



NTNU

Norges teknisk-naturvitenskapelige universitet

Nonsmooth Optimization on Riemannian Manifolds in Manopt.jl

Ronny Bergmann

33rd European Conference on Operational Research (Euro) 2024

Software for Optimization — Optimization Frameworks

Copenhagen,

July 3, 2024.

The Rayleigh Quotient

When minimizing the **Rayleigh quotient** for a symmetric $A \in \mathbb{R}^{n \times n}$

$$\arg \min_{x \in \mathbb{R}^n \setminus \{0\}} \frac{x^T A x}{\|x\|^2}$$

⚠ Any eigenvector x^* to the smallest EV λ is a minimizer

👎 no isolated minima **and** Newton's method diverges

💡 Constrain the problem to unit vectors $\|x\| = 1$!

classic constrained optimization (ALM, EPM,...)

Today Utilize the geometry of the sphere

⬆ unconstrained optimization $\arg \min_{p \in \mathbb{S}^{n-1}} p^T A p$


☰ adapt unconstrained optimization to **Riemannian manifolds**.


The Generalized Rayleigh Quotient

More general. Find a basis for the space of eigenvectors to $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_k$:


$$\arg \min_{X \in \text{St}(n,k)} \text{tr}(X^T A X), \quad \text{St}(n,k) := \{X \in \mathbb{R}^{n \times k} \mid X^T X = I\},$$

 a problem on the **Stiefel** manifold $\text{St}(n,k)$

 Invariant under rotations within a k -dim subspace.

 Find the best subspace!

$$\arg \min_{\text{span}(X) \in \text{Gr}(n,k)} \text{tr}(X^T A X), \quad \text{Gr}(n,k) := \{\text{span}(X) \mid X \in \text{St}(n,k)\},$$

 a problem on the **Grassmann** manifold $\text{Gr}(n,k) = \text{St}(n,k)/O(k)$.

Nonsmooth Optimization on Riemannian Manifolds

We are looking for **numerical algorithms** to find

$$\arg \min_{p \in \mathcal{M}} f(p)$$

where

- ▶ \mathcal{M} is a Riemannian manifold
- ▶ $f: \mathcal{M} \rightarrow \overline{\mathbb{R}}$ is a function
- ⚠ f might be **nonsmooth** and/or **nonconvex**
- ⚠ \mathcal{M} might be **high-dimensional**

A Riemannian Manifold \mathcal{M}

A d -dimensional Riemannian manifold can be informally defined as a set \mathcal{M} covered with a “suitable” collection of charts, that identify subsets of \mathcal{M} with open subsets of \mathbb{R}^d and a continuously varying inner product on the tangent spaces.

[Absil, Mahony, and Sepulchre [2008](#)]

A Riemannian Manifold \mathcal{M}

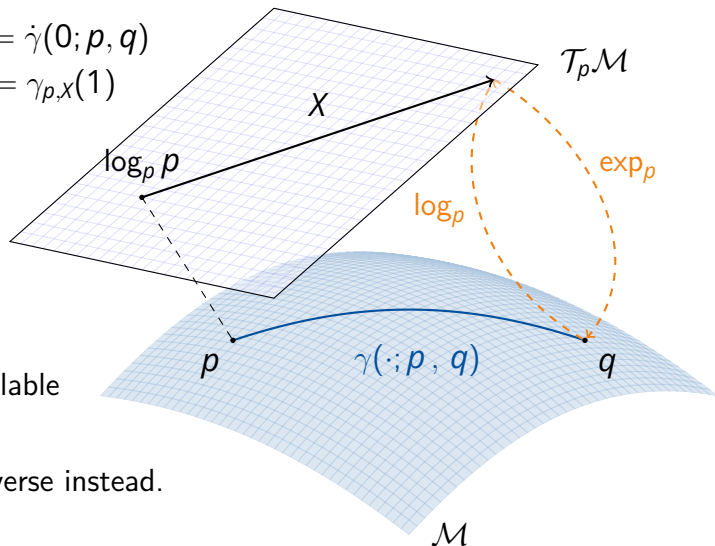
Notation.

