Riemannian Optimization and Averaging Symmetric Positive Definite Matrices

Wen Huang[1]
with Xinru Yuan[2], Kyle A. Gallivan[2], and Pierre-Antoine Absil[3]

[1]Xiamen University, [2]Florida State University [3]Université Catholique de Louvain

October 19, 2018
Beijing University

Outline

- Karcher mean computation on $\mathcal{S}_{++}^{n}$
- Divergence-based means on $\mathcal{S}_{++}^{n}$
- Riemannian L^1 median computation on $\mathcal{S}_{++}^{n}$
- Riemannian L^∞ minimax center computation on $\mathcal{S}_{++}^{n}$
- Applications
- Conclusions

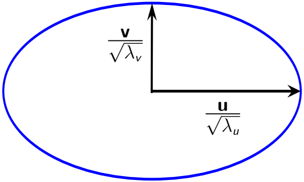
Symmetric Positive Definite (SPD) Matrix

Definition

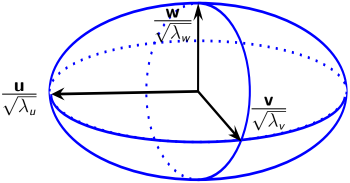
A symmetric matrix A is called positive definite $A \succ 0$ iff all its eigenvalues are positive.

$$\mathcal{S}^{n}_{++} = \{A \in \mathbb{R}^{n \times n} : A = A^T, A \succ 0\}$$

2×2 SPD matrix

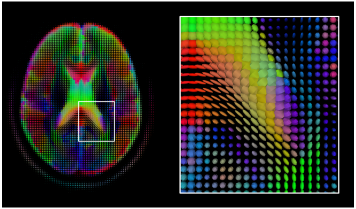


3×3 SPD matrix



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
  - denoising / interpolation
  - clustering / classification



Averaging Schemes: from Scalars to Matrices

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

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

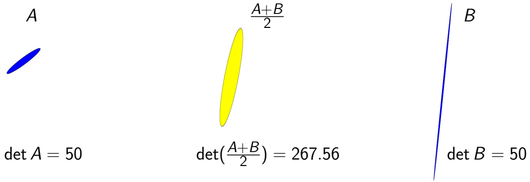
  $\rightarrow$ Not appropriate in many practical applications

Averaging Schemes: from Scalars to Matrices

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

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

  $\rightarrow$ Not appropriate in many practical applications

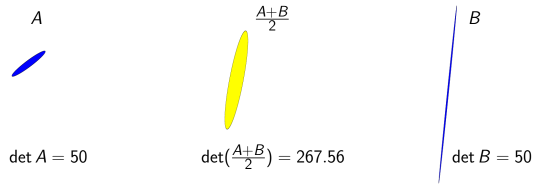


Averaging Schemes: from Scalars to Matrices

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

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

  $\rightarrow$ Not appropriate in many practical applications



- Generalized geometric mean: $(A_1 \cdots A_K)^{1/K}$

  $\rightarrow$ Not appropriate due to non-commutativity

  $\rightarrow$ How to define a matrix geometric mean?

Desired Properties of a Matrix Geometric Mean

The desired properties are given in the ALM list[1], some of which are:

- $G(A_{\pi(1)}, \dots, A_{\pi(K)}) = G(A_1, \dots, A_K)$ with π a permutation of $(1, \dots, K)$
- if $A_1, \dots, A_K$ commute, then $G(A_1, \dots, A_K) = (A_1, \dots, A_K)^{1/K}$
- $G(A_1, \dots, A_K)^{-1} = G(A_1^{-1}, \dots, A_K^{-1})$
- $\det(G(A_1, \dots, A_K)) = (\det(A_1) \cdots \det(A_K))^{1/K}$

ALM list

[1] T. Ando, C.-K. Li, and R. Mathias, *Geometric means*, Linear Algebra and Its Applications, 385:305-334, 2004

Geometric Mean of SPD Matrices

- A well-known mean on the manifold of SPD matrices is the **Karcher mean** [Kar77]:

$$G(A_1, \ldots, A_K) = \underset{X \in \mathcal{S}_{++}^n}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{1}$$

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

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

- The Karcher mean defined in (1) satisfies all the geometric properties in the ALM list [LL11]

Geometric Mean of SPD Matrices: Uniqueness

Theorem ([Afs11, AN13, FVJ09, ATV13])

Let $\mathcal{M}$ be a complete manifold with injectivity radius inj$\mathcal{M}$ and Δ be the upper bound of the sectional curvatures. Given a set of points $\{q_1, \dots, q_K\} \subset B(X_0, \rho) \subset \mathcal{M}$, the Riemannian L^p center of mass is defined as the minimizer of

$$\mu^p = \arg\min_{X \in \mathcal{M}} \sum_{i=1}^{K} \delta^p(q_i, X), \text{ with } 1 \leq p \leq \infty.$$

Then μ^p is unique in the open ball $B(X_0, \rho)$ if

$$\rho < \begin{cases} \frac{1}{2}\min\{inj\mathcal{M}, \frac{\pi}{2\sqrt{\Delta}}\}, & \text{for } 1 \leq p < 2 \\ \frac{1}{2}\min\{inj\mathcal{M}, \frac{\pi}{\sqrt{\Delta}}\}, & \text{for } 2 \leq p \leq \infty \end{cases}$$

- For $\mathcal{S}_{++}^{n}$, the Riemannian L^p center of mass is unique if all the data points are positive definite, since
  - The sectional curvature is negative
  - $\text{inj}\,\mathcal{S}_{++}^{n} = \infty$

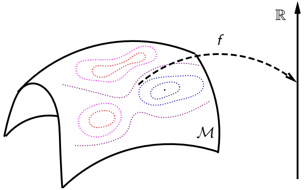
Optimization on Manifolds

$$G(A_1, \dots, A_k) = \underset{X \in \mathcal{S}^n_{++}}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i)$$

Problem: Given $f(x) : \mathcal{M} \to \mathbb{R}$,
solve

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

where $\mathcal{M}$ is a Riemannian manifold.



Unconstrained optimization problem on a constrained space.

Iterations on the Manifold

Consider the following generic update for an iterative Euclidean optimization algorithm:

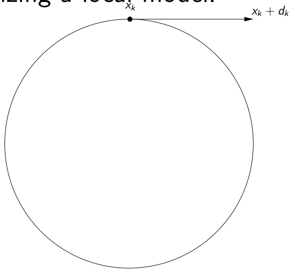
$$x_{k+1} = x_k + \Delta x_k = x_k + \alpha_k s_k \ .$$

This iteration is implemented in numerous ways, e.g.:

- Steepest descent: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$
- Newton's method: $x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$
- Trust region method: Δx_k is set by optimizing a local model.

Riemannian Manifolds Provide

- Riemannian concepts describing directions and movement on the manifold
- Riemannian analogues for gradient and Hessian



Retractions

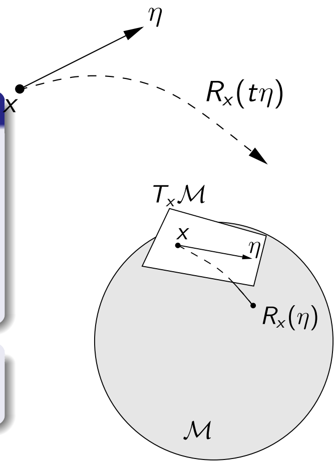
Euclidean	Riemannian
$x_{k+1} = x_k + \alpha_k d_k$	$x_{k+1} = R_{x_k}(\alpha_k \eta_k)$

Definition

A retraction is a mapping R from TM to M satisfying the following:

- R is continuously differentiable
- $R_x(0) = x$
- $\mathrm{D}\, R_x(0)[\eta] = \eta$

- maps tangent vectors back to the manifold
- defines curves in a direction



Categories of Riemannian optimization methods

Retraction-based: local information only

Line search-based: use local tangent vector and $R_x(t\eta)$ to define line

- Steepest decent
- Newton

Local model-based: series of flat space problems

- Riemannian trust region Newton (RTR)
- Riemannian adaptive cubic overestimation (RACO)

Categories of Riemannian optimization methods

Retraction and transport-based: information from multiple tangent spaces

- Nonlinear conjugate gradient: multiple tangent vectors
- Quasi-Newton e.g. Riemannian BFGS: transport operators between tangent spaces

Additional element required for optimizing a cost function (M, g):

- formulas for combining information from multiple tangent spaces.

