

Nonsmooth Optimization on Riemannian Manifolds

Ronny Bergmann

NTNU, Trondheim, Norway.

Seminar at RIKEN AIP

Tokyo, Japan, February 13, 2026.



Nonsmooth Optimization on Riemannian Manifolds

NTNU

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

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

where

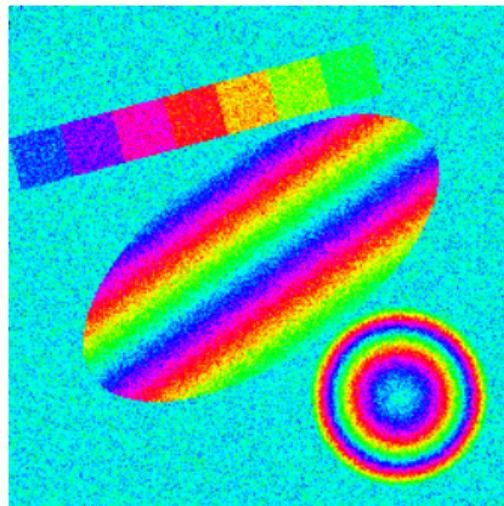
- ▶ \mathcal{M} is a Riemannian manifold
- ▶ $f: \mathcal{M} \rightarrow \bar{\mathbb{R}}$ is a function
- ⚠ f might be **nonsmooth** and/or **nonconvex**
- ⚠ \mathcal{M} might be **high-dimensional**
- 💡 f has some “nice structure”

Manifold-valued signal and image processing

- ▶ variational models for
denoising, inpainting, deconvolution, segmentation, ...
- ▶ applications in medical imaging, computer vision
- ▲ nonlinear (non-Euclidean) data

Examples

- ▶ phase-valued data (\mathbb{S}^1)
- ▶ wind-fields, GPS (\mathbb{S}^2)
- ▶ DT-MRI ($\mathcal{P}(3)$)
- ▶ EBSD, (grain) orientations ($\text{SO}(n)$)



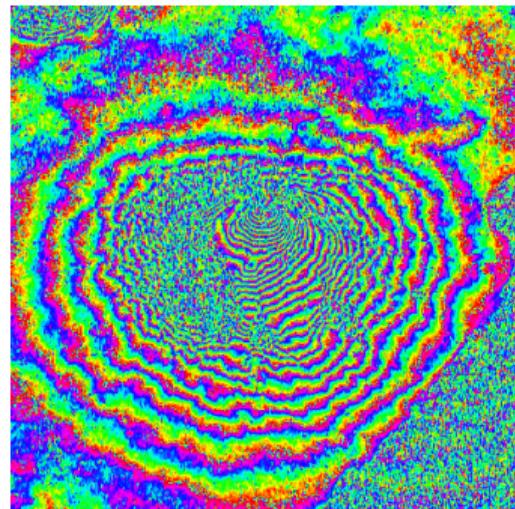
Artificial noisy phase-valued data.

Manifold-valued signal and image processing

- ▶ variational models for
denoising, inpainting, deconvolution, segmentation, ...
- ▶ applications in medical imaging, computer vision
- ▲ nonlinear (non-Euclidean) data

Examples

- ▶ phase-valued data (\mathbb{S}^1)
- ▶ wind-fields, GPS (\mathbb{S}^2)
- ▶ DT-MRI ($\mathcal{P}(3)$)
- ▶ EBSD, (grain) orientations ($\text{SO}(n)$)



InSAR-Data of Mt. Vesuvius.

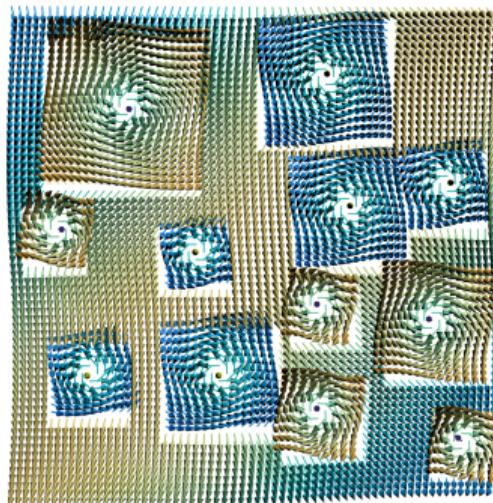
[Rocca, Prati, Guarnieri, 1997]

Manifold-valued signal and image processing

- ▶ variational models for
denoising, inpainting, deconvolution, segmentation, ...
- ▶ applications in medical imaging, computer vision
- ▲ nonlinear (non-Euclidean) data

Examples

- ▶ phase-valued data (\mathbb{S}^1)
- ▶ wind-fields, GPS (\mathbb{S}^2)
- ▶ DT-MRI ($\mathcal{P}(3)$)
- ▶ EBSD, (grain) orientations ($\text{SO}(n)$)



