Riemannian Optimization and the Computation of the Divergences and the Karcher Mean of Symmetric Positive Definite Matrices

Xinru Yuan¹, Wen Huang², Kyle A. Gallivan¹, and P.-A. Absil³

¹, Florida State University    ², Rice University    ³, Université Catholique de Louvain

May 7, 2018
SIAM Conference on Applied Linear Algebra
Definition

A symmetric matrix $A$ is called **positive definite** iff all its eigenvalues are positive.

### 2 × 2 SPD matrix

- $u_{\sqrt{\lambda_u}}$
- $v_{\sqrt{\lambda_v}}$

### 3 × 3 SPD matrix

- $u_{\sqrt{\lambda_u}}$
- $v_{\sqrt{\lambda_v}}$
- $w_{\sqrt{\lambda_w}}$
Motivation of Averaging SPD Matrices

Possible applications of SPD matrices

- Diffusion tensors in medical imaging [CSV12, FJ07, RTM07]
- Describing images and video [LWM13, SFD02, ASF+05, TPM06, HWSC15]

Motivation of averaging SPD matrices

- Aggregate several noisy measurements of the same object
- Subtask in interpolation methods, segmentation, and clustering
Let $A_1, \ldots, A_K$ be SPD matrices.

- Generalized arithmetic mean: \[ \frac{1}{K} \sum_{i=1}^{K} A_i \]
  \[ \rightarrow \text{Not appropriate in many practical applications} \]
Let $A_1, \ldots, A_K$ be SPD matrices.

- **Generalized arithmetic mean:** $\frac{1}{K} \sum_{i=1}^{K} A_i$

  → Not appropriate in many practical applications

\[
\det A = 50 \quad \text{det} \left( \frac{A+B}{2} \right) = 267.56 \quad \det B = 50
\]
Averaging Schemes: from Scalars to Matrices

Let $A_1, \ldots, A_K$ be SPD matrices.

- Generalized arithmetic mean: $\frac{1}{K} \sum_{i=1}^{K} A_i$
  
  → Not appropriate in many practical applications

- Generalized geometric mean: $(A_1 \cdots A_K)^{1/K}$
  
  → Not appropriate due to non-commutativity
  
  → How to define a matrix geometric mean?
The desired properties are given in the ALM list\(^1\), some of which are:

1. \( G(A_{\pi(1)}, \ldots, A_{\pi(K)}) = G(A_1, \ldots, A_K) \) with \( \pi \) a permutation of \((1, \ldots, K)\)
2. if \( A_1, \ldots, A_K \) commute, then \( G(A_1, \ldots, A_K) = (A_1, \ldots, A_K)^{1/K} \)
3. \( G(A_1, \ldots, A_K)^{-1} = G(A_1^{-1}, \ldots, A_K^{-1}) \)
4. \( \det(G(A_1, \ldots, A_K)) = (\det(A_1) \cdots \det(A_K))^{1/K} \)

A well-known mean on the manifold of SPD matrices is the \textbf{Karcher mean} [Kar77]:

\[
G(A_1, \ldots, A_K) = \arg \min_{X \in S^n_{++}} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i),
\]

(1)

where \( \delta(X, Y) = \| \log(X^{-1/2} Y X^{-1/2}) \|_F \) is the geodesic distance under the affine-invariant metric

\[
g(\eta_X, \xi_X) = \text{trace}(\eta_X X^{-1} \xi_X X^{-1})
\]

The Karcher mean defined in (1) satisfies all the geometric properties in the ALM list [LL11]
\[ G(A_1, \ldots, A_k) = \arg \min_{X \in S_+^n} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \]

- Riemannian steepest descent [RA11, Ren13]
- Riemannian Barzilai-Borwein method [IP15]
- Riemannian Newton method [RA11]
- Richardson-like iteration [BI13]
- Riemannian steepest descent, conjugate gradient, BFGS, and trust region Newton methods [JVV12]
- Limited-memory Riemannian BFGS method [YHAG16]
Conditioning of the Objective Function

Hemstitching phenomenon for steepest descent

well-conditioned Hessian

ill-conditioned Hessian

- **Small** condition number $\Rightarrow$ **fast** convergence
- **Large** condition number $\Rightarrow$ **slow** convergence
Conditioning of the Karcher Mean Objective Function