Vector Transports

Vector Transport

- Vector transport: Transport a tangent vector from one tangent space to another
- $\mathcal{T}_{\eta_x}\xi_x$, denotes transport of ξ_x to tangent space of $R_x(\eta_x)$. R is a retraction associated with $\mathcal{T}$

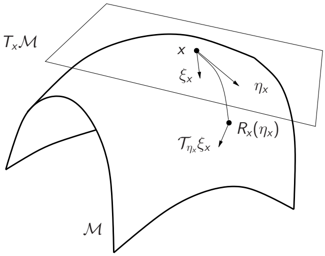


Figure: Vector transport.

Algorithms

$$G(A_1, \dots, A_k) = \arg\min_{X \in \mathcal{S}_{++}^n} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i)$$

- Riemannian steepest descent [RA11] for Karcher mean
- Riemannian steepest descent, conjugate gradient, BFGS, and trust region Newton methods [JVV12] for general problems applied to Karch mean
- Richardson-like iteration [BI13] for Karcher mean
- Riemannian Barzilai-Borwein method with nonmonotone line search and the Karcher mean computation [IP17]

Jeuris et al. [JVV12] Results

$$G(A_1, \ldots, A_k) = \underset{X \in \mathcal{S}_{++}^n}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i)$$

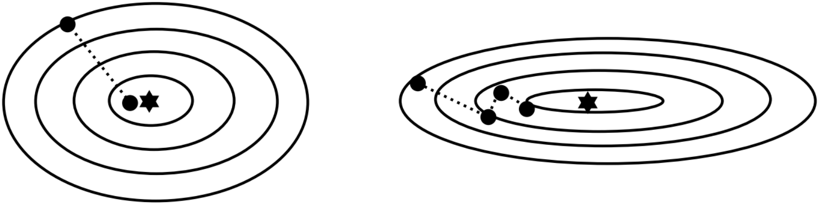
- Considered RTR-Newton-CG, RBFGS, RSD, RCG.
- First two considerably more complex per step than last two.
- RSD and RCG preferred.
- Higher rate of convergence for RBFGS (superlinear) and RTR-Newton-CG (quadratic) did not make up for extra complexity.
- Simpler first order methods recommended over a wide range of problems.

Recent results on SPD Karcher mean computation

Based on Xinru Yuan, Wen Huang, Pierre-Antoine Absil, & Kyle A. Gallivan, *A Riemannian quasi-Newton method for computing the Karcher mean of symmetric positive definite matrices* , 2018.

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**:

$$g_X(\xi, \eta) = \mathrm{trace}(\xi X^{-1} \eta X^{-1})$$

Condition number κ of Hessian at the minimizer μ:

- Hessian of Riemannian metric:
  - $\kappa(H^R) \leq 1 + \frac{\ln(\max \kappa_i)}{2}$, where $\kappa_i = \kappa(\mu^{-1/2} A_i \mu^{-1/2})$
  - $\kappa(H^R) \leq 20$ if $\max(\kappa_i) = 10^{16}$

Conditioning of the Karcher Mean Objective Function

- **Riemannian metric**:

$$g_X(\xi, \eta) = \mathrm{trace}(\xi X^{-1} \eta X^{-1})$$

- **Euclidean metric**:

$$g_X(\xi, \eta) = \mathrm{trace}(\xi \eta)$$

Condition number κ of Hessian at the minimizer μ:

- Hessian of Riemannian metric:
  - $\kappa(H^R) \leq 1 + \frac{\ln(\max \kappa_i)}{2}$, where $\kappa_i = \kappa(\mu^{-1/2} A_i \mu^{-1/2})$
  - $\kappa(H^R) \leq 20$ if $\max(\kappa_i) = 10^{16}$
- Hessian of Euclidean metric:
  - $\frac{\kappa^2(\mu)}{\kappa(H^{\mathrm{R}})} \leq \kappa(H^{\mathrm{E}}) \leq \kappa(H^{\mathrm{R}})\kappa^2(\mu)$
  - $\kappa(H^E) \geq \kappa^2(\mu)/20$

BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

- Update formula:

$$x_{k+1} = \underline{x_k + \alpha_k \eta_k}$$

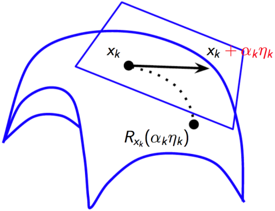
- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- 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 $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$



Optimization on a Manifold

BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

replace by $R_{x_k}(\eta_k)$ Retraction

- Update formula:

$$x_{k+1} = x_k \underline{+ \alpha_k \eta_k}$$

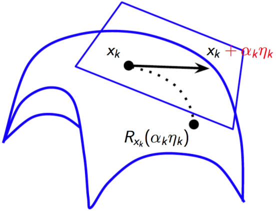
- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- 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 $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$



Optimization on a Manifold

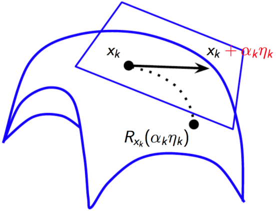
BFGS Quasi-Newton Algorithm: from Euclidean to Riemannian

replace by $R_{x_k}(\eta_k)$ Retraction

$\downarrow$

- Update formula:

$$x_{k+1} = \underline{x_k + \alpha_k \eta_k}$$



Optimization on a Manifold

- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- 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 $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$

$\uparrow$ $\uparrow$

replaced by $R_{x_k}^{-1}(x_{k+1})$

on different tangent spaces Vector Trans.

Riemannian BFGS (RBFGS) Algorithm

- Update formula:

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k) \text{ with } \eta_k = -\mathcal{B}_k^{-1} \operatorname{grad} f(x_k)$$

- $\mathcal{B}_k$ update [HGA15]:

$$\mathcal{B}_{k+1} = \tilde{\mathcal{B}}_k - \frac{\tilde{\mathcal{B}}_k s_k (\tilde{\mathcal{B}}_k s_k)^\flat}{(\tilde{\mathcal{B}}_k s_k)^\flat s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k},$$

  where $s_k = \mathcal{T}_{\alpha_k \eta_k} \alpha_k \eta_k$, $y_k = \beta_k^{-1} \operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k} \operatorname{grad} f(x_k)$, and $\tilde{\mathcal{B}}_k = \mathcal{T}_{\alpha_k \eta_k} \circ \mathcal{B}_k \circ \mathcal{T}_{\alpha_k \eta_k}^{-1}$.

- Stores and transports $\mathcal{B}_k^{-1}$ as a dense matrix

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

Limited-memory RBFGS (LRBFGS)

Riemannian BFGS:

- Let $\mathcal{H}_{k+1} = \mathcal{B}_{k+1}^{-1}$
- $\mathcal{H}_{k+1} = (\mathrm{id} - \rho_k y_k s_k^\flat)\tilde{\mathcal{H}}_k(\mathrm{id} - \rho_k y_k s_k^\flat) + \rho_k s_k s_k^\flat$
  where $s_k = \mathcal{T}_{\alpha_k \eta_k} \alpha_k \eta_k$, $y_k = \beta_k^{-1} \operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k} \operatorname{grad} f(x_k)$,
  $\rho_k = 1/g(y_k, s_k)$ and $\tilde{\mathcal{H}}_k = \mathcal{T}_{\alpha_k \eta_k} \circ \mathcal{H}_k \circ \mathcal{T}_{\alpha_k \eta_k}^{-1}$

Limited-memory Riemannian BFGS:

- Stores only the m most recent s_k and y_k
- Transports these vectors to the new tangent space rather than $\mathcal{H}_k$
- Computational and storage complexity depends upon m

Implementations

- Representations of tangent vectors
- Retraction
- Vector transport

Implementations

- Representations of tangent vectors: $T_X \mathcal{S}^{\mathrm{n}}_{++} = \{S \in \mathbb{R}^{n \times n} | S = S^T\}$
  - Extrinsic representation: n^2-dimensional vector
  - Intrinsic representation: d-dimensional vector where $d = n(n+1)/2$ Detail
- Retraction
- Vector transport

Implementations

- Representations of tangent vectors: $T_X \mathcal{S}_{++}^{n} = \{S \in \mathbb{R}^{n\times n} | S = S^T\}$
    - Extrinsic representation: n^2-dimensional vector
    - Intrinsic representation: d-dimensional vector where $d = n(n+1)/2$ Detail
- Retraction
    - Exponential mapping: $\mathrm{Exp}_X(\xi) = X^{1/2} \exp(X^{-1/2} \xi X^{-1/2}) X^{1/2}$
- Vector transport
    - Parallel translation: $\mathcal{T}_{p_\eta}(\xi) = Q \xi Q^T$, with $Q = X^{\frac{1}{2}} \exp(\frac{X^{-\frac{1}{2}} \eta X^{-\frac{1}{2}}}{2}) X^{-\frac{1}{2}}$