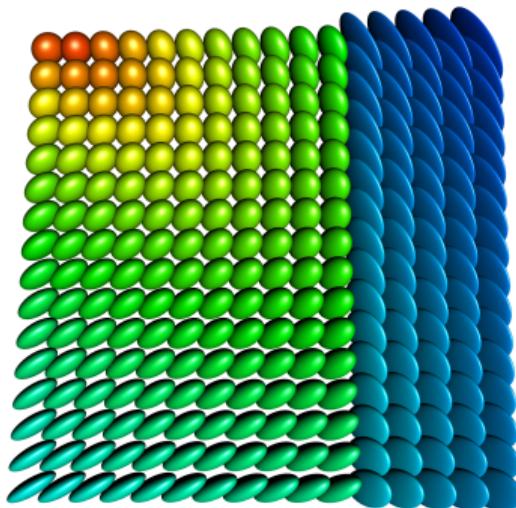
Artificial noisy data on the sphere \mathbb{S}^2 .

Manifold-valued signal and image processing

- ▶ variational models for
denoising, inpainting, deconvolution, segmentation, ...
- ▶ applications in medical imaging, computer vision
- ▲ nonlinear (non-Euclidean) data

Examples

- ▶ phase-valued data (\mathbb{S}^1)
- ▶ wind-fields, GPS (\mathbb{S}^2)
- ▶ DT-MRI ($\mathcal{P}(3)$)
- ▶ EBSD, (grain) orientations ($\text{SO}(n)$)



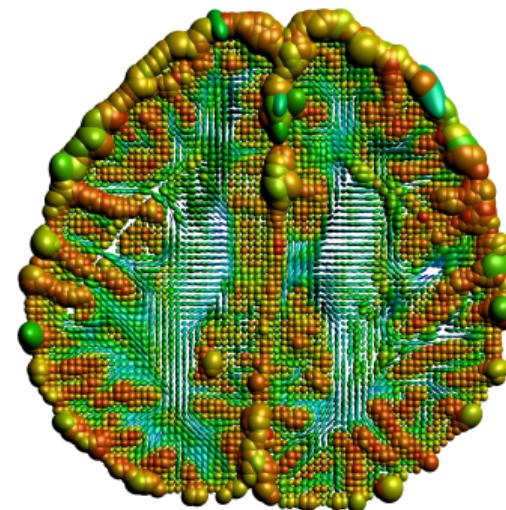
Artificial diffusion data,
each pixel is a sym. pos. def. matrix.

Manifold-valued signal and image processing

- ▶ variational models for
denoising, inpainting, deconvolution, segmentation, ...
- ▶ applications in medical imaging, computer vision
- ▲ nonlinear (non-Euclidean) data

Examples

- ▶ phase-valued data (\mathbb{S}^1)
- ▶ wind-fields, GPS (\mathbb{S}^2)
- ▶ DT-MRI ($\mathcal{P}(3)$)
- ▶ EBSD, (grain) orientations ($\text{SO}(n)$)



DT-MRI of the human brain.

Camino Project: cmic.cs.ucl.ac.uk/camino

Manifold-valued signal and image processing

- ▶ variational models for
denoising, inpainting, deconvolution, segmentation, ...
- ▶ applications in medical imaging, computer vision
- ▲ nonlinear (non-Euclidean) data

Examples

- ▶ phase-valued data (\mathbb{S}^1)
- ▶ wind-fields, GPS (\mathbb{S}^2)
- ▶ DT-MRI ($\mathcal{P}(3)$)
- ▶ EBSD, (grain) orientations ($\text{SO}(n)$)



Grain orientations in EBSD data.

MTEX toolbox: mtex-toolbox.github.io



Constraints and/or geometry

NTNU

constraints

- ▶ needs an embedding
- ▶ might not always yield a manifold
- 😊 slightly more flexible
- 🙁 algorithms have to deal with constraints
- 🙁 results might be infeasible

geometry

- ▶ might work agnostic of an embedding
- 😊 quotient manifolds
- 😊 we can use any unconstrained algorithm...
- 🙁 ...after adapting it to the manifold setting
- 😊 algorithms stay on the manifold
- ➡ always feasible

We can also consider a combination of both:
constrained optimization on manifolds.

[Liu, Boumal, 2019; RB, Herzog, 2019]