- ▶ Logarithmic map $\log_p q = \dot{\gamma}(0; p, q)$
- ▶ Exponential map $\exp_p X = \gamma_{p,X}(1)$
- ▶ Geodesic $\gamma(\cdot; p, q)$
- ▶ Tangent space $\mathcal{T}_p \mathcal{M}$
- ▶ inner product $(\cdot, \cdot)_p$

Numerics.

\exp_p and \log_p maybe not available efficiently/ in closed form

\Rightarrow use a retraction and its inverse instead.





Manifolds.jl & Manopt.jl – Why Julia?

Goals.

- ▶ abstract definition of manifolds
 - ⇒ implement abstract solvers on a generic manifold
 - ▶ well-documented and well-tested
 - ▶ fast.
- ⇒ “Run your favourite solver on your favourite manifold”.

Why Julia?

julialang.org

- ▶ high-level language, properly typed
- ▶ multiple dispatch (cf. `f(x)`, `f(x::Number)`, `f(x::Int)`)
- ▶ just-in-time compilation, solves two-language problem
⇒ “nice to write” and as fast as C/C++
- ▶ I like the community



[Axen, Baran, RB, and Rzecki 2023]

Goal. Provide an interface to implement and use Riemannian manifolds.

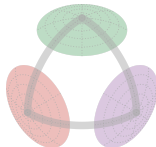
Interface `AbstractManifold` to model manifolds

Functions like `exp(M, p, X)`, `log(M, p, X)` or `retract(M, p, X, method)`.

Decorators for implicit or explicit specification of an embedding, a metric, or a group,

Efficiency by providing in-place variants like `exp!(M, q, p, X)`

Manifolds.jl



[Axen, Baran, RB, and Rzecki 2023]

Goal. Provide a library of Riemannian manifolds, that is efficiently implemented and well-documented

Meta. generic implementations for $\mathcal{M}^{n \times m}$, $\mathcal{M}_1 \times \mathcal{M}_2$, vector- and tangent-bundles, esp. $T_p\mathcal{M}$, or Lie groups

Library. Implemented functions for

- ▶ Circle, Sphere, Torus, Hyperbolic, Projective Spaces, Hamiltonian
- ▶ (generalized, symplectic) Stiefel, Rotations
- ▶ (generalized, symplectic) Grassmann, fixed rank matrices
- ▶ Symmetric Positive Definite matrices, with fixed determinant
- ▶ (several) Multinomial, (skew-)symmetric, and symplectic matrices
- ▶ Tucker & Oblique manifold, Kendall's Shape space
- ▶ probability simplex, orthogonal and unitary matrices, Rotations, ...

Concrete Manifold Examples.

Before first run] `add Manifolds` to install the package.

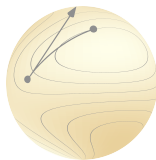
Load packages with `using Manifolds` and

- ▶ Euclidean space `M1 = \mathbb{R}^3` and 2-sphere `M2 = Sphere(2)`
- ▶ their product manifold `M3 = M1 \times M2`
- ▶ A signal of rotations `M4 = SpecialOrthogonal(3)^10`
- ▶ SPDs `M5 = SymmetricPositiveDefinite(3)` (affine invariant metric)
- ▶ a different metric `M6 = MetricManifold(M5, LogCholeskyMetric())`

Then for `any` of these

- ▶ Generate a point `p=rand(M)` and a vector `X = rand(M; vector_at=p)`
- ▶ and for example `exp(M, p, X)`, or in-place `exp!(M, q, p, X)`

Manopt.jl



Goal. Provide optimization algorithms on Riemannian manifolds.