Implementations

- Representations of tangent vectors: $T_X \mathcal{S}^{\mathrm{n}}_{++} = \{S \in \mathbb{R}^{n\times n} | S = S^T\}$
  - Extrinsic representation: n^2-dimensional vector
  - Intrinsic representation: d-dimensional vector where $d = n(n+1)/2$ Detail
- Retraction
  - Exponential mapping: $\mathrm{Exp}_X(\xi) = X^{1/2} \exp(X^{-1/2} \xi X^{-1/2}) X^{1/2}$
  - Second order approximation retraction [JVV12]:
    $$R_X(\xi) = X + \xi + \frac{1}{2}\xi X^{-1} \xi$$
- Vector transport
  - Parallel translation: $\mathcal{T}_{p_\eta}(\xi) = Q \xi Q^T$, with $Q = X^{\frac{1}{2}} \exp(\frac{X^{-\frac{1}{2}} \eta X^{-\frac{1}{2}}}{2}) X^{-\frac{1}{2}}$
  - Vector transport by parallelization [HAG15]: essentially an identity
    Parallelization

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient

Intrinsic approach:

- Function
- Riemannian gradient

Both approaches have the same complexities: $f + \nabla f$ cost

Complexity Comparison for LRBFGS

Extrinsic approach:

- Function
- Riemannian gradient
- Retraction
  - Evaluate $R_X(\eta)$

Intrinsic approach:

- Function
- Riemannian gradient
- Retraction
  - Compute η from $\tilde{\eta}^d$
  - Evaluate $R_X(\eta)$

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

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:

- $f + \nabla f +$
  $27n^3 + 12mn^2 +$
  $\underline{2m \times \text{ Vector transport cost}}$

- $f + \nabla f +$
  $22n^3/3 + 4mn^2$

Problem Related Functions

- Cost function:

$$F(X) = \frac{1}{2K}\sum_{i=1}^{K} \mathrm{dist}^2(A_i, X) = \frac{1}{2K}\sum_{i=1}^{K} \| \log(A_i^{-1/2} X A_i^{-1/2}) \|_F^2$$

- Riemannian gradient:

$$\mathrm{grad}\, F(X) = \frac{1}{K}\sum_{i=1}^{K} A_i^{1/2} \log(A_i^{-1/2} X A_i^{-1/2}) A_i^{-1/2} X^{1/2}$$

- Riemannian Hessian action on tangent vector:

$$\begin{aligned}\mathrm{Hess}\, F(X)[\xi_X] = & \frac{1}{2K}\sum_{i=1}^{K} \xi_X \log(A_i^{-1} X) - \frac{1}{2K}\sum_{i=1}^{K} \log(X A_i^{-1}) \xi_X \\ & + \frac{1}{K}\sum_{i=1}^{K} X \mathrm{D}(\log)(A_i^{-1} X)[A_i^{-1} \xi_X]\end{aligned}$$

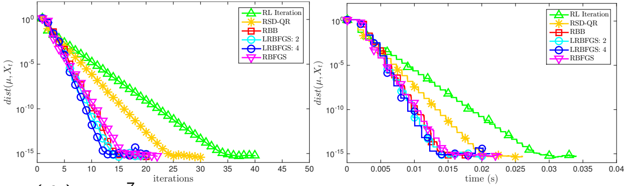
Computational Complexity

- Many eigenvalue problems must be solved for these objects.
- K eigenvalue problems for evaluation of $F(X)$.
- $F(X)$ and $\mathrm{grad}\, F(X)$ share many computations. (library must support such sharing)
- $X = LL^T$ and $A_i = L_i L_i^T$ are available and useful for representation reasons.

Numerical Results: Comparison of Different Algorithms

$K = 100$, size $= 3 \times 3$, $d = 6$

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



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

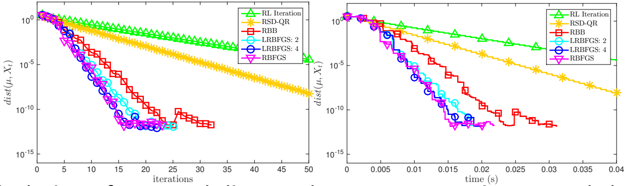


Figure: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Numerical Results: Comparison of Different Algorithms

$K = 30$, size $= 100 \times 100$, $d = 5050$

- $1 \leq \kappa(A_i) \leq 20$

- $10^4 \leq \kappa(A_i) \leq 10^7$

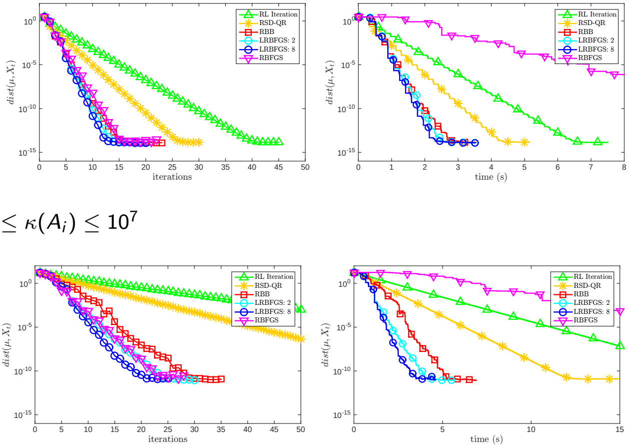
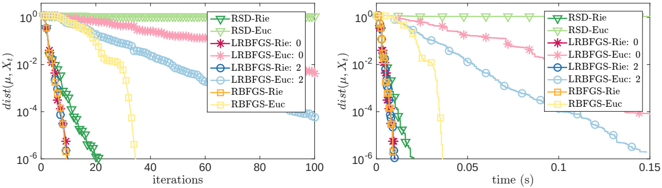


Figure: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Numerical Results: Riemannian vs. Euclidean Metrics

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



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

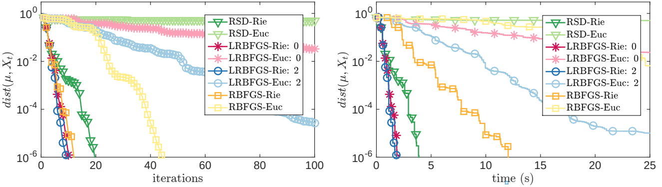


Figure: Evolution of averaged distance between current iterate and the exact Karcher mean with respect to time and iterations

Summary

- Identify the conditioning of the Hessian of the Karcher cost function as explanation for previous observations
- Apply a Riemannian version of limited-memory BFGS method to computing the SPD Karcher mean
- Present efficient implementations
- Provide theoretical and empirical suggestions on how to choose between various geometric objects and methods
- Recommend using LRBFGS as the default method for the SPD Karcher mean computation

Outline

- Karcher mean computation on $\mathcal{S}^{\mathsf{n}}_{++}$
- Divergence-based means on $\mathcal{S}^{\mathsf{n}}_{++}$
- Riemannian L^1 median computation on $\mathcal{S}^{\mathsf{n}}_{++}$
- Riemannian L^∞ minimax center computation on $\mathcal{S}^{\mathsf{n}}_{++}$
- Applications
- Conclusions

Motivations

- Karcher mean

$$K(A_1, \ldots, A_K) = \underset{X \in \mathcal{S}^n_{++}}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{1}$$

  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

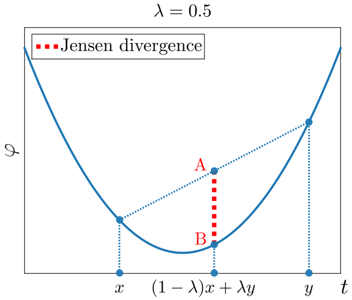
- A divergence is like a distance except it lacks
  - triangle inequality
  - symmetry

α-divergence from Jensen Convexity Gap

- Let $\varphi : \Omega \to \mathbb{R}$ be a differentiable, strictly convex function defined on a closed convex set $\Omega \subset \mathbb{R}^m$

- The Jensen divergence generated by φ with parameter λ:

$$\delta^2_{\varphi,\lambda}(x,y) = \frac{1}{\lambda(1-\lambda)}[(1-\lambda)\varphi(x) + \lambda\varphi(y) - \varphi((1-\lambda)x + \lambda y)]$$



- The α-divergence is obtained with the change of parameter $\lambda = \frac{1+\alpha}{2}$