A Riemannian Manifold \mathcal{M}

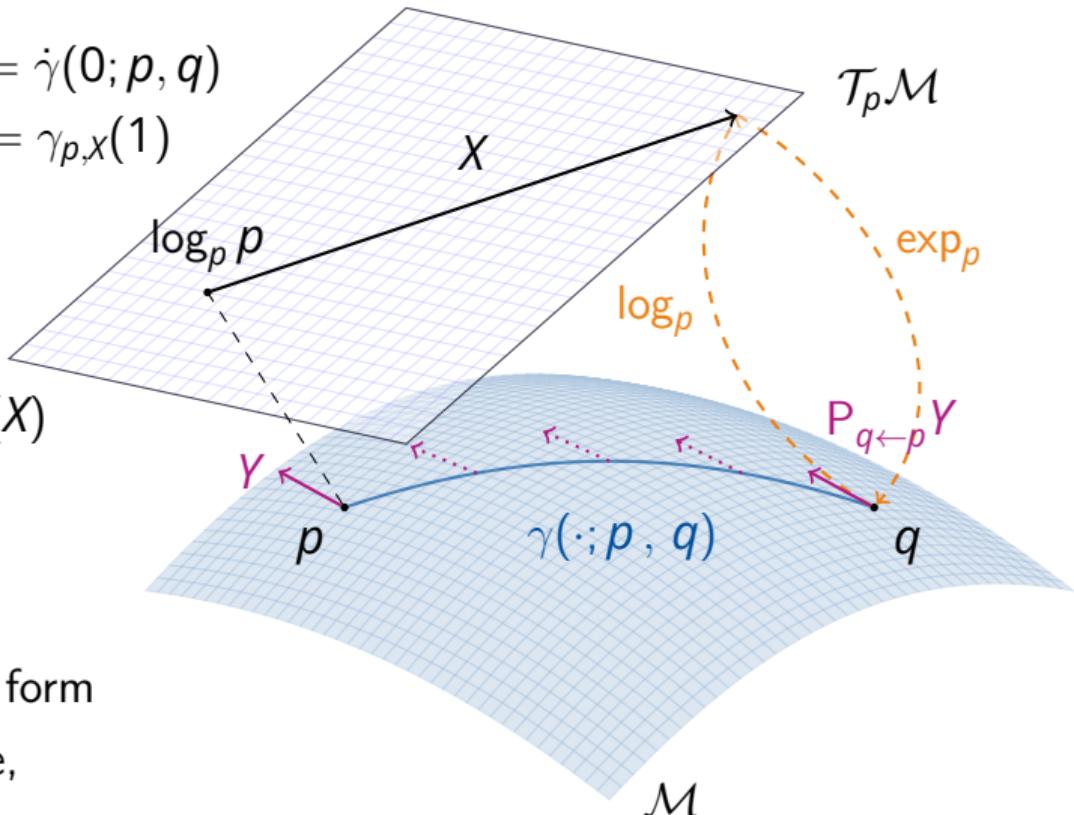
Notation.

- ▶ 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 $T_p\mathcal{M}$
- ▶ inner product $(\cdot, \cdot)_p$
- ▶ parallel transport $\text{PT}_{p \leftarrow q}(X)$
- ▶ distance function $d(p, q)$

Numerics.

\exp_p , \log_p , $\text{PT}_{p \leftarrow q}$ maybe
not avail, efficiently/in closed form

⇒ use a retraction, its inverse,
a vector transport instead



(Geodesic) Convexity



NTNU

[Sakai, 1996; Udriște, 1994]

A set $\mathcal{C} \subset \mathcal{M}$ is called (strongly geodesically) **convex**
if for all $p, q \in \mathcal{C}$ the geodesic $\gamma(\cdot; p, q)$ is unique and lies in \mathcal{C} .

A function $f: \mathcal{C} \rightarrow \overline{\mathbb{R}}$ is called (geodesically) **convex**
if for all $p, q \in \mathcal{C}$ the composition $f(\gamma(t; p, q)), t \in [0, 1]$, is convex.



The Riemannian Subdifferential

NTNU

Let \mathcal{C} be a convex set.

The **subdifferential** of f at $p \in \mathcal{C}$ is given by

[Ferreira, Oliveira, 2002; Lee, 2003; Udriște, 1994]

$$\partial_{\mathcal{M}} f(p) := \left\{ \xi \in \mathcal{T}_p^* \mathcal{M} \mid f(q) \geq f(p) + \langle \xi, \log_p q \rangle_p \text{ for } q \in \mathcal{C} \right\},$$

where

- ▶ $\mathcal{T}_p^* \mathcal{M}$ is the dual space of $\mathcal{T}_p \mathcal{M}$, also called **cotangent space**
- ▶ $\langle \cdot, \cdot \rangle_p$ denotes the duality pairing on $\mathcal{T}_p^* \mathcal{M} \times \mathcal{T}_p \mathcal{M}$
- ▶ numerically we use musical isomorphisms $X = \xi^\flat \in \mathcal{T}_p \mathcal{M}$ to obtain a subset of $\mathcal{T}_p \mathcal{M}$



The Proximal Point Algorithm

Euclidean case. For $f: \mathbb{R}^n \rightarrow \overline{\mathbb{R}}$, $\lambda > 0$, the proximal map given by

[Moreau, 1965; Rockafellar, 1970]

$$\text{prox}_{\lambda f}(x) = \arg \min_{y \in \mathbb{R}^n} \left\{ f(y) + \frac{1}{2\lambda} \|y - x\|^2 \right\}.$$

Riemannian case. For $f: \mathcal{M} \rightarrow \overline{\mathbb{R}}$, $\lambda > 0$, the proximal map is given by

[Ferreira, Oliveira, 2002]

$$\text{prox}_{\lambda f}(p) = \arg \min_{q \in \mathcal{M}} \left\{ f(q) + \frac{1}{2\lambda} d(p, q)^2 \right\}.$$

For both. A minimizer p^* of f is a fixed point for $\text{prox}_{\lambda f}$.

