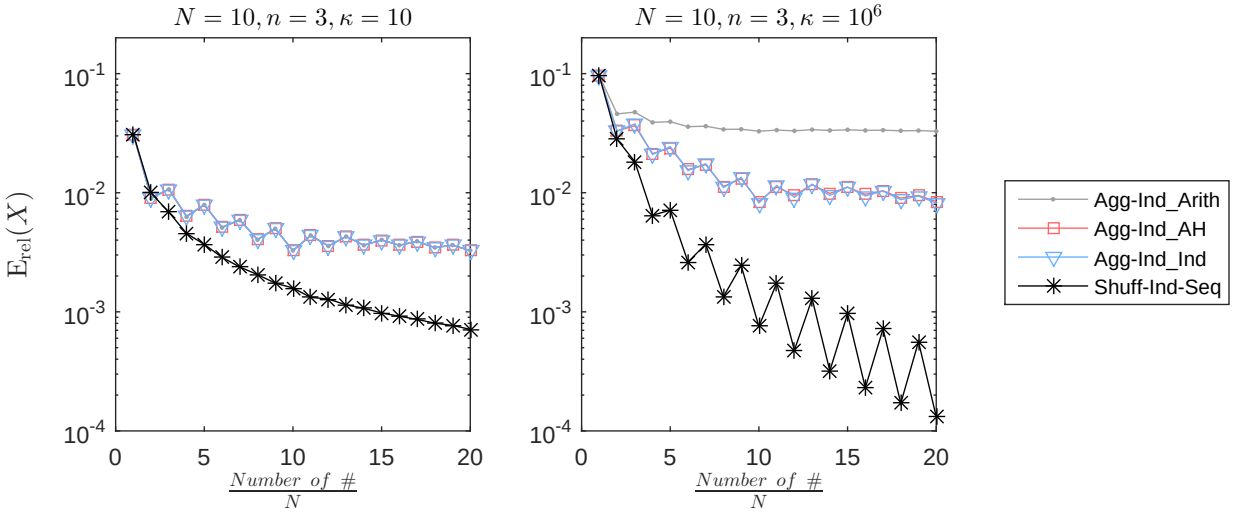

Figure 10: Comparison between the parallelizable and sequential variants. The x-axis is the total number of two-variable geometric means evaluated by the algorithm.

6 Conclusions

We have proposed new algorithms for approximating least squares means on NPC spaces using Inductive sequences. Those sequences are obtained by taking successive steps towards data points, with decreasing step lengths. They might be the only tool available on arbitrary NPC spaces for estimating least squares means. We first investigated convergence properties of the Cyclic Inductive sequence $(X_k^{\mathrm{CI}})_{k\in\mathbb{N}_0}$ proposed in the literature, which consists in visiting the data points according to the cyclic order $1, 2, \ldots, N, 1, 2, \ldots$ and has been proved to converge to the least squares mean of the data. We have illustrated on $\mathbb{P}_n$ that the subsequence $(X_{kN}^{\mathrm{CI}})_{k\in\mathbb{N}_0}$ is biased towards the last data points $A_N, A_{N-1}, \ldots,$ which has a negative impact on the convergence speed of the main sequence $(X_k^{\mathrm{CI}})_{k\in\mathbb{N}_0}$. As a remedy, we proposed to shuffle repeatedly the data points: we have defined a so-called Shuffled Inductive sequence $(X_k^{\mathrm{S}})_{k\in\mathbb{N}_0}$, in which for each $j \in \mathbb{N}_0$, the points $(X_{jN+k}^{\mathrm{S}})_{k=1,\ldots,N}$ are obtained by visiting the data points $(A_1, \ldots, A_N)$ along an order $p^{(j)}$, where $p^{(j)}$ is either defined as the reverse order of $p^{(j-1)}$ (if j is even) or as the result of another shuffling algorithm (the in-shuffle procedure described in Section 3). The underlying idea is to choose for points $(X_{jN+k}^{\mathrm{S}})_{k=1,\ldots,N}$ a permutation $p^{(j)}$ reducing the bias affecting the precedent point (here, the point (X_{jN}^{S})).