α-divergence on $\mathcal{S}^n_{++}$

- Let $\phi : \mathcal{S}^n_{++} \to \mathbb{R}$ be a differentiable, strictly convex function

- The α-divergence $\delta^2_{\phi,\alpha} : \mathcal{S}^n_{++} \times \mathcal{S}^n_{++} \to \mathbb{R}$ is defined as

$$\delta^2_{\phi,\alpha}(X, Y) = \frac{4}{1-\alpha^2}[\frac{1-\alpha}{2}\phi(X) + \frac{1+\alpha}{2}\phi(Y) - \phi(\frac{1-\alpha}{2}X + \frac{1+\alpha}{2}Y)]$$

  with $\alpha \in (-1, 1)$

- Commonly used convex functions on $\mathcal{S}^n_{++}$:

  1. quadratic function: $\phi(X) = \mathrm{tr}(X^T X)$
  2. von Neumann function: $\phi(X) = \mathrm{tr}(X \log X - X)$
  3. log-determinant function: $\phi(X) = -\log\det X$

α-divergence on $\mathcal{S}^n_{++}$

- Let $\phi : \mathcal{S}^n_{++} \to \mathbb{R}$ be a differentiable, strictly convex function
- The α-divergence $\delta^2_{\phi,\alpha} : \mathcal{S}^n_{++} \times \mathcal{S}^n_{++} \to \mathbb{R}$ is defined as

$$\delta^2_{\phi,\alpha}(X, Y) = \frac{4}{1-\alpha^2}[\frac{1-\alpha}{2}\phi(X) + \frac{1+\alpha}{2}\phi(Y) - \phi(\frac{1-\alpha}{2}X + \frac{1+\alpha}{2}Y)]$$

with $\alpha \in (-1, 1)$

- Commonly used convex functions on $\mathcal{S}^n_{++}$:
  1. quadratic function: $\phi(X) = \mathrm{tr}(X^T X)$
  2. von Neumann function: $\phi(X) = \mathrm{tr}(X \log X - X)$
  3. log-determinant function: $\phi(X) = -\log\det X$

LogDet α-divergence and Associated Mean

- The LogDet α-divergence on $\mathcal{S}_{++}^{n}$ is given by

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

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

- The right mean based on the LogDet α-divergence is defined as

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

- The left mean and the symmetric mean are defined in a similar way.

Karcher Mean vs. LogDet α-divergence Mean

- Complexity comparison for problem-related operations

	function	gradient	total
LD α-div. mean	$\frac{2Kn^3}{3}$	$3Kn^3$	$\frac{11Kn^3}{3}$
Karcher mean	$18Kn^3$	$5Kn^3$	$23Kn^3$

- Invariance properties

	scaling invariance	rotation invariance	congruence invariance	inversion invariance
LD α-div. mean	✓	✓	✓	✗
Karcher mean	✓	✓	✓	✓

LogDet α-divergence Mean

Theorem

The function $\delta^2_{\mathrm{LD},\alpha}(X, Y)$ with $\alpha \in (-1, 1)$ is jointly geodesically convex.

LogDet α-divergence Mean

Theorem

The function $\delta^2_{\mathrm{LD},\alpha}(X, Y)$ with $\alpha \in (-1, 1)$ is jointly geodesically convex.

- Any local minimum is global
- Chebbi and Moakher proposed a fixed-point iteration to compute the mean in [CM12]
- We apply our Riemannian optimization techniques to solve this problem, which outperforms the state-of-the-art method
- We cast Chebbi and Moakher's iteration into Riemannian optimization framework

LogDet α-divergence Mean: CM's Fixed-point Iteration

Chebbi and Moakher's fixed-point iteration [CM12] can be rewritten as

$$Y_{k+1} = \frac{1}{K}\sum_{i=1}^{K}\left(\frac{1-\alpha}{2}A_i + \frac{1+\alpha}{2}Y_k^{-1}\right)^{-1} \tag{1}$$

$$\boxed{= Y_k - \frac{1-\alpha}{2K}\ \mathrm{grad}\, f(Y_k)} \tag{2}$$

where $\mathrm{grad}\, f(Y)$ denotes the Riemannian gradient of $f(Y)$ and

$$f(Y) = \frac{4}{1-\alpha^2}\sum_{i=1}^{K}\left\{\log\det\left(\frac{1-\alpha}{2}A_i + \frac{1+\alpha}{2}Y^{-1}\right) + \frac{1+\alpha}{2}\log\det Y\right\}$$

LogDet α-divergence Mean: CM's Fixed-point Iteration

Chebbi and Moakher's fixed-point iteration [CM12] can be rewritten as

$$Y_{k+1} = \frac{1}{K}\sum_{i=1}^{K}(\frac{1-\alpha}{2}A_i + \frac{1+\alpha}{2}Y_k^{-1})^{-1} \tag{1}$$

$$= Y_k - \frac{1-\alpha}{2K} \operatorname{grad} f(Y_k) \tag{2}$$

where $\operatorname{grad} f(Y)$ denotes the Riemannian gradient of $f(Y)$ and

$$f(Y) = \frac{4}{1-\alpha^2}\sum_{i=1}^{K}\{\log\det(\frac{1-\alpha}{2}A_i + \frac{1+\alpha}{2}Y^{-1}) + \frac{1+\alpha}{2}\log\det Y\}$$

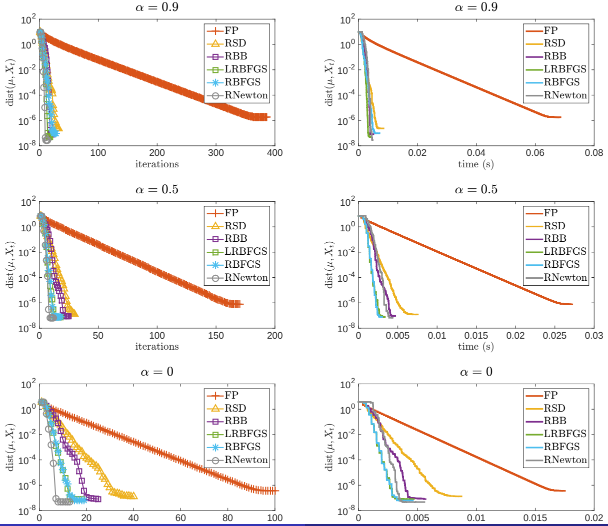
The fixed-point iteration is a Riemannian steepest descent using

- a constant stepsize $(1-\alpha)/2K$
- Euclidean retraction $R_X(\eta_X) = X + \eta_X$

Numerical Experiment I: Comparions of Different Algorithms

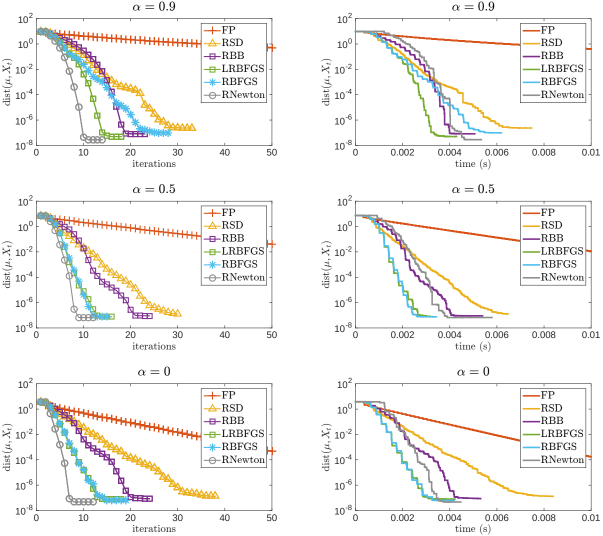
- $K = 100$, $n = 3$, and $10 \leq \kappa(A_i) \leq 10^6$

FP stepsize $= \dfrac{1-\alpha}{2K}$



Numerical Experiment I: Comparions of Different Algorithms

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



Outline

- Karcher mean computation on $\mathcal{S}^n_{++}$
- Divergence-based means on $\mathcal{S}^n_{++}$
- Riemannian L^1 median computation on $\mathcal{S}^n_{++}$
- Riemannian L^∞ minimax center computation on $\mathcal{S}^n_{++}$
- Applications
- Conclusions

Motivations

- The mean is sensitive to outliers
- The median is less sensitive to outliers

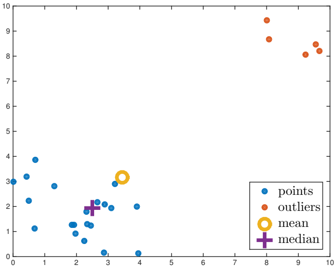


Figure: The geometric mean and median in $\mathbb{R}^2$ space.