- **Riemannian metric:**

  \[ g_X(\xi, \eta) = \text{trace}(\xi X^{-1} \eta X^{-1}) \]

- **Euclidean metric:**

  \[ g_X(\xi, \eta) = \text{trace}(\xi \eta) \]

**Condition number \( \kappa \) of Hessian at the minimizer \( \mu \):**

- **Hessian of Riemannian metric:**

  \[ - \kappa(H^R) \leq 1 + \frac{\ln(\max \kappa_i)}{2}, \text{ where } \kappa_i = \kappa(\mu^{-1/2} A_i \mu^{-1/2}) \]

  \[ - \kappa(H^R) \leq 20 \text{ if } \max(\kappa_i) = 10^{16} \]

- **Hessian of Euclidean metric:**

  \[ - \frac{\kappa^2(\mu)}{\kappa(H^R)} \leq \kappa(H^E) \leq \kappa(H^R) \kappa^2(\mu) \]

  \[ - \kappa(H^E) \geq \kappa^2(\mu)/20 \]
BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

- **Update formula:**
  \[ x_{k+1} = x_k + \alpha_k \eta_k \]

- **Search direction:**
  \[ \eta_k = -B_k^{-1} \text{grad } f(x_k) \]

- **\( B_k \) update:**
  \[ B_{k+1} = B_k - B_k s_k s_k^T B_k + \frac{y_k y_k^T}{y_k^T s_k} \]
  where \( s_k = x_{k+1} - x_k \), and \( y_k = \text{grad } f(x_{k+1}) - \text{grad } f(x_k) \)
BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

- **Update formula:**
  \[ x_{k+1} = x_k + \alpha_k \eta_k \]

- **Search direction:**
  \[ \eta_k = -B_k^{-1} \nabla f(x_k) \]

- **\( B_k \) update:**
  \[
  B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},
  \]

  where \( s_k = x_{k+1} - x_k \), and \( y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \)
BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

- **Update formula:**
  \[ x_{k+1} = x_k + \alpha_k \eta_k \]

- **Search direction:**
  \[ \eta_k = -B_k^{-1} \nabla f(x_k) \]

- **\( B_k \) update:**
  \[ B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} \]

  where \( s_k = x_{k+1} - x_k \), and \( y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \)

**Replace by** \( R_{x_k}(\eta_k) \)

**Optimization on a Manifold**

**Means of SPD Matrices**
Riemannian BFGS (RBFGS) Algorithm

- Update formula:
  
  \[ x_{k+1} = R_{x_k}(\alpha_k \eta_k) \text{ with } \eta_k = -B_k^{-1} \text{grad } f(x_k) \]

- \( B_k \) update [HGA15]:
  
  \[
  B_{k+1} = \tilde{B}_k - \frac{\tilde{B}_k s_k (\tilde{B}_k s_k)^b}{(\tilde{B}_k s_k)^b s_k} + \frac{y_k y_k^b}{y_k^b s_k},
  \]

  where \( s_k = T_{\alpha_k \eta_k} \alpha_k \eta_k \), \( y_k = \beta_k^{-1} \text{grad } f(x_{k+1}) - T_{\alpha_k \eta_k} \text{grad } f(x_k) \),
  and \( \tilde{B}_k = T_{\alpha_k \eta_k} \circ B_k \circ T_{\alpha_k \eta_k}^{-1} \).

- Stores and transports \( B_k^{-1} \) as a dense matrix

- Requires excessive computation time and storage space for large-scale problem
Riemannian BFGS:

\[ B_{k+1} = \tilde{B}_k - \frac{\tilde{B}_k s_k (\tilde{B}_k s_k)^b}{(\tilde{B}_k s_k)^b s_k} + \frac{y_k y_k^b}{y_k^b s_k}, \]

where \( s_k = T_{\alpha_k \eta_k} \alpha_k \eta_k \), \( y_k = \beta_k^{-1} \text{grad } f(x_{k+1}) - T_{\alpha_k \eta_k} \text{grad } f(x_k) \),

and \( \tilde{B}_k = T_{\alpha_k \eta_k} \circ B_k \circ T_{\alpha_k \eta_k}^{-1} \)

Limited-memory Riemannian BFGS:

- Stores only the \( m \) most recent \( s_k \) and \( y_k \)
- Transports those vectors to the new tangent space rather than the entire matrix \( B_k^{-1} \)
- Computational and storage complexity depends upon \( m \)
Implementations

- Representations of tangent vectors
- Retraction
- Vector transport
Implementations

- Representations of tangent vectors: \( T_X S_{++}^n = \{ S \in \mathbb{R}^{n \times n} | S = S^T \} \)
  - Extrinsic representation: \( n^2 \)-dimensional vector
  - Intrinsic representation: \( d \)-dimensional vector where \( d = n(n+1)/2 \)

- Retraction

- Vector transport
Implementations

- Representations of tangent vectors: \( T_X S_{++} = \{ S \in \mathbb{R}^{n \times n} | S = S^T \} \)
  - Extrinsic representation: \( n^2 \)-dimensional vector
  - Intrinsic representation: \( d \)-dimensional vector where \( d = n(n + 1)/2 \)

- Retraction
  - Exponential mapping: \( \text{Exp}_X(\xi) = X^{1/2} \exp(X^{-1/2} \xi X^{-1/2}) X^{1/2} \)

- Vector transport
  - Parallel translation: \( T_{p\eta}(\xi) = Q \xi Q^T \), with \( Q = X^{1/2} \exp(\frac{X^{-1/2} \eta X^{-1/2}}{2}) X^{-1/2} \)
Implementations

Representations of tangent vectors: \( T_X S_{++} = \{ S \in \mathbb{R}^{n \times n} | S = S^T \} \)

- Extrinsic representation: \( n^2 \)-dimensional vector
- Intrinsic representation: \( d \)-dimensional vector where \( d = n(n + 1)/2 \)

Retraction

- Exponential mapping: \( \text{Exp}_X(\xi) = X^{1/2} \exp(X^{-1/2}\xi X^{-1/2})X^{1/2} \)
- Second order approximation retraction [JVV12]:
  \[ R_X(\xi) = X + \xi + \frac{1}{2} \xi X^{-1} \xi \]

Vector transport

- Parallel translation: \( T_{p_\eta}(\xi) = Q\xi Q^T \), with \( Q = X^{1/2} \exp\left(\frac{X^{-1/2}\eta X^{-1/2}}{2}\right)X^{-1/2} \)
- Vector transport by parallelization [HAG15]: essentially an identity

Means of SPD Matrices
Complexity Comparison for LRBFGS

Extrinsic approach:
- Function
- Riemannian gradient

Intrinsic approach:
- Function
- Riemannian gradient

Both approaches have the same complexities: $f + \nabla f$ cost
**Extrinsic approach:**
- Function
- Riemannian gradient
- Retraction
  - Evaluate $R_X(\eta)$

**Intrinsic approach:**
- Function
- Riemannian gradient
- Retraction
  - Compute $\eta$ from $\tilde{\eta}^d$
  - Evaluate $R_X(\eta)$

Intrinsic cost $= \text{Extrinsic cost} + 2n^3 + o(n^3)$
Complexity Comparison for LRBFGS

Extrinsic approach:
- Function
- Riemannian gradient
- Retraction
- Riemannian metric
  \(- 6n^3 + o(n^3)\)

Intrinsic approach:
- Function
- Riemannian gradient
- Retraction
- Reduces to Euclidean metric
  \(- n^2 + o(n^2)\)
Complexity Comparison for LRBFGS

Extrinsic approach:
- Function
- Riemannian gradient
- Retraction
- Riemannian metric
- \((2m)\) times of vector transport

Intrinsic approach:
- Function
- Riemannian gradient
- Retraction
- Reduces to Euclidean metric
- No explicit vector transport
Complexity Comparison for LRBFGS

Extrinsic approach:
- Function
- Riemannian gradient
- Retraction
- Riemannian metric
- \((2m)\) times of vector transport

Intrinsic approach:
- Function
- Riemannian gradient
- Retraction
- Reduces to Euclidean metric
- No explicit vector transport

Complexity comparison:

Extrinsic:
\[
f + \nabla f + 27n^3 + 12mn^2 + 2m \times \text{Vector transport cost}
\]

Intrinsic:
\[
f + \nabla f + \frac{22n^3}{3} + 4mn^2
\]
Numerical Results: Comparison of Different Algorithms

\( K = 100, \text{ size } = 3 \times 3, d = 6 \)