Features. Given a `Problem p` and a `SolverState s`,
implement `initialize_solver!(p, s)` and `step_solver!(p, s, i)`
⇒ an algorithm in the `Manopt.jl` interface

Highlevel interfaces like `gradient_descent(M, f, grad_f)`
on any manifold `M` from `Manifolds.jl`.

All provide `debug` output, `recording`, `cache` & `counting` capabilities,
as well as a library of `step sizes` and `stopping criteria`.

Manopt family.



manoptjl.org

[RB 2022]



manopt.org

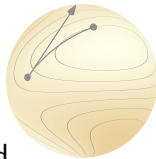
[Boumal, Mishra, Absil, and Sepulchre 2014]



pymanopt.org

[Townsend, Koep, and Weichwald 2016]

List of Algorithms in Manopt.jl



Derivative Free Nelder-Mead, Particle Swarm, CMA-ES

Subgradient-based Subgradient Method, Convex Bundle Method,
Proximal Bundle Method

Gradient-based Gradient Descent, Conjugate Gradient, Stochastic,
Momentum, Nesterov, Averaged, ...
Quasi-Newton with (L-)BFGS, DFP, Broyden, SR1,...
Levenberg-Marquard

Hessian-based Trust Regions, Adaptive Regularized Cubics (ARC)

nonsmooth Chambolle-Pock, Douglas-Rachford, Cyclic Proximal Point

constrained Augmented Lagrangian, Exact Penalty, Frank-Wolfe

nonconvex Difference of Convex Algorithm, DCPA



Illustrating a few Keyword Arguments

Given cost $f(M, p)$ and gradient $\text{grad}_f(M, p)$, a manifold M and a start point p_0 .

- ▶ `q = gradient_descent(M, f, grad_f, p0)` to perform gradient descent
- ▶ With Euclidean cost $f(E, p)$ and gradient $\nabla f(E, p)$, use for conversion
`q = gradient_descent(M, f, ∇f , p0; objective_type=:Euclidean)`
- ▶ print iteration number, cost and change every 10th iterate

```
q = gradient_descent(M, f, grad_f, p0;
    debug=[:Iteration, :Cost, :Change, 10, "\n"]
)
```
- ▶ record `record=[:Iterate, :Cost, :Change]`, `return_state=true`
 Access: `get_solver_result(q)` and `get_record(q)`
- ▶ modify stop: `stopping_criterion = StopAfterIteration(100)`
- ▶ cache calls `cache=(:LRU, [:Cost, :Gradient], 25)` (uses `LRUCache.jl`)
- ▶ count calls `count=[:Cost, :Gradient]` (prints with `return_state=true`)

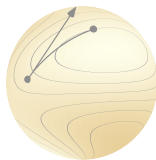
Numerical Examples

The Riemannian Convex Bundle Method

[RB, Herzog, and Jasa 2024]

- ▶ Given $f: \mathcal{C} \rightarrow \mathbb{R}$ on a (geodesically) convex set $\mathcal{C} \subset \mathcal{M}$
- ▶ collect
 - ▶ subgradients $X_{q^{(k)}} \in \partial f(q^{(k)})$
 - ▶ stabilisation centers $p^{(k)}$ (“best” iterates)
- ▶ use this information to
 - ▶ determine the next descent direction $d^{(k)} \in \mathcal{T}_{p^{(k)}}\mathcal{M}$ by solving a QP in $\mathcal{T}_{p^{(k)}}\mathcal{M}$
 - ▶ where $d^{(k)} \in \partial_{\mathcal{C}^{(k)}}f(p^{(k)})$
- ▶ we stop when both
 - ▶ the approximation $\partial_{\mathcal{C}^{(k)}}f(p^{(k)})$ of $\partial f(p^{(k)})$ is “good enough”
 - ▶ $\|d^{(k)}\|$ is “small enough”