Riemannian Median of SPD Matrices

The Riemannian median of a set of SPD matrices is defined as

$$M(A_1, \ldots, A_K) = \underset{X \in \mathcal{S}^{\mathrm{n}}_{++}}{\arg\min} \frac{1}{2K} \sum_{i=1}^{K} \delta(A_i, X) \; ,$$

where δ is a distance or the square root of a divergence function

- The cost function is nonsmooth at $X = A_i$
- If δ is the geodesic distance, the median is unique

Algorithms

$$M(A_1, \ldots, A_K) = \operatorname*{arg\,min}_{X \in \mathcal{S}_{++}^{n}} \frac{1}{2K} \sum_{i=1}^{K} \delta(A_i, X)$$

- Riemannian Weiszfeld's algorithm [FVJ09]
- Our approach: Riemannian quasi-Newton algorithms
  - Smooth RBFGS [HAG18]
  - Modified RBFGS [Hua12]
  - Nonsmooth RBFGS [HHY18]
  - Limited-memory versions of the above three [HAGH16]

RBFGS for Partly Smooth Functions

Modified RBFGS [Hua12]

- Uses the same search direction as smooth RBFGS
- Modify the line search
- Modify the stopping criterion
  - $G_k = \{\mathcal{T}_{y_j^{(k)} \to x_k}(\mathrm{grad}\, f(y_j^{(k)})) : y_j^{(k)} \in \mathrm{cl} B(x_k, \tau)\}$
  - $d_k = \arg\min\{\|d\| : d \in \mathrm{conv}\, G_k\}$
  - Stop when $\|d_k\| \leq \epsilon$

Nonsmooth RBFGS [HHY18]

- Uses a different search direction
  - $W_k = \{\mathcal{T}_{y_j^{(k)} \to x_k} \partial f(y_j^{(k)}) : y_j^{(k)} \in \mathrm{cl} B(x_k, \tau)\}$
  - $g_k = \arg\min_{v \in \mathrm{conv} W_k} \|v\|_{\mathcal{H}_k}$
  - $\eta_k = -\mathcal{H}_k g_k$

Numerical Results for L^1 Median Computation on $\mathcal{S}_{++}^n$: Comparison of Different Algorithms

- $K = 100$, size $= 3 \times 3$
- well-conditioned A_i

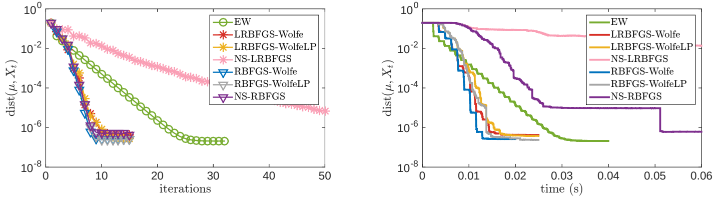


Figure: Evolution of averaged distance between current iterate and the exact Riemannian median with respect to time and iterations.

Numerical Results for L^1 Median Computation on $\mathcal{S}^n_{++}$: Comparison of Different Algorithms

- $K = 100$, size $= 3 \times 3$
- well-conditioned A_i + 5% ill-conditioned outliers

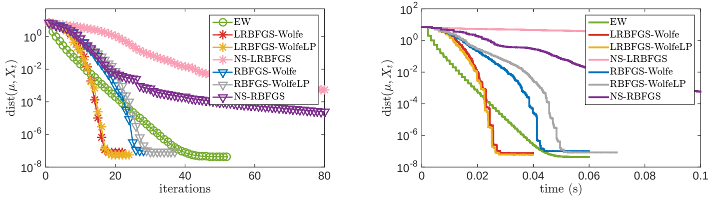


Figure: Evolution of averaged distance between current iterate and the exact Riemannian median with respect to time and iterations.

Numerical Results for L^1 Median Computation on $\mathcal{S}^n_{++}$: Comparison of Different Algorithms

- $K = 100$, size $= 100 \times 100$
- well-conditioned A_i

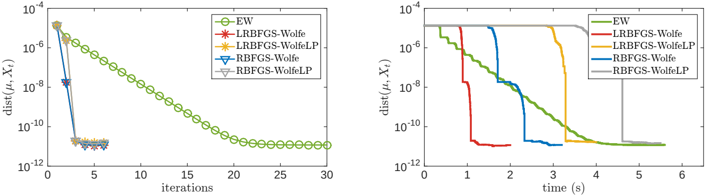


Figure: Evolution of averaged distance between current iterate and the exact Riemannian median with respect to time and iterations.

Outline

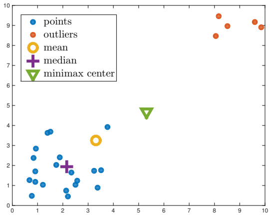
- Karcher mean computation on $\mathcal{S}^{\mathrm{n}}_{++}$
- Divergence-based means on $\mathcal{S}^{\mathrm{n}}_{++}$
- Riemannian L^1 median computation on $\mathcal{S}^{\mathrm{n}}_{++}$
- Riemannian L^∞ minimax center computation on $\mathcal{S}^{\mathrm{n}}_{++}$
- Applications
- Conclusions

Riemannian L^∞ minimax center on $\mathcal{S}^n_{++}$

The minimax center of a set of SPD matrices is defined as

$$c^\infty(A_1, \ldots, A_K) = \underset{X \in \mathcal{S}^n_{++}}{\arg\min} \max_{1 \le i \le K} \delta(X, A_i)$$

- Also called the minimum/smallest enclosing ball center

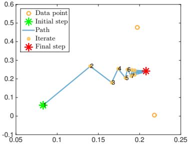


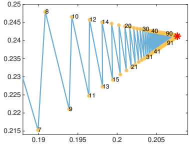
- The cost function is nonsmooth when there is a tie

Algorithms

$$c^{\infty}(A_1, \ldots, A_K) = \arg\min_{X \in \mathcal{S}_{++}^n} \max_{1 \le i \le K} \delta(X, A_i)$$

- Classical Arnaudon and Nielsen's (AN) algorithm [AN13]





- Riemannian gradient sampling algorithm (RGS) [Hua12]

- Our approach: Riemannian quasi-Newton algorithms
  - Modified RBFGS and LRBFGS
  - Nonsmooth RBFGS and LRBFGS

Numerical Results for L^∞ Minimax Center Computation on $\mathcal{S}^n_{++}$: Comparison of Different Algorithms

- $K = 100$, size $= 3 \times 3$
- A_is are separated into 4 clusters

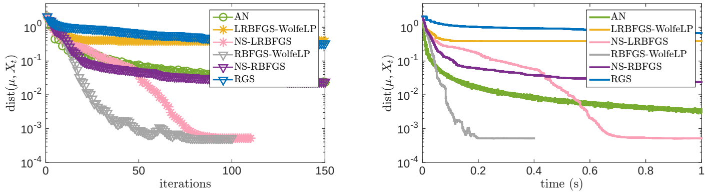
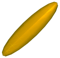


Figure: Evolution of averaged distance between current iterate and the exact minimax center with respect to time and iterations.

Comparison between Mean, Median and Minimax Center

Original data ($K = 50$)

Outliers

outliers	0	1	5	10	25	50
Mean						
Median						
Minimax center						

Outline

- Karcher mean computation on $\mathcal{S}^n_{++}$
- Divergence-based means on $\mathcal{S}^n_{++}$
- Riemannian L^1 median computation on $\mathcal{S}^n_{++}$
- Riemannian L^∞ minimax center computation on $\mathcal{S}^n_{++}$
- Applications
    - Application I: Structure tensor image denoising
    - Application II: EEG classification based on the minimum distance to mean classifier
    - Application III: Image clustering
- Conclusions

Application I: Structure Tensor Image Denoising

- A structure tensor image is a spatial structured matrix field

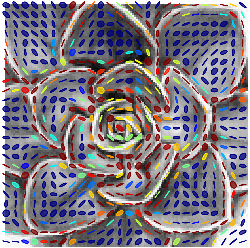
$$\mathcal{I} : \Omega \subset \mathbb{Z}^2 \to \mathcal{S}^{n}_{++}$$

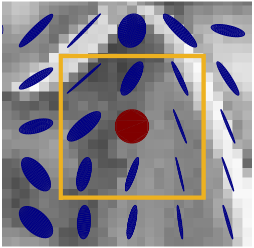
- Noisy tensor images are simulated by replacing the pixel values by an outlier tensor with a given probability Pr

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

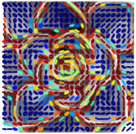
- Mean Riemannian Error:

$$MRE = \frac{1}{\#\Omega} \sum_{(i,j)\in\Omega} \delta_R(\mathcal{I}_{i,j}, \tilde{\mathcal{I}}_{i,j})$$

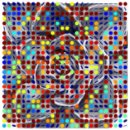




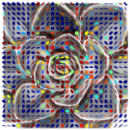
Structure Tensor Image Denoising: $Pr = 0.1$



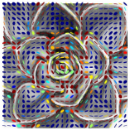
(a) Original image



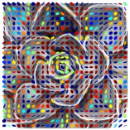
(b) E.-Mean



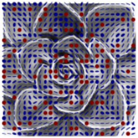
(c) R.-Mean



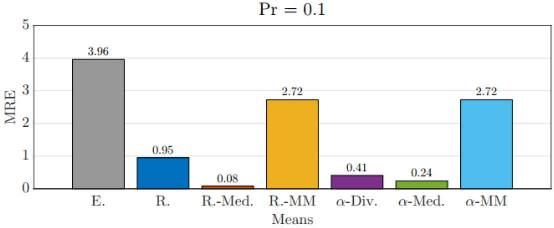
(d) R.-Med.