- \( 1 \leq \kappa(A_i) \leq 200 \)

- \( 10^3 \leq \kappa(A_i) \leq 10^7 \)

**Figure:** Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Means of SPD Matrices
Numerical Results: Comparison of Different Algorithms

\[ K = 30, \text{ size } = 100 \times 100, d = 5050 \]

- \[ 1 \leq \kappa(A_i) \leq 20 \]

- \[ 10^4 \leq \kappa(A_i) \leq 10^7 \]

**Figure**: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations
Numerical Results: Riemannian vs. EuclideanMetrics

- $K = 100$, $n = 3$, and $1 \leq \kappa(A_i) \leq 10^6$.

- $K = 30$, $n = 100$, and $1 \leq \kappa(A_i) \leq 10^5$.

**Figure:** Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations.
Outline

- Karcher mean computation on $S^n_{++}$
- Divergence-based means on $S^n_{++}$
- $L^1$-norm median computation on $S^n_{++}$
- Application: Structure tensor image denoising
- Summary
Motivations

- Karcher mean

\[ K(A_1, \ldots, A_K) = \arg \min_{X \in S^n_{++}} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{1} \]

where \( \delta(X, Y) = \| \log(X^{-1/2} Y X^{-1/2}) \|_F \)

- Pros: holds desired properties
- Cons: high computational cost

- Use **divergences** as alternatives to the geodesic distance due to their computational and empirical benefits

- A **divergence** is like a distance except it lacks
  - triangle inequality
  - symmetry
The LogDet $\alpha$-divergence is defined as

$$G(A_1, \ldots, A_k) = \arg \min_{X \in S^n_{++}} \frac{1}{2K} \sum_{i=1}^{K} \delta^2_{LD,\alpha}(A_i, X),$$

where the LogDet $\alpha$-divergence on $S^n_{++}$ is given by

$$\delta^2_{LD,\alpha}(X, Y) = \frac{4}{1 - \alpha^2} \log \frac{\det(\frac{1-\alpha}{2} X + \frac{1+\alpha}{2} Y)}{\det(X)^{\frac{1-\alpha}{2}} \det(Y)^{\frac{1+\alpha}{2}}}$$

- The LogDet $\alpha$-divergence is asymmetric in general, except for $\alpha = 0$

- (2) defines the right mean. The left mean can be defined in a similar way.
Karcher Mean vs. LogDet $\alpha$-divergence Mean

- **Complexity comparison for problem-related operations**

<table>
<thead>
<tr>
<th></th>
<th>function</th>
<th>gradient</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD $\alpha$-div. mean</td>
<td>$\frac{2Kn^3}{3}$</td>
<td>$3Kn^3$</td>
<td>$\frac{11Kn^3}{3}$</td>
</tr>
<tr>
<td>Karcher mean</td>
<td>$18Kn^3$</td>
<td>$5Kn^3$</td>
<td>$23Kn^3$</td>
</tr>
</tbody>
</table>

- **Invariance properties**

<table>
<thead>
<tr>
<th></th>
<th>scaling invariance</th>
<th>rotation invariance</th>
<th>congruence invariance</th>
<th>inversion invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD $\alpha$-div. mean</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Karcher mean</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
\( K = 100, n = 3, \) and \( 10 \leq \kappa(A_i) \leq 10^6 \)


**Numerical Experiment: Comparisons of Different Algorithms**

---

**Fixed Point iteration**

\[ \alpha = 0.9 \]

\[ \alpha = 0.5 \]

\[ \alpha = 0 \]

---

\( \frac{1 - \alpha}{2K} \)
Numerical Experiment: Comparisons of Different Algorithms

- $K = 100$, $n = 3$, and $10 \leq \kappa(A_i) \leq 10^6$
Outline

- Karcher mean computation on $S_{++}^n$
- Divergence-based means on $S_{++}^n$
- $L^1$-norm median computation on $S_{++}^n$
- Application: Structure tensor image denoising
- Conclusion
Motivations

- The mean of a set of points is sensitive to outliers
- The median is robust to outliers

**Figure:** The geometric mean and median in $\mathbb{R}^2$ space.
The Riemannian median of a set of SPD matrices is defined as:

\[
M(A_1, \ldots, A_K) = \arg \min_{X \in S_+^n} \frac{1}{2K} \sum_{i=1}^{K} \delta(A_i, X),
\]

where \(\delta(X, Y)\) is a distance or the square root of a divergence function.