We have proved in the general framework of NPC spaces, which include the set $\mathbb{P}_n$, the convergence of shuffled Inductive sequences to the least squares mean of the data, under weak assumptions on the shuffling strategy. We have illustrated numerically on $\mathbb{P}_n$ the gain in convergence speed achieved by the shuffled sequence $(X_k^{\mathrm{S}})_{k\in\mathbb{N}_0}$. Numerical experiments indicate also that the first points of the subsequence $(X_{kN}^{\mathrm{S}})_{k\in\mathbb{N}_0}$ are good estimates of the least squares mean: they are non-dominated by other quasi-geometric means regarding the computation time vs distance to the least squares mean criterion, in particular when working with a large number of data matrices, or when the data are ill-conditioned. These experiments also show that those points are good initializers for state-of-the-art optimization algorithms (i.e., the Steepest Descent algorithm with automatic step length choice proposed in the Matrix Mean Toolbox and the LRBFGS algorithm proposed in [YHAG16]). Indeed, in several settings, these two algorithms take less time for reaching a given accuracy when initialized by the points $(X_{kN}^{\mathrm{S}})_{k\in\mathbb{N}_0}$ than when initialized with other quasi-geometric means. Again, the gain in performance is the most important when the data matrices are ill-conditioned. In future work, it would be interesting to study how the proposed Shuffled Inductive sequence performs on other types of synthetic data and on real-life data. Other types of shuffling methods could also be considered.

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: $\frac{1}{N}\sum_{i=1}^N A_i \geq \mathbf{G}(A_1, ..., A_N) \geq \left(\frac{1}{N}\sum_{i=1}^N A_i^{-1}\right)^{-1}$.

B Proof of Theorem 3.1

We prove here Theorem 3.1 given in Section 3.

Theorem B.1. *When $N = 3$, $M_{\text{Ind}}(A_1, A_2, A_3)$ satisfies*

$$M_{\text{Ind}}(A_1, A_2, A_3) = \arg\min_{X \in \mathbb{P}_n} \left(d_M^2(A_1, X) + d_M^2(A_2, X) + \delta^2(A_3, X) \right) \tag{19}$$

with $M := A_1 \# A_2$ and $d_M(A, \cdot) : \mathbb{P}_n \to \mathbb{R}_+ : X \mapsto ||\log(M^{\frac{-T}{2}} A M^{\frac{-1}{2}}) - \log(M^{\frac{-T}{2}} X M^{\frac{-1}{2}}))||_F$. Moreover, for all A and $M \in \mathbb{P}_n$, the function $d_M(A, \cdot)$ is upper bounded by the Riemannian distance between A and X:

$$d_M(A, X) \leq \delta(A, X) \tag{20}$$

where the equality holds iff A and X commute.

Proof. We first prove that the Inductive mean is the solution to the optimization problem (19). Let $M := A_1 \# A_2$. The definition of the Inductive mean gives $M_{\text{Ind}}(A_1, A_2, A_3) = M \#_{1/3} A_3$. Because of the uniqueness of the weighted 2-variable geometric mean, M_{Ind} is also the weighted least squares mean of matrices M and A_3:

$$M_{\text{Ind}}(A_1, A_2, A_3) = \arg\min_{X \in \mathbb{P}_n} \left(\frac{2}{3} \delta^2(M, X) + \frac{1}{3} \delta^2(A_3, X) \right).$$

We now rewrite the distance $\delta(M, X)$, to introduce a dependency on A_1 and A_2. The main tool we will use is the parallelogram equality:

Lemma B.1 (Parallelogram equality, [You88,Bha09])**.** *For $a, b, x \in \mathbb{R}^{n \times n}$ and $c = (a+b)/2$, the parallelogram law gives:*

$$||c - x||_F^2 = \frac{1}{2}||a - x||_F^2 + \frac{1}{2}||b - x||_F^2 - \frac{1}{4}||a - b||_F^2.$$

However, this law is only valid on Euclidean spaces. We therefore first map all the matrices involved on an Euclidean space, and use invariance properties of the Riemannian distance to rewrite $\delta(M, X)$ as an Euclidean distance, which can be summarized by the following Lemma.

Lemma B.2. *Consider the congruence transformation $\Gamma_M(A) : \mathbb{P}_n \to \mathbb{P}_n : A \mapsto M^{\frac{-T}{2}} A M^{\frac{-1}{2}}$, and define*

$$\tilde{X} = \log\left(\Gamma_M(X)\right) = \log\left(M^{\frac{-T}{2}} X M^{\frac{-1}{2}}\right)$$
$$\tilde{M} = \log\left(\Gamma_M(M)\right) = \log\left(I\right) = 0$$

Then, the Euclidean distance between $\tilde{M}$ and $\tilde{X}$ is equal to the Riemannian distance between M and X:

$$\delta(M, X) = ||\tilde{M} - \tilde{X}||_F.$$

Proof. Using the invariance under congruence property of the affine-invariant metric [Bha09], we obtain:

$$\delta(M, X) = \delta\left(\Gamma_M(M), \Gamma_M(X)\right) = \delta\left(I, \Gamma_M(X)\right) = ||\log(\Gamma_M(X))||_F = ||\tilde{M} - \tilde{X}||_F.$$