Proximal Point Algorithm (PPA). Given $p^{(0)} \in \mathcal{M}$, $\lambda_k > 0$, iterate

$$p^{(k+1)} = \text{prox}_{\lambda_k f}(p^{(k)}).$$



The Cyclic Proximal Point Algorithm

[Bertsekas, 2011; Bačák, 2014]

For a splitting $f(p) = \sum_{i=1}^c f_i(p)$ and some $p_0 \in \mathcal{M}$, we can use

$$p_{k+\frac{i+1}{c}} = \text{prox}_{\lambda_k f_i}(p_{k+\frac{i}{c}}), \quad i = 0, \dots, c-1, \quad k = 0, 1, \dots$$

On a Hadamard manifold \mathcal{M} : Convergence to a minimizer of f if

- ▶ all f_i proper, convex, lower semi-continuous
- ▶ $\{\lambda_k\}_{k \in \mathbb{N}} \in \ell_2(\mathbb{N}) \setminus \ell_1(\mathbb{N})$.
- ▶ also for
 - ▶ random order of the $\text{prox}_{\lambda f_i}$
 - ▶ inexact evaluations of the $\text{prox}_{\lambda f_i}$

[Bačák, RB, Steidl, Weinmann, 2016]

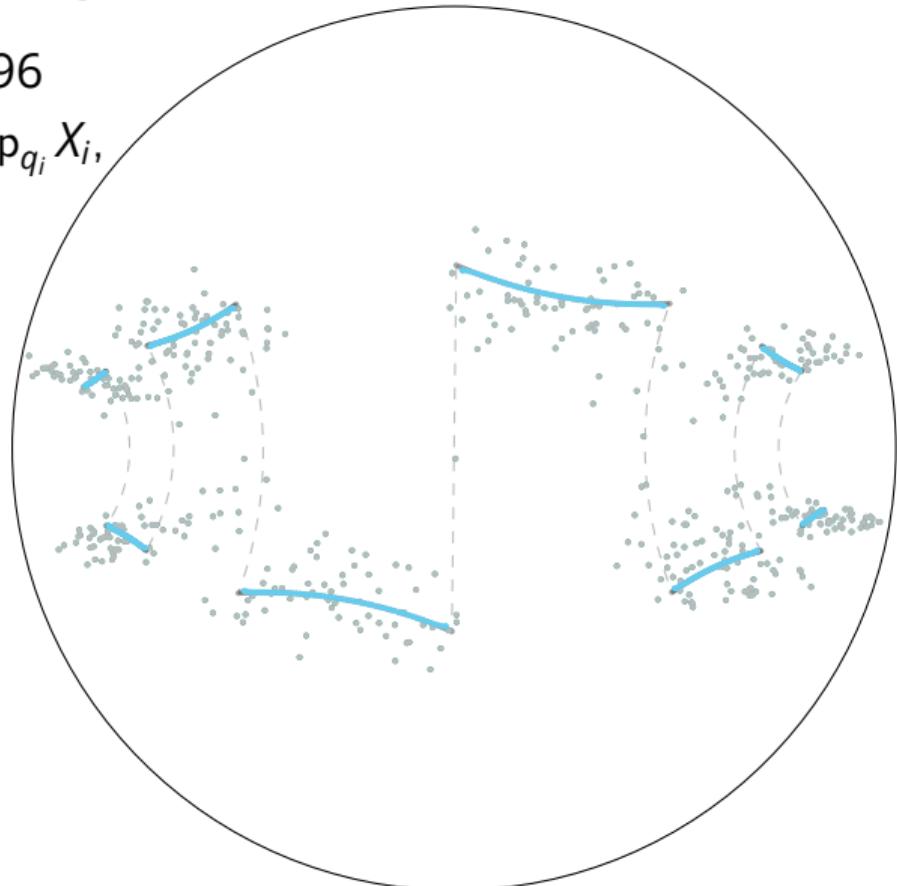
! no convergence rate

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:

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

- ▶ Setting $\alpha = 0.05$ yields reconstruction p^* .