- The cost function is non-smooth at \(X = A_i\)
- There may exist multiple local minima for some \(\delta\)
Numerical Experiment

Original data \((K = 50)\)

<table>
<thead>
<tr>
<th>outliers</th>
<th>0</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>25</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\delta_R)</td>
<td>Median</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

- Karcher mean computation on $S^{n+}_+$
- Divergence-based means on $S^{n+}_+$
- Riemannian $L^1$-norm median computation on $S^{n+}_+$
- Application: Structure tensor image denoising
- Conclusion
A structure tensor image is a spatial structured matrix field

\[ \mathcal{I} : \Omega \subset \mathbb{Z}^2 \rightarrow S^n_{++} \]

Noisy tensor images are simulated by replacing the pixel values by an outlier tensor with a given probability \( Pr \)

Denoising is done by averaging matrices in the neighborhood of each pixel

Mean Riemannian Error:

\[ MRE = \frac{1}{\#\Omega} \sum_{(i,j) \in \Omega} \delta_R(\mathcal{I}_{i,j}, \tilde{\mathcal{I}}_{i,j}) \]
Structure Tensor Image Denoising: $Pr = 0.02$

- (a) Original image
- (b) A-mean
- (c) K-mean
- (d) R-median
- (e) $\alpha$-mean

(f) Noisy image $Pr = 0.02$
Structure Tensor Image Denoising: MRE and Time

- **MRE comparison**

![Graph showing MRE comparison](image)

- **Time comparison**

![Graph showing time comparison](image)
Structure Tensor Image Denoising: $Pr = 0.1$

(g) Original image
(h) A-mean
(i) K-mean
(j) R-median
(k) $\alpha$-mean

(l) Noisy image $Pr = 0.1$
Structure Tensor Image Denoising: MRE and Time

**MRE comparison**

- Means
  - A-m
  - LE-m
  - J-m
  - K-m
  - R-median
  - $\alpha$-m
  - $\alpha$-median

- MRE
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 3.959
  - 0.947
  - 2.071
  - 0.948
  - 0.078
  - 0.407
  - 0.055

- Pr = 0.1

**Time comparison**

- Means
  - A-m
  - LE-m
  - J-m
  - K-m
  - R-median
  - $\alpha$-m
  - $\alpha$-median

- Time (s)
  - 0
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12
  - 14
  - 16
  - 0.04
  - 0.40
  - 0.31
  - 6.47
  - 14.31
  - 4.66
  - 11.32

- Pr = 0.1

Means of SPD Matrices
Summary

- Karcher mean for SPD matrices
  - Analyze the conditioning of the Hessian of the Karcher mean cost function
  - Apply a limited-memory Riemannian BFGS method to computing the SPD Karcher mean with efficient implementations
  - Recommend using LRBFGS as the default method for the SPD Karcher mean computation

- Other averaging techniques for SPD matrices
  - Investigate divergence-based means and Riemannian $L^1$-norm medians on $S^n_{++}$
  - Use recent development in Riemannian optimization to develop efficient and robust algorithm on $S^n_{++}$

- Evaluate the performance of different averaging techniques in applications
Thank you!
Ognjen Arandjelovic, Gregory Shakhnarovich, John Fisher, Roberto Cipolla, and Trevor Darrell.  

D. A. Bini and B. Iannazzo.  

Guang Cheng, Hesamoddin Salehian, and Baba Vemuri.  


Bruno Iannazzo and Margherita Porcelli. 
The Riemannian Barzilai-Borwein method with nonmonotone line-search and the Karcher mean computation. 
*Optimization online, December, 2015.*

B. Jeuris, R. Vandebril, and B. Vandereycken. 
A survey and comparison of contemporary algorithms for computing the matrix geometric mean. 

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

J. Lawson and Y. Lim. 
Monotonic properties of the least squares mean. 
Refs IV

Jiwen Lu, Gang Wang, and Pierre Moulin.
Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning.

Algorithm comparison for Karcher mean computation of rotation matrices and diffusion tensors.

Q. Rentmeesters.
*Algorithms for data fitting on some common homogeneous spaces.*
References V