(e) R.-MM

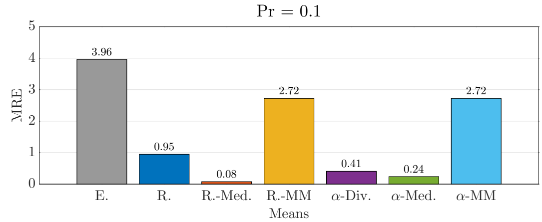


(f) Noisy image $Pr = 0.1$

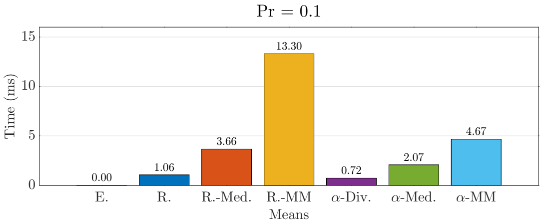


Structure Tensor Image Denoising: MRE and Time

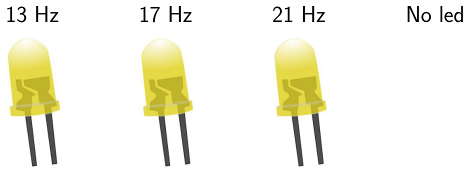
- MRE comparison



- Time comparison

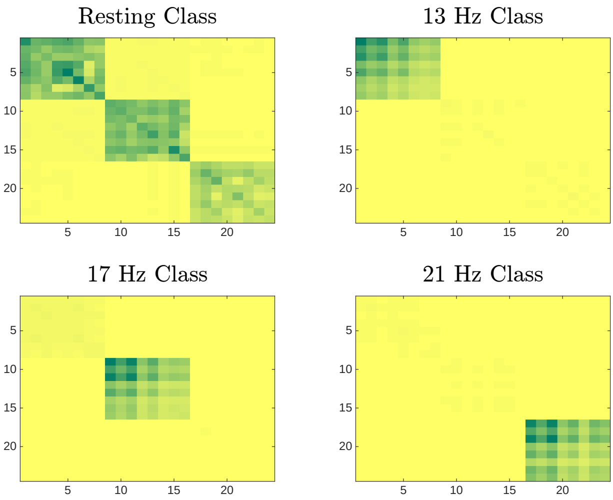


Application II: Electroencephalography (EEG) Classification



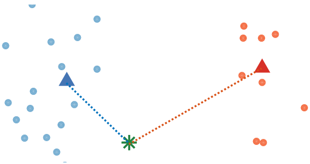
- The subject is either asked to focus on one specific blinking LED or a location without LED
- EEG system is used to record brain signals
- Covariance matrices of size 24×24 are used to represent EEG recordings [KCB+15, MC17]

EEG Classification: Examples of Covariance Matrices



EEG Classification: Minimum Distance to Mean classier

Goal: classify new covariance matrix using Minimum Distance to Mean Classifier

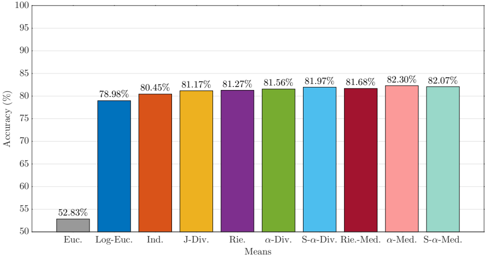


- For each class $k = 1, \dots, K$, compute the center μ_k of the covariance matrices in the training set that belong to class k
- Classify a new covariance matrix X according to

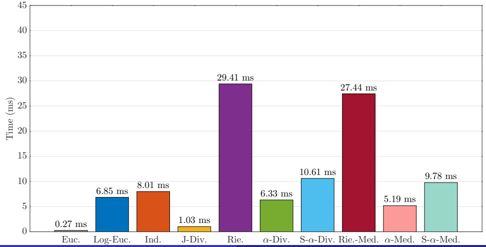
$$\hat{k} = \underset{1 \le k \le K}{\arg\min}\, \delta(X, \mu_k)$$

EEG Classfification: Accuracy and Computation Time

- Accuracy comparison

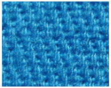
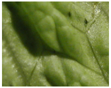


- Computation time comparison



Application III: Image Clustering Using K-means Method

- The KTH-TIPS2 dataset [MFT+06]

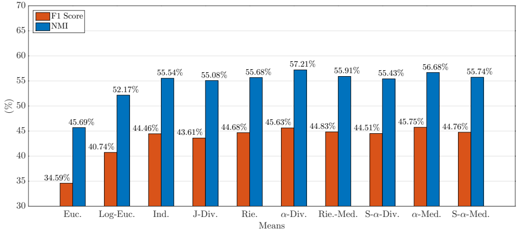
  - 4752 samples, 11 categories, 432 samples per category
  - Region Covariance Matrices: 23×23

- Performance metrics to measure the quality of K-means clustering
  - F1-Score
  - Normalized mutual information (NMI)

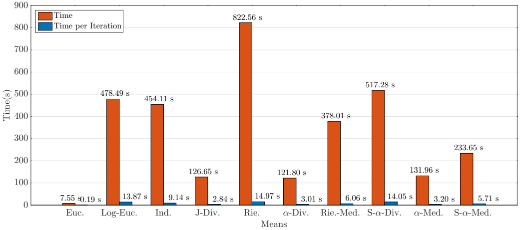
- Performance metrics to measure the timing of K-means clustering
  - Total computation time
  - Computation time per iteration

Image Clustering: Comparison of Different K-means Variants

- Quality comparison



- Timing comparison



Outline

- Karcher mean computation on $\mathcal{S}_{++}^{n}$
- Divergence-based means on $\mathcal{S}_{++}^{n}$
- Riemannian L^1 median computation on $\mathcal{S}_{++}^{n}$
- Riemannian L^∞ minimax center computation on $\mathcal{S}_{++}^{n}$
- Applications
- Conclusions

Conclusions

- Investigate different averaging techniques for SPD matrices, including the computation of means, medians, and minimax centers
- Use recent developments in Riemannian optimization to develop efficient and robust algorithms on $\mathcal{S}_{++}^{n}$
- Provide empirical assessments and comparisons of the performance of considered Riemannian optimization algorithms and existing stat-of-the-art algorithms
- Contribute a C++ toolbox for various averaging techniques (based on ROPTLIB)
- Provide user guidelines on how to choose between various methods and parameters
- Evaluate the performance of different averaging techniques in applications

Thank you!

References I

Bijan Afsari.
Riemannian L^p center of mass: existence, uniqueness, and convexity.
Proceedings of the American Mathematical Society, 139(2):655–673, 2011.

Marc Arnaudon and Frank Nielsen.
On approximating the Riemannian 1-center.
Computational Geometry, 46(1):93–104, 2013.

Ognjen Arandjelovic, Gregory Shakhnarovich, John Fisher, Roberto Cipolla, and Trevor Darrell.
Face recognition with image sets using manifold density divergence.
In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 581–588. IEEE, 2005.

References II

- B. Afsari, R. Tron, and R. Vidal.
  On the convergence of gradient descent for finding the Riemannian center of mass.
  SIAM Journal on Control and Optimization, 51(3):2230–2260, 2013.

- D. A. Bini and B. Iannazzo.
  Computing the Karcher mean of symmetric positive definite matrices.
  Linear Algebra and its Applications, 438(4):1700–1710, 2013.