Algorithms for Denoising a Signal

NTNU

- ▶ Riemannian Convex Bundle Method (RCBM) [RB, Herzog, Jasa, 2024]
- ▶ Proximal Bundle Algorithm (PBA) [Hoseini Monjezi, Nobakhtian, Pouryayevali, 2021]
- ▶ Subgradient Method (SGM) [Ferreira, 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 Douglas Rachford Algorithm

For a splitting $f = g + h$, where both are possibly nonsmooth, use the reflection at the proximal map

$$R_{\lambda f}(p) = \exp_{\text{prox}_{\lambda f}(p)}(-\log_{\text{prox}_{\lambda f}(p)}(p)) \quad (\text{Euclidean: } 2\text{ prox}_{\lambda f}(x) - x)$$

The Douglas Rachford algorithm reads for some $r^{(0)} \in \mathcal{M}$, $\eta > 0$

$$p^{(k)} = R_{\eta g}(r^{(k)})$$

[RB, Persch, Steidl, 2016]

$$q^{(k)} = R_{\eta h}(p^{(k)})$$

$$r^{(k+1)} = \gamma(\lambda_k; r^{(k)}, q^{(k)}) \quad (\gamma \text{ is a geodesic})$$

- ▶ converges on Hadamard manifolds if
 - ▶ g, h proper, convex, lsc.
 - ▶ $\lambda_k \in [0, 1]$ and $\sum_k \lambda_k(1 - \lambda_k) = \infty$
- ▶ ...to a fixed point of $R_{\lambda g} \circ R_{\lambda h}$ (in $r^{(k)}$)
- ▶ ...to a minimizer of f in the “shadow iterates” $\text{prox}_{\eta g}(r^{(k)})$



The Fenchel Conjugate

NTNU

The Fenchel conjugate of a function $f: \mathbb{R}^n \rightarrow \overline{\mathbb{R}}$ is given by

$$f^*(\xi) := \sup_{x \in \mathbb{R}^n} \langle \xi, x \rangle - f(x) = \sup_{x \in \mathbb{R}^n} \begin{pmatrix} \xi \\ -1 \end{pmatrix}^\top \begin{pmatrix} x \\ f(x) \end{pmatrix}$$

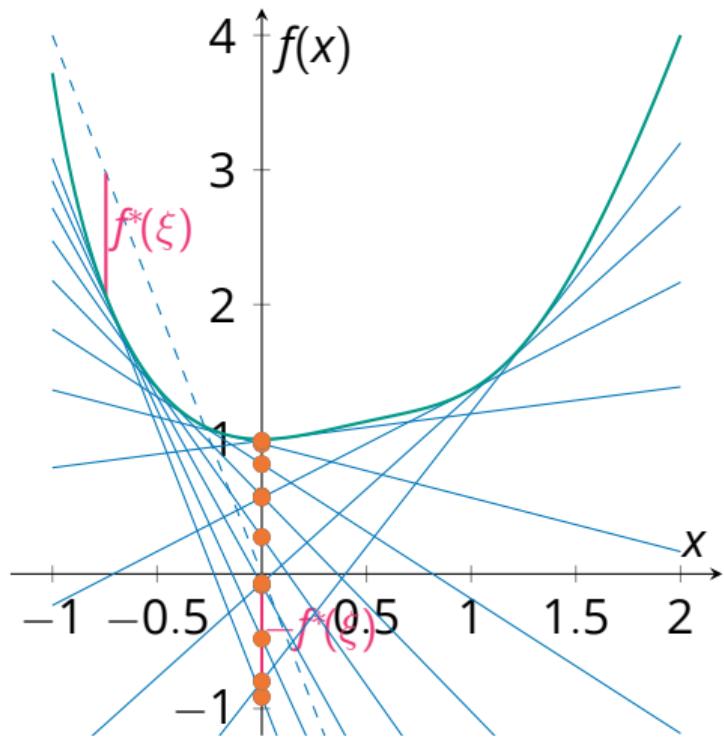
- ▶ given $\xi \in \mathbb{R}^n$: maximize the distance between $\xi^\top \cdot$ and f
- ▶ can also be written in the epigraph

The Fenchel biconjugate reads

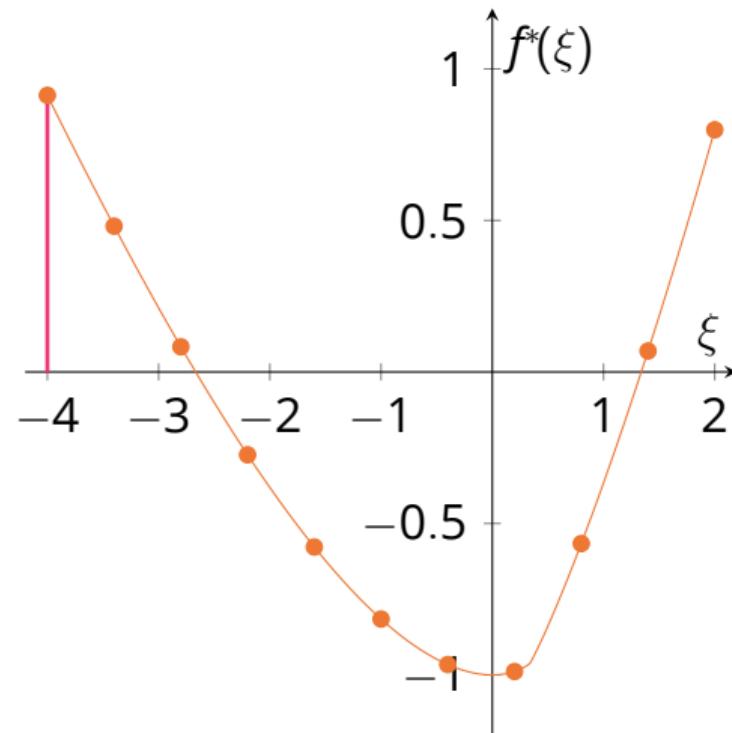
$$f^{**}(x) = (f^*)^*(x) = \sup_{\xi \in \mathbb{R}^n} \langle \xi, x \rangle - f^*(\xi).$$

Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*





The (Riemannian) m -Fenchel Conjugate

[RB, Herzog, Silva Louzeiro, Tenbrinck, Vidal-Núñez, 2021]

Idea. Localize to $\mathcal{C} \subset \mathcal{M}$ around a point m which “acts as” 0.

The m -Fenchel conjugate of a function $f: \mathcal{C} \rightarrow \overline{\mathbb{R}}$ is given by

$$f_m^*(\xi_m) := \sup_{X \in \mathcal{L}_{\mathcal{C},m}} \{ \langle \xi_m, X \rangle - f(\exp_m X) \},$$

where $\mathcal{L}_{\mathcal{C},m} := \{X \in T_m \mathcal{M} \mid q = \exp_m X \in \mathcal{C} \text{ and } \|X\|_p = d(q, p)\}$.

Let $m' \in \mathcal{C}$. The mm' -Fenchel-biconjugate $F_{mm'}^{**}: \mathcal{C} \rightarrow \overline{\mathbb{R}}$ is given by

$$F_{mm'}^{**}(p) = \sup_{\xi_{m'} \in T_{m'}^* \mathcal{M}} \{ \langle \xi_{m'}, \log_{m'} p \rangle - F_m^*(P_{m \leftarrow m'} \xi_{m'}) \},$$

where usually we only use the case $m = m'$.

The exact Riemannian Chambolle–Pock Algorithm

[RB, Herzog, Silva Louzeiro, Tenbrinck, Vidal-Núñez, 2021; Valkonen, 2014; Chambolle, Pock, 2011]

To solve

$$\arg \min_{p \in \mathcal{C} \subset \mathcal{M}} \{f(p) + g(\Lambda(p))\},$$

Input: $m, p^{(0)} \in \mathcal{C} \subset \mathcal{M}$, $n = \Lambda(m)$, $\xi_n^{(0)} \in \mathcal{T}_n^*\mathcal{N}$, and $\sigma, \tau, \theta > 0$

1: $k \leftarrow 0$

2: $\bar{p}^{(0)} \leftarrow p^{(0)}$

3: **while** not converged **do**

4: $\xi_n^{(k+1)} \leftarrow \text{prox}_{\tau g_n^*}(\xi_n^{(k)} + \tau (\log_n \Lambda(\bar{p}^{(k)}))^{\flat})$

5: $p^{(k+1)} \leftarrow \text{prox}_{\sigma f}\left(\exp_{p^{(k)}}\left(P_{p^{(k)} \leftarrow m}(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}])^\sharp\right)\right)$

6: $\bar{p}^{(k+1)} \leftarrow \exp_{p^{(k+1)}}(-\theta \log_{p^{(k+1)}} p^{(k)})$

7: $k \leftarrow k + 1$

8: **end while**

Output: $p^{(k)}$



Proximal Gradient

NTNU

For a splitting $f = g + h$, where g is smooth and h is possibly nonsmooth, both are convex.

The proximal gradient method reads for given $p^{(0)} \in \mathcal{M}$, $\lambda_k \in (0, \frac{1}{L}]$ reads

[RB, Jasa, John, Pfeffer, 2025b]

$$p^{(k+1)} = \text{prox}_{\lambda_k h}(\exp_{p^{(k)}}(-\lambda_k \text{grad } g(p^{(k)}))).$$

- ▶ convergence rates: sublinear (convex) linear (strongly convex)
 - ▶ a generalization of the prox-grad inequality
 - ▶ even the **nonconvex** case: sublinear convergence to ε -stationary points
- [RB, Jasa, John, Pfeffer, 2025a]
- ! though here: proximal map maybe not unique minimizer



The Riemannian DC Algorithm

To solve a Difference of Convex problem

[RB, Ferreira, Santos, Souza, 2024]

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

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**

Convergence of the Riemannian DCA

Let $\{p^{(k)}\}_{k \in \mathbb{N}}$ and $\{X^{(k)}\}_{k \in \mathbb{N}}$ be the iterates and subgradients of the RDCA.

Theorem.