The Convex Bundle Method in Manopt.jl



In `Manopt.jl` a solver call looks like¹

```
p = convex_bundle_method(M, f, ∂f, p0;
    diameter = δ, k_max = Ω, m = 10-3, kwargs...)
)
```

where

- ▶ `M` is a Riemannian manifold
- ▶ `f` is the objective function
- ▶ `∂f` is a subgradient of the objective function
- ▶ `p0` is an initial point on the manifold

The default stopping criterion for the algorithm is set to

$$-\xi^{(k)} \leq 10^{-8}.$$

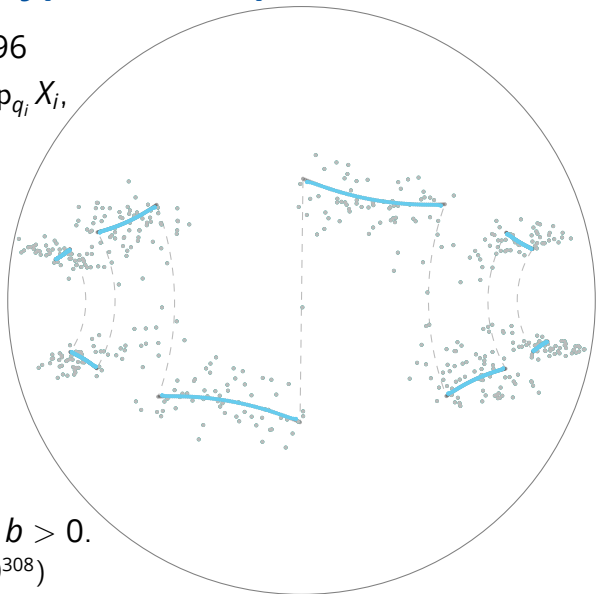
¹full documentation: manoptjl.org/stable/solvers/convex_bundle_method/

Denoising a Signal on Hyperbolic Space \mathcal{H}^2

- ▶ signal $q \in \mathcal{M}$, $(\mathcal{H}^2)^n$, $n = 496$
- ▶ noisy signal $\bar{q} \in \mathcal{M}$, $\bar{q}_i = \exp_{q_i} X_i$, $\sigma = 0.1$
- ▶ ROF Model:

$$\arg \min_{p \in \mathcal{M}} \frac{1}{n} d_{\mathcal{M}}(p, q)^2 + \alpha \sum_{i=1}^{n-1} d_{\mathcal{H}^2}(p_i, p_{i+1})$$

- ▶ Setting $\alpha = 0.05$ yields reconstruction p^* .
- ▶ in RCBM: set $\text{diam}(\text{dom } f) = b > 0$.
(in practice: $b = \text{floatmax}() \approx 10^{308}$)



Algorithms for Denoising a Signal

- ▶ Riemannian Convex Bundle Method (RCBM) [RB, Herzog, and Jasa 2024]
- ▶ Proximal Bundle Algorithm (PBA) [Hoseini Monjezi, Nobakhtian, and Pouryayevali 2021]
- ▶ Subgradient Method (SGM) [O. Ferreira and Oliveira 1998]
- ▶ Cyclic Proximal Point Algorithm (CPPA) [Bačák 2014]

Algorithm	Iter.	Time (sec.)	Objective	Error
RCBM	3417	51.393	1.7929×10^{-3}	3.3194×10^{-4}
PBA	15 000	102.387	1.8153×10^{-3}	4.3874×10^{-4}
SGM	15 000	99.604	1.7920×10^{-3}	3.3080×10^{-4}
CPPA	15 000	94.200	1.7928×10^{-3}	3.3230×10^{-4}

The Riemannian DC Algorithm

[RB, O. P. Ferreira, Santos, and Souza 2024]

To solve a Difference of Convex problem

$$\arg \min_{p \in \mathcal{M}} g(p) - h(p).$$

use

The Riemannian Difference of Convex Algorithm.