- Andrzej Cichocki, Sergio Cruces, and Shun-ichi Amari.
  Log-determinant divergences revisited: Alpha-beta and gamma log-det divergences.
  Entropy, 17(5):2988–3034, 2015.

- Zeineb Chebbi and Maher Moakher.
  Means of Hermitian positive-definite matrices based on the log-determinant α-divergence function.
  Linear Algebra and its Applications, 436(7):1872–1889, 2012.

References III

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

P. T. Fletcher and S. Joshi.
Riemannian geometry for the statistical analysis of diffusion tensor data.
Signal Processing, 87(2):250–262, 2007.

P Thomas Fletcher, Suresh Venkatasubramanian, and Sarang Joshi.
The geometric median on Riemannian manifolds with application to robust atlas estimation.
NeuroImage, 45(1):S143–S152, 2009.

Wen Huang, P-A Absil, and Kyle A Gallivan.
A Riemannian symmetric rank-one trust-region method.
Mathematical Programming, 150(2):179–216, 2015.

References IV

Wen Huang, P.-A. Absil, and K. A. Gallivan.
A Riemannian BFGS Method without Differentiated Retraction for Nonconvex Optimization Problems.
SIAM Journal on Optimization, 28(1):470–495, 2018.

Wen Huang, PA Absil, KA Gallivan, and Paul Hand.
ROPTLIB: an object-oriented C++ library for optimization on Riemannian manifolds.
Technical report, Technical Report FSU16-14, Florida State University, 2016.

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

References V

Seyedehsomayeh Hosseini, Wen Huang, and Rohollah Yousefpour.
Line search algorithms for locally Lipschitz functions on Riemannian manifolds.
SIAM Journal on Optimization, 28(1):596–619, 2018.

Wen Huang.
Optimization algorithms on Riemannian manifolds with applications.
PhD thesis, Department of Mathematics, Florida State University, 2012.

Zhiwu Huang, Ruiping Wang, Shiguang Shan, and Xilin Chen.
Face recognition on large-scale video in the wild with hybrid Euclidean-and-Riemannian metric learning.
Pattern Recognition, 48(10):3113–3124, 2015.

Bruno Iannazzo and Margherita Porcelli.
The Riemannian Barzilai–Borwein method with nonmonotone line search and the matrix geometric mean computation.
IMA Journal of Numerical Analysis, 38(1):495–517, 2017.

References VI

B. Jeuris, R. Vandebril, and B. Vandereycken.
A survey and comparison of contemporary algorithms for computing the matrix geometric mean.
Electronic Transactions on Numerical Analysis, 39:379–402, 2012.

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

Emmanuel K Kalunga, Sylvain Chevallier, Quentin Barthélemy, Karim Djouani, Yskandar Hamam, and Eric Monacelli.
From Euclidean to Riemannian means: Information geometry for SSVEP classification.
In *International Conference on Networked Geometric Science of Information*, pages 595–604. Springer, 2015.

J. Lawson and Y. Lim.
Monotonic properties of the least squares mean.
Mathematische Annalen, 351(2):267–279, 2011.

References VII

Jiwen Lu, Gang Wang, and Pierre Moulin.
Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning.
In *Proceedings of the IEEE International Conference on Computer Vision*, pages 329–336, 2013.

Estelle M Massart and Sylvain Chevallier.
Inductive means and sequences applied to online classication of EEG.
Technical report, ICTEAM Institute, Universite Catholiqué de Louvain, 2017.

P Mallikarjuna, M Fritz, A Tavakoli Targhi, E Hayman, B Caputo, and JO Eklundh.
The KTH-TIPS and KTH-TIPS2 databases, 2006.

References VIII

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

Y. Rathi, A. Tannenbaum, and O. Michailovich.
Segmenting images on the tensor manifold.
In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

Gregory Shakhnarovich, John W Fisher, and Trevor Darrell.
Face recognition from long-term observations.
In *European Conference on Computer Vision*, pages 851–865. Springer, 2002.

References IX

Oncel Tuzel, Fatih Porikli, and Peter Meer.
Region covariance: A fast descriptor for detection and classification.
In *European conference on computer vision*, pages 589–600.
Springer, 2006.

S. J. Wright and J. Nocedal.
Numerical optimization.
Springer New York, 2 edition, 2006.

Intrinsic Representation of Tangent Vectors

- Tangent vector $\eta_X \in \mathrm{T}_X \mathcal{M}$ can be represented by its intrinsic representation, i.e., a d-dimensional vector of coordinates n a given basis of B_X of $\mathrm{T}_X \mathcal{M}$
- If $B_X = \{b_1, \ldots b_d\}$, $\eta_X = \alpha_1 b_1 + \cdots + \alpha_d b_d$, then $\eta_X^d = B_X^\flat \eta_X$ and $\eta_X^d = (\alpha_1, \ldots, \alpha_d)^T$
- Reduces storage of tangent vectors and simplifies certain Riemannian objects if B_X is orthonormal and the coefficients α_i's are easy to compute

Go Back $\longleftarrow$

Vector Transport by Parallelization

- Vector transport by parallelization is defined as $\mathcal{T} = B_Y B_X^\flat$, where B_Y and B_X are bases of $\mathrm{T}_Y \mathcal{M}$ and $\mathrm{T}_X \mathcal{X}$, respectively
- If B_Y and B_X are orthonormal bases of $\mathrm{T}_Y \mathcal{M}$ and $\mathrm{T}_X \mathcal{M}$, respectively, then the vector transport by parallelization is the identity
- Parallelization is an isometric vector transport

Go Back ⟵

Jointly Geodesically Convexity

Definition

Let $(\mathcal{M}, g)$ be a Riemannian manifold. A function $f : \mathcal{M} \to \mathbb{R}$ is said to be geodesically convex if for any $x, y \in \mathcal{M}$, a geodesic γ such that $\gamma_1(0) = x_1$ and $\gamma(1) = y$, and $t \in [0, 1]$, it holds that

$$f(\gamma(t)) \leq (1-t)f(x) + tf(y) \tag{2}$$

Definition

Let $(\mathcal{M}, g)$ be a Riemannian manifold. A function $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is said to be jointly geodesically convex if for any $x_1, x_2, y_1, y_2 \in \mathcal{M}$, geodesics γ_x and γ_y such that $\gamma_x(0) = x_1$, $\gamma_x(1) = x_2$, $\gamma_y(0) = y_1$ and $\gamma_y(1) = y_2$, and $t \in [0, 1]$, it holds that

$$f(\gamma_x(t), \gamma_y(t)) \leq (1-t)f(x_1, y_1) + tf(x_2, y_2). \tag{3}$$

Go Back ⟵

Divergence Symmetrization

A divergence is asymmetric in general. There are two common ways to symmetrize a divergence [CCA15]:

- Type 1:
$$\delta_{\mathrm{S}\phi}^2(X, Y) = \frac{1}{2}(\delta_\phi^2(X, Y) + \delta_\phi^2(Y, X)),$$
- Type 2:
$$\delta_{\mathrm{S}\phi}^2(X, Y) = \frac{1}{2}(\delta_\phi^2(X, \frac{X+Y}{2}) + \delta_\phi^2(Y, \frac{X+Y}{2})).$$

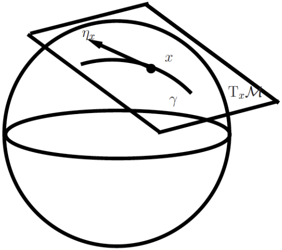
ALM List

- P1 Consistency with scalars. If $A_1, \ldots, A_K$ commute then $G(A_1, \ldots, A_K) = (A_1 \cdots A_K)^{1/K}$.
- P2 Joint homogeneity. $G(\alpha_1 A_1, \ldots, \alpha_K A_K) = (\alpha_1 \cdots \alpha_K)^{1/K} G(A_1, \ldots, A_K)$.
- P3 Permutation invariance. For any permutation $\pi(A_1, \ldots, A_K)$ of $(A_1, \ldots, A_K)$, $G(A_1, \ldots, A_K) = G(\pi(A_1, \ldots, A_K))$.
- P4 Monotonicity. If $A_i \geq B_i$ for all i, then $G(A_1, \ldots, A_K) \geq G(B_1, \ldots, B_K)$ in the positive semidefinite ordering.
- P5 Continuity from above. If $\{A_1^{(n)}\}, \ldots, \{A_k^{(n)}\}$ are monotonic decreasing sequences (in the positive semidefinite ordering) converging to $A_1, \ldots, A_K$, respectively, then $G(A_1^{(n)}, \ldots, A_K^{(n)})$ converges to $G(A_1, \ldots, A_K)$.
- P6 Congruence invariance. $G(S^T A_1 S, \ldots, S^T A_K S) = S^T G(A_1, \ldots, A_K) S$ for any invertible S.
- P7 Joint concavity. $G(\lambda A_1 + (1-\lambda) B_1, \ldots, \lambda A_K + (1-\lambda) A_K) \geq \lambda G(A_1, \ldots, A_K) + (1-\lambda) G(B_1, \ldots, B_K)$.
- P8 Invariance under inversion. $G(A_1, \ldots, A_K)^{-1} = G(A_1^{-1}, \ldots, A_K^{-1})$.
- P9 Determinant identity. $\det G(A_1, \ldots, A_K) = (\det A_1 \cdots \det A_K)^{1/K}$.