[RB, Ferreira, Santos, Souza, 2024]

If \bar{p} is a cluster point of $\{p^{(k)}\}_{k \in \mathbb{N}}$, then $\bar{p} \in \text{dom}(g)$ and there exists a cluster point \bar{X} of $\{X^{(k)}\}_{k \in \mathbb{N}}$ s.t. $\bar{X} \in \partial g(\bar{p}) \cap \partial h(\bar{p})$.

⇒ Every cluster point of $\{p^{(k)}\}_{k \in \mathbb{N}}$, if any, is a critical point of f .

Proposition.

[RB, Ferreira, Santos, Souza, 2024]

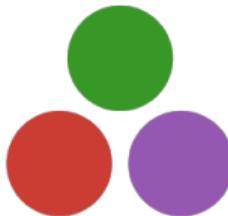
Let g be σ -strongly (geodesically) convex. Then

$$f(p^{(k+1)}) \leq f(p^{(k)}) - \frac{\sigma}{2} d^2(p^{(k)}, p^{(k+1)})$$

and $\sum_{k=0}^{\infty} d^2(p^{(k)}, p^{(k+1)}) < \infty$, so in particular $\lim_{k \rightarrow \infty} d(p^{(k)}, p^{(k+1)}) = 0$.

Software

Goals of the Software – Why Julia?



Goals.

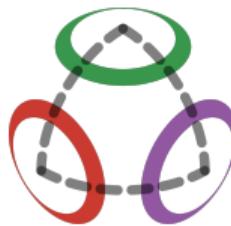
- ▶ abstract definition of manifolds and optimization thereon
- ⇒ implement abstract solvers on a generic manifold
- ▶ well-documented and well-tested
- ▶ fast.
- ⇒ “Run your favourite solver on your favourite manifold”.

Why ⚙️ Julia?

julialang.org

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

ManifoldsBase.jl – Motivation



Goal. Provide a generic interface to manifolds for

- ▶ defining own (new) manifolds
- ▶ implementing **generic** algorithms on an arbitrary manifold \mathcal{M}

A Manifold. a Riemannian manifold is a subtype of `AbstractManifold{F}`

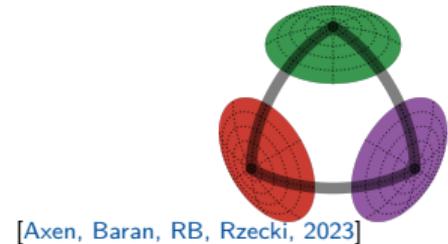
- ▶ $F \in \{\mathbb{R}, \mathbb{C}, \mathbb{H}\}$: field the manifold is build on
- ▶ stores all “general” information, (mainly) the manifold dimension
- ▶ example (from `Manifolds.jl`): `M = Sphere(2)`

Points and Tangent vectors.

- ▶ by default not typed, often `<:AbstractArray`
- ▶ we provide `<:AbstractManifoldPoint` and `<:TVector` for more advanced ones

Manifolds.jl

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



[Axen, Baran, RB, Rzecki, 2023]

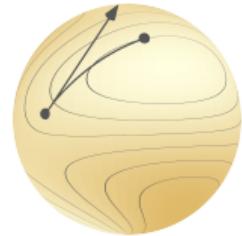
Meta. generic implementations for $\mathcal{M}^{n \times m}$, $\mathcal{M}_1 \times \mathcal{M}_2$,
vector- and tangent-bundles, esp. $T_p\mathcal{M}$, or Lie groups

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, ...

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

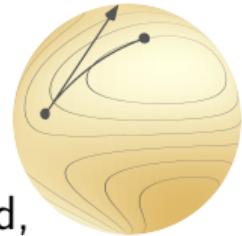
[Boumal, Mishra, Absil, Sepulchre, 2014]



pymanopt.org

[Townsend, Koep, Weichwald, 2016]

List of Algorithms in Manopt.jl



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

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-Marquardt

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

splitting Chambolle-Pock, Douglas-Rachford, Cyclic Proximal Point,
Proximal Gradient

constrained Augmented Lagrangian, Exact Penalty, Frank-Wolfe,
Projected Gradient, Interior Point Newton

nonconvex Difference of Convex Algorithm, DCPPA

A Numerical Example



The Difference of Convex Algorithm in Manopt.jl

The algorithm is implemented and released in Julia using `Manopt.jl`¹. 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 `∂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

Rosenbrock and First Order Methods

Problem. We consider the classical Rosenbrock example²

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

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

NTNU

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 $(X, Y)_p = X^\top G_p Y$

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

NTNU

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 $\nabla f(p) = (f'_1(p) \ f'_2(p))^\top$ 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