Input: An initial point $p^{(0)} \in \text{dom}(g)$, g and $\partial_{\mathcal{M}} h$

- 1: Set $k = 0$.
- 2: **while** not converged **do**
- 3: Take $X^{(k)} \in \partial_{\mathcal{M}} h(p^{(k)})$
- 4: Compute the next iterate $p^{(k+1)}$ as

$$p^{(k+1)} \in \arg \min_{p \in \mathcal{M}} g(p) - (X^{(k)}, \log_{p^{(k)}} p)_{p^{(k)}}.$$

- 5: Set $k \leftarrow k + 1$
- 6: **end while**

The Difference of Convex Algorithm in Manopt.jl

The algorithm is implemented and released in Julia using `Manopt.jl`². It can be used with any manifold from `Manifolds.jl`

A solver call looks like

```
q = difference_of_convex_algorithm(M, f, g, ∂h, p0)
```

where one has to implement $f(M, p)$, $g(M, p)$, and $\partial h(M, p)$.

- ▶ a sub problem is generated if keyword `grad_g=` is set
- ▶ an efficient version of its cost and gradient is provided
- ▶ you can specify the sub-solver using `sub_state=` to also set up the specific parameters of your favourite algorithm

²see https://manoptjl.org/stable/solvers/difference_of_convex/

Rosenbrock and First Order Methods

Problem. We consider the classical Rosenbrock example³

$$\arg \min_{x \in \mathbb{R}^2} a(x_1^2 - x_2)^2 + (x_1 - b)^2,$$

where $a, b > 0$, usually $b = 1$ and $a \gg b$, here: $a = 2 \cdot 10^5$.

Known Minimizer $x^* = \begin{pmatrix} b \\ b^2 \end{pmatrix}$ with cost $f(x^*) = 0$.

Goal. Compare first-order methods, e. g. using the (Euclidean) gradient

$$\nabla f(x) = \begin{pmatrix} 4a(x_1^2 - x_2) \\ -2a(x_1^2 - x_2) \end{pmatrix} + \begin{pmatrix} 2(x_1 - b) \\ 0 \end{pmatrix}$$

A “Rosenbrock-Metric” on \mathbb{R}^2

In our Riemannian framework, we can introduce a new metric on \mathbb{R}^2 as

$$G_p := \begin{pmatrix} 1 + 4p_1^2 & -2p_1 \\ -2p_1 & 1 \end{pmatrix}, \text{ with inverse } G_p^{-1} = \begin{pmatrix} 1 & 2p_1 \\ 2p_1 & 1 + 4p_1^2 \end{pmatrix}.$$

We obtain $(X, Y)_p = X^T G_p Y$

The exponential and logarithmic map are given as

$$\exp_p(X) = \begin{pmatrix} p_1 + X_1 \\ p_2 + X_2 + X_1^2 \end{pmatrix}, \quad \log_p(q) = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 - (q_1 - p_1)^2 \end{pmatrix}.$$

`Manifolds.jl`:

Implement these functions on `MetricManifold(\mathbb{R}^2 , RosenbrockMetric())`.

The Riemannian Gradient w.r.t. the new Metric

Let $f: \mathcal{M} \rightarrow \mathbb{R}$. Given the Euclidean gradient $\nabla f(p)$, its Riemannian gradient $\text{grad} f: \mathcal{M} \rightarrow T\mathcal{M}$ is given by

$$\text{grad} f(p) = G_p^{-1} \nabla f(p).$$

While we could implement this denoting $\nabla f(p) = (f'_1(p) \ f'_2(p))^T$ using

$$\left\langle \text{grad} f(q), \log_q p \right\rangle_q = (p_1 - q_1)f'_1(q) + (p_2 - q_2 - (p_1 - q_1)^2)f'_2(q),$$

but it is [automatically](#) done in [Manopt.jl](#).

The Experiment Setup

Algorithms. We now compare

1. The Euclidean gradient descent algorithm on \mathbb{R}^2 ,
2. The Riemannian gradient descent algorithm on \mathcal{M} ,
3. The Difference of Convex Algorithm on \mathbb{R}^2 ,
4. The Difference of Convex Algorithm on \mathcal{M} .