Tangent Vector

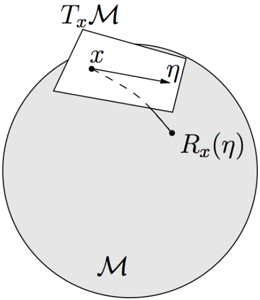
- **Definition:** The tangent space $\mathrm{T}_X\,\mathcal{M}$ is the vector space comprised of the tangent vectors at $X \in \mathcal{M}$. The Riemannian metric is an inner product on each tangent space.

- Tangent vectors can be represented by an intrinsic representation, which reduces the storage and simplifies certain Riemannian objects Intrinsic Representation



Retraction

- Maps tangent vectors back to the manifold
- **Definition:** A retraction is a mapping R from $T\mathcal{M}$ to $\mathcal{M}$ satisfying the following:
  - R is continuously differentiable
  - $R_x(0) = x$
  - $\mathrm{D}R_x(0)(\eta) = \eta$

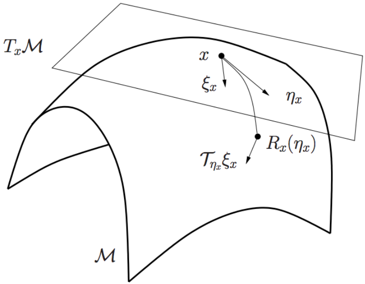


Euclidean	Riemannian
$x_{k+1} = x_k + \alpha_k \eta_k$	$x_{k+1} = R_{x_k}(\alpha_k \eta_k)$

Go Back

Vector Transport

- Some algorithms need to combine information on different tangent spaces to determine the next search direction
- Vector transport: transport a tangent vector from one tangent space to another
- $\mathcal{T}_{\eta_x}\xi_x$, denotes transport of ξ_x to tangent space of $R_x(\eta_x)$



Go Back

Stepsizes for RSD

- Classical stepsize strategy in [WN06, (3.44)]

$$\alpha_{k+1} = \min\{1, 1.01 \cdot \frac{2(f(x_{k+1}) - f(x_k))}{g(\mathrm{grad}\, f(x_{k+1}), -\mathrm{grad}\, f(x_k))}\}$$

- Different versions of BB stepsizes
  - $s_k = \mathcal{T}_{\alpha_k \eta_k}(\alpha_k \eta_k)$, $y_k = \mathrm{grad}\, f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k}(\mathrm{grad}\, f(x_k))$
  - BB1: $\alpha_{k+1} = g(s_k, s_k)/g(s_k, y_k)$
  - BB2: $\alpha_{k+1} = g(s_k, y_k)/g(y_k, y_k)$
  - $\text{ABB}_{\min}$:
    $$\alpha_{k+1} = \begin{cases} \min\{\alpha_j^{\text{BB2}} : j = \max(1, k - m_a), \dots, k\}, & \text{if } \alpha_{k+1}^{\text{BB2}}/\alpha_{k+1}^{\text{BB1}} < \tau \\ \alpha_{k+1}^{\text{BB1}}, & \text{otherwise} \end{cases}$$

Go Back

Stepsizes for RSD

- Classical stepsize strategy in [WN06, (3.44)]

$$\alpha_{k+1} = \min\{1, 1.01 \cdot \frac{2(f(x_{k+1}) - f(x_k))}{g(\mathrm{grad}\, f(x_{k+1}), -\mathrm{grad}\, f(x_k))}\}$$

- Different versions of BB stepsizes
  - $s_k = \mathcal{T}_{\alpha_k \eta_k}(\alpha_k \eta_k)$, $y_k = \mathrm{grad}\, f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k}(\mathrm{grad}\, f(x_k))$
  - BB1: $\alpha_{k+1} = g(s_k, s_k)/g(s_k, y_k)$
  - BB2: $\alpha_{k+1} = g(s_k, y_k)/g(y_k, y_k)$
  - $\mathrm{ABB}_{\mathrm{min}}$:
    $$\alpha_{k+1} = \begin{cases} \min\{\alpha_j^{\mathrm{BB2}} : j = \max(1, k - m_a), \ldots, k\}, & \text{if } \alpha_{k+1}^{\mathrm{BB2}}/\alpha_{k+1}^{\mathrm{BB1}} < \tau \\ \alpha_{k+1}^{\mathrm{BB1}}, & \text{otherwise} \end{cases}$$

Go Back

LogDet α-divergence Mean: CM's Fixed-point Iteration

Chebbi and Moakher's fixed-point iteration [CM12] can be rewritten as

$$Y_{k+1} = \frac{1}{K}\sum_{i=1}^{K}\left(\frac{1-\alpha}{2}A_i + \frac{1+\alpha}{2}Y_k^{-1}\right)^{-1} \tag{1}$$

$$= Y_k - \frac{1-\alpha}{2K}\,\mathrm{grad}\, f(Y_k) \tag{2}$$

where $\mathrm{grad}\, f(Y)$ denotes the Riemannian gradient of $f(Y)$ and

$$f(Y) = \frac{4}{1-\alpha^2}\sum_{i=1}^{K}\left\{\log\det\left(\frac{1-\alpha}{2}A_i + \frac{1+\alpha}{2}Y^{-1}\right) + \frac{1+\alpha}{2}\log\det Y\right\}$$

The fixed-point iteration is a Riemannian steepest descent using

- a constant stepsize $(1-\alpha)/2K$
- Euclidean retraction $R_X(\eta_X) = X + \eta_X$

Background

Update for steepest descent:

$$\eta_k = -\alpha_k \operatorname{grad} f(x_k)$$
$$x_{k+1} = R_{x_k}(\eta_k)$$

- RSD:
  - α_k is taken as the classical strategy in [WN06] Formula
  - no use of second order information
- RBB:
  - choose α_k so that $-\alpha_k \operatorname{grad} f(x_k)$ approximates $-\operatorname{Hess} f(x_k)^{-1} \operatorname{grad} f(x_k)$

    i.e., $\alpha_k I$ approximates $\operatorname{Hess} f(x_k)^{-1}$
  - make use of second order information BB Stepsize

Background

Update for steepest descent:

$$\eta_k = -\alpha_k \operatorname{grad} f(x_k)$$
$$x_{k+1} = R_{x_k}(\eta_k)$$

- RSD:
  - α_k is taken as the classical strategy in [WN06] Formula
  - no use of second order information
- RBB:
  - choose α_k so that $-\alpha_k \operatorname{grad} f(x_k)$ approximates $-\operatorname{Hess} f(x_k)^{-1} \operatorname{grad} f(x_k)$

    i.e., $\alpha_k I$ approximates $\operatorname{Hess} f(x_k)^{-1}$
  - make use of second order information BB Stepsize

Goal: investigate the relationship between the BB stepsizes and the eigenvalues of the Riemannian Hessian of the objective function

Numerical Experiment II: BB Stepsizes and the Hessian Eigenvalues

Goal: investigate the relationship between the BB stepsizes and the eigenvalues of the Riemannian Hessian of the objective function

- Objective function used: $f(X) = \frac{1}{2K} \sum_{i=1}^{K} \delta^2_{\mathrm{LD},\alpha}(A_i, X)$
- $\{\lambda_1^{(k)}, \ldots, \lambda_d^{(k)}\}$ are eigenvalues of the Riemannian Hessian of f
- Compare $1/\alpha_k$ and $\{\lambda_1^{(k)}, \ldots, \lambda_d^{(k)}\}$
- RBB is used with Armijo backtracking line search
- BB1, BB2, $\mathrm{ABB}_{\mathrm{min}}$ are compared BB Stepsize

Numerical Experiment III: $\alpha = 0.5$

- $K = 200$, 6×6, $d = 21$