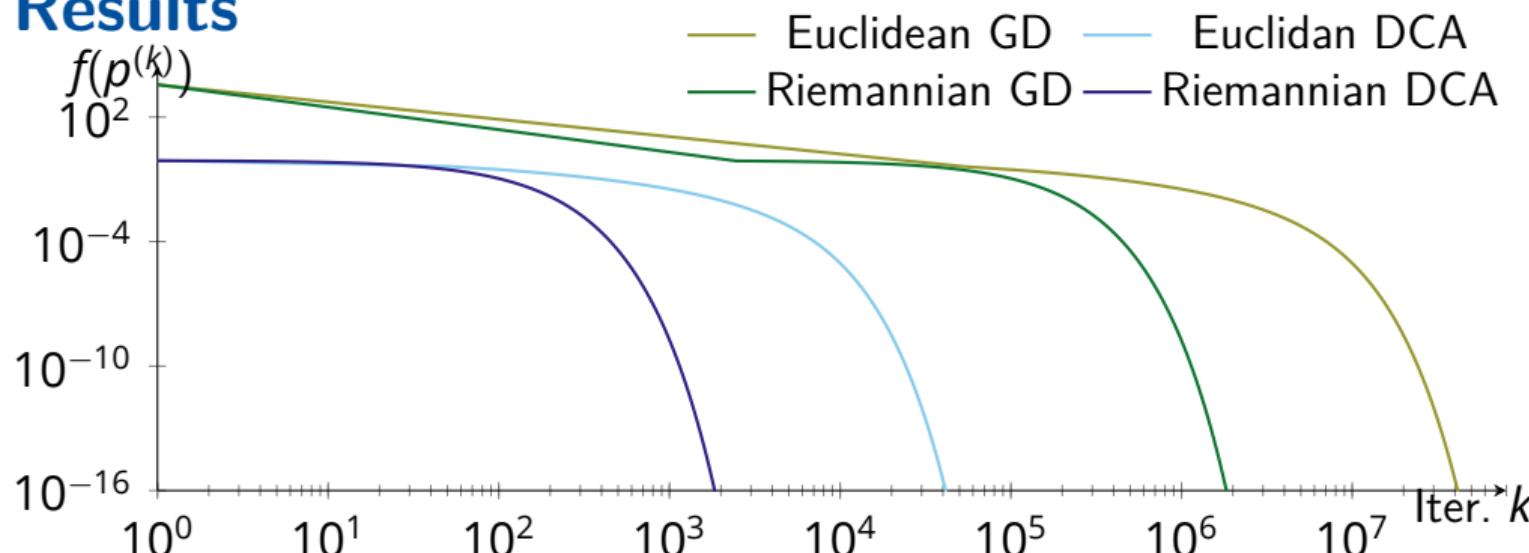
$$g(x) = a(x_1^2 - x_2)^2 + 2(x_1 - b)^2 \quad \text{and} \quad h(x) = (x_1 - b)^2.$$

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

NTNU

Nonsmooth optimization on manifolds appears in several applications.

- ▶ many algorithms can be generalized
- ▶ many properties carry over, like convergence results
- ▶ Fenchel duality can be generalized [Schiela, Herzog, RB, 2024]
- ▶ Manifolds.jl & Manopt.jl [RB, 2022; Axen, Baran, RB, Rzecki, 2023]
- ▶ numerical examples available in ManoptExamples.jl

- ▶ **Next.** LieGroups.jl [RB, Baran, 2026]

Selected References

-  Axen, S. D.; M. Baran; RB; K. Rzecki (2023). "Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds". *ACM Transactions on Mathematical Software* 49.4. DOI: [10.1145/3618296](https://doi.org/10.1145/3618296).
-  RB (2022). "Manopt.jl: Optimization on Manifolds in Julia". *Journal of Open Source Software* 7.70, p. 3866. DOI: [10.21105/joss.03866](https://doi.org/10.21105/joss.03866).
-  RB; M. Baran (2026). "Groups and smooth geometry using LieGroups.jl". *Proceedings of the JuliaCon Conferences*. Vol. 8. 79. *The Open Journal*, p. 195. DOI: [10.21105/jcon.00195](https://doi.org/10.21105/jcon.00195).
-  RB; O. P. Ferreira; E. M. Santos; J. C. d. O. Souza (2024). "The difference of convex algorithm on Hadamard manifolds". *Journal of Optimization Theory and Applications*. DOI: [10.1007/s10957-024-02392-8](https://doi.org/10.1007/s10957-024-02392-8).
-  RB; H. Jasa; P. John; M. Pfeffer (2025a). *The Intrinsic Riemannian Proximal Gradient Method for Nonconvex Optimization*. arXiv: [2506.09775](https://arxiv.org/abs/2506.09775).
-  — (2025b). *The Intrinsic Riemannian Proximal Gradient Method for Convex Optimization*. arXiv: [2507.16055](https://arxiv.org/abs/2507.16055).
-  RB; J. Persch; G. Steidl (2016). "A parallel Douglas Rachford algorithm for minimizing ROF-like functionals on images with values in symmetric Hadamard manifolds". *SIAM Journal on Imaging Sciences* 9.4, pp. 901–937. DOI: [10.1137/15M1052858](https://doi.org/10.1137/15M1052858).