□

Similarly as in Lemma B.2, we define $\tilde{A}_i = \log\left(\Gamma_M(A_i)\right) = \log\left(M^{\frac{-T}{2}} A_i M^{\frac{-1}{2}}\right)$, $i = 1, 2, 3$. Remember that M is the geometric mean of A_1 and A_2. Because of the invariance under congruence property of the matrix geometric mean, $\Gamma_M(M) = I$ is the geometric mean of $\Gamma_M(A_1)$ and $\Gamma_M(A_2)$, and due to Proposition 6.1.8 of [Bha09], $\tilde{M} = 0$ is the midpoint of the segment between $\tilde{A}_1$ and $\tilde{A}_2$. Hence, we can apply the parallelogram equality to rewrite $||\tilde{M} - \tilde{X}||_F$ as:

$$||\tilde{M} - \tilde{X}||_F^2 = \tfrac{1}{2}||\tilde{A}_1 - \tilde{X}||_F^2 + \tfrac{1}{2}||\tilde{A}_2 - \tilde{X}||_F^2 - \tfrac{1}{4}||\tilde{A}_1 - \tilde{A}_2||_F^2.$$

The last term of the right hand side does not depend on variable $\tilde{X}$. So, we obtain:

$$\begin{aligned} M_{\text{Ind}} &= \arg\min_{X \in \mathbb{P}_n} \left(\frac{2}{3}\delta^2(M, X) + \frac{1}{3}\delta^2(A_3, X) \right) \\ &= \arg\min_{X \in \mathbb{P}_n} \frac{1}{3} \left(||\tilde{A}_1 - \tilde{X}||_F^2 + ||\tilde{A}_2 - \tilde{X}||_F^2 + \delta^2(A_3, X) \right) \\ &= \arg\min_{X \in \mathbb{P}_n} \frac{1}{3} \left(d_M^2(A_1, X) + d_M^2(A_2, X) + \delta^2(A_3, X) \right). \end{aligned}$$

We now prove that $d_M(A, X) \leq \delta(A, X)$ for any set of matrices $A, X, M \in \mathbb{P}_n$. This follows from the Exponential Metric Increasing property [Bha09]: if matrices $A, X \in \mathbb{P}_n$ do not commute, then

$$d_M^2(A, X) = ||\tilde{A} - \tilde{X}||_F^2 < \delta^2(\Gamma(A), \Gamma(X)) = \delta^2(A, X) \tag{21}$$

If instead A and X commute, then the strict inequality in equation (21) becomes an equality. □

References

[AKL07] Eunkyung Ahn, Sejung Kim, and Yongdo Lim. An extended lie–trotter formula and its applications. *Linear Algebra and Its Applications*, 427(2):190–196, 2007.

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

[Bac14] Miroslav Bacák. Computing medians and means in Hadamard spaces. *SIAM Journal on Optimization*, 24(3):1542–1566, 2014.

[Bha09] Rajendra Bhatia. *Positive definite matrices*. Princeton university press, 2009.

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

[BI13] Dario 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.

[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.

[Hol12] John Holbrook. No dice: a deterministic approach to the Cartan centroid. *Journal of the Ramanujan Mathematical Society*, 27(4):509–521, 2012.

[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`.

[LP14] Yongdo Lim and Miklós Pálfia. Weighted deterministic walks for the least squares mean on Hadamard spaces. *Bulletin of the London Mathematical Society*, 46(3):561–570, 2014.

[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.

[RA11] Quentin Rentmeesters and P-A Absil. Algorithm comparison for Karcher mean computation of rotation matrices and diffusion tensors. In *Signal Processing Conference, 2011 19th European*, pages 2229–2233. IEEE, 2011.

[ST94] Masahiko Sagae and Kunio Tanabe. Upper and lower bounds for the arithmetic-geometric-harmonic means of positive definite matrices. *Linear and Multilinear Algebra*, 37(4):279–282, 1994.

[Stu03] Karl-Theodor Sturm. Probability measures on metric spaces of nonpositive curvature. *Heat Kernels and Analysis on Manifolds, Graphs, and Metric Spaces: Lecture Notes from a Quarter Program on Heat Kernels, Random Walks, and Analysis on Manifolds and Graphs: April 16-July 13, 2002, Emile Borel Centre of the Henri Poincaré Institute, Paris, France*, 338:357, 2003.

[YHAG16] Xinru Yuan, Wen Huang, P-A Absil, and Kyle A. Gallivan. A Riemannian limited-memory BFGS algorithm for computing the matrix geometric mean. *Procedia Computer Science*, 80:2147–2157, 2016.

[You88] Nicholas Young. *An introduction to Hilbert space*. Cambridge university press, 1988.

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