For DCA third we split f into $f(x) = g(x) - h(x)$ with

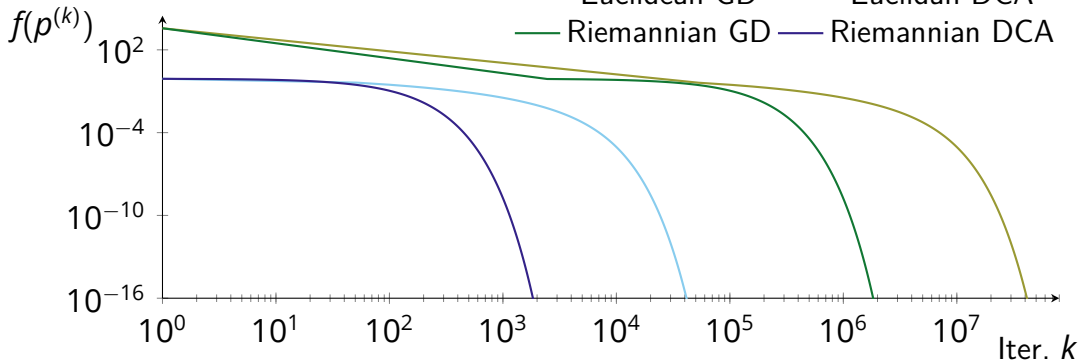
$$g(x) = a(x_1^2 - x_2)^2 + 2(x_1 - b)^2 \quad \text{and} \quad h(x) = (x_1 - b)^2.$$

Initial point. $p_0 = \frac{1}{10} \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ with cost $f(p_0) \approx 7220.81$.

Stopping Criterion.

$$d_{\mathcal{M}}(p^{(k)}, p^{(k-1)}) < 10^{-16} \text{ or } \|\text{grad} f(p^{(k)})\|_p < 10^{-16}.$$

Results



Algorithm	Runtime (sec.)	# Iterations
Euclidean GD	305.567	53 073 227
Euclidean DCA	58.268	50 588
Riemannian GD	18.894	2 454 017
Riemannian DCA	7.704	2 459

Summary

- ▶ `ManifolddsBase.jl` provides an interface to implement a manifold
- ▶ `Manifolds.jl` implements a library of manifolds using the interface
- ▶ `Manopt.jl` provides optimization algorithms on these manifolds

Outlook.

- ▶ couple `Manopt.jl` with (Euclidean) AD tools using `ManifoldDiff.jl`
- ▶ What is (Fenchel) duality on manifolds?

Selected References



Axen, S. D., M. Baran, RB, and K. Rzecki (2023). “Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds”. In: *ACM Transactions on Mathematical Software*. Accepted for publication. DOI: [10.1145/3618296](https://doi.org/10.1145/3618296). arXiv: [2106.08777](https://arxiv.org/abs/2106.08777).



RB (2022). “Manopt.jl: Optimization on Manifolds in Julia”. In: *Journal of Open Source Software* 7.70, p. 3866. DOI: [10.21105/joss.03866](https://doi.org/10.21105/joss.03866).



RB, O. P. Ferreira, E. M. Santos, and J. C. d. O. Souza (2024). “The difference of convex algorithm on Hadamard manifolds”. In: *Journal of Optimization Theory and Applications*. DOI: [10.1007/s10957-024-02392-8](https://doi.org/10.1007/s10957-024-02392-8). arXiv: [2112.05250](https://arxiv.org/abs/2112.05250).



RB, R. Herzog, and H. Jasa (2024). *The Riemannian convex bundle method*. arXiv: [2402.13670](https://arxiv.org/abs/2402.13670).



Boumal, N. (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press. URL: <https://www.nicolasboumal.net/book>.

Interested in Numerical Differential Geometry?

Join  numdiffgeo.zulipchat.com!



ronnybergmann.net/talks/2024-EURO-Manoptjl.pdf