The Riemannian Chambolle–Pock Algorithm and Optimization on Manifolds in Julia

Ronny Bergmann

joint work with
Seth Axen, Mateusz Baran,
Roland Herzog, Maurício Silva Louzeiro, Daniel Tenbrinck, José Vidal-Núñez.

Technische Universität Chemnitz, Chemnitz, Germany

Oberwolfach-from-home
November 18, 2020

Contents

1. The Riemannian Chambolle–Pock Algorithm

The Model

We consider a minimization problem

$$\operatorname*{arg\,min}_{p\in\mathcal{C}} F(p) + G(\Lambda(p))$$

- $\mathcal{M}, \mathcal{N}$ are (high-dimensional) Riemannian Manifolds
- $F\colon \mathcal{M} \to \overline{\mathbb{R}}$ nonsmooth, (locally, geodesically) convex
- $G\colon \mathcal{N} \to \overline{\mathbb{R}}$ nonsmooth, (locally) convex
- $\Lambda\colon \mathcal{M} \to \mathcal{N}$ nonlinear
- $\mathcal{C} \subset \mathcal{M}$ strongly geodesically convex.

➔ In image processing:
choose a model, such that finding a minimizer yields the reconstruction

The ℓ^2-TV Model

[Rudin, Osher, and Fatemi 1992; Lellmann, Strekalovskiy, Koetter, and Cremers 2013; Weinmann, Demaret, and Storath 2014]

For a manifold-valued image $f \in \mathcal{M}$, $\mathcal{M} = \mathcal{N}^{d_1,\, d_2}$, we compute

$$\operatorname*{arg\,min}_{p\in\mathcal{M}} \frac{1}{\alpha}F(p) + G(\Lambda(p)), \qquad \alpha > 0,$$

with

- data term $F(p) = \frac{1}{2}d^2_{\mathcal{M}}(p, f)$
- “forward differences” $\Lambda\colon \mathcal{M} \to (T\mathcal{M})^{d_1-1,\, d_2-1,\, 2}$,

$$p \mapsto \Lambda(p) = \Big((\log_{p_i} p_{i+e_1},\ \log_{p_i} p_{i+e_2}) \Big)_{i\in\{1,\ldots,d_1-1\}\times\{1,\ldots,d_2-1\}}$$

- prior $G(X) = \lVert X \rVert_{g,q,1}$ similar to a collaborative TV [Duran, Moeller, Sbert, and Cremers 2016]

Splitting Methods & Algorithms

On a Riemannian manifold $\mathcal{M}$ we have

- Cyclic Proximal Point Algorithm (CPPA) [Bačák 2014]
- (parallel) Douglas–Rachford Algorithm (PDRA) [RB, Persch, and Steidl 2016]]

On $\mathbb{R}^n$ PDRA is known to be equivalent to [O'Connor and Vandenberghe 2018; Setzer 2011]

- Primal-Dual Hybrid Gradient Algorithm (PDHGA) [Esser, Zhang, and Chan 2010]
- Chambolle-Pock Algorithm (CPA) [Chambolle and Pock 2011; Pock, Cremers, Bischof, and Chambolle 2009]

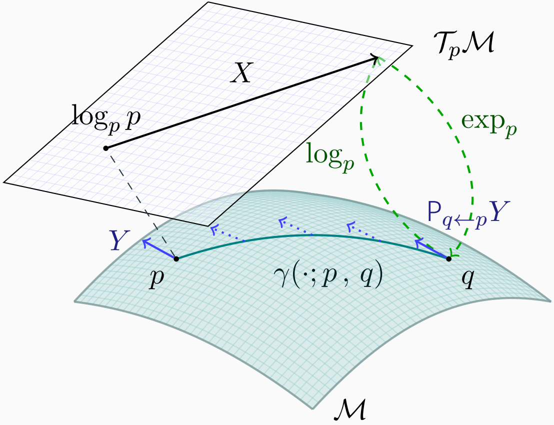
But on a Riemannian manifold $\mathcal{M}$: ⚠ no duality theory!

Goals of this talk.

Formulate Duality on a Manifold

Derive a Riemannian Chambolle–Pock Algorithm (RCPA)

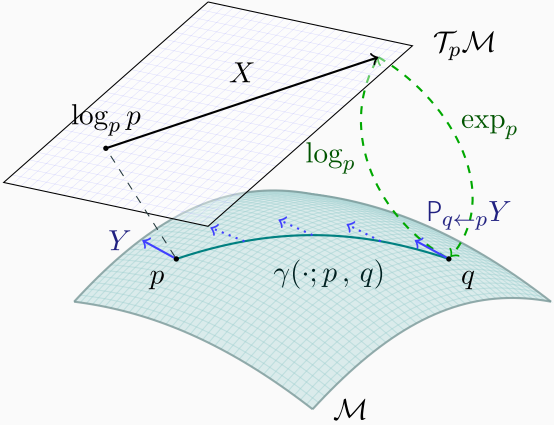
A d-dimensional Riemannian Manifold $\mathcal{M}$



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 d-dimensional Riemannian Manifold $\mathcal{M}$



Geodesic $\gamma(\cdot; p, q)$

a shortest path between $p, q \in \mathcal{M}$

Tangent space $\mathcal{T}_p\mathcal{M}$ at p

with inner product $(\cdot, \cdot)_p$

Logarithmic map $\log_p q = \dot\gamma(0; p, q)$

"speed towards q"

Exponential map $\exp_p X = \gamma_{p,X}(1)$,

where $\gamma_{p,X}(0) = p$ and $\dot\gamma_{p,X}(0) = X$

Parallel transport $\mathsf{P}_{q\leftarrow p}Y$

from $\mathcal{T}_p\mathcal{M}$ along $\gamma(\cdot; p, q)$ to $\mathcal{T}_q\mathcal{M}$

Convexity

[Sakai 1996; Udrişte 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 $F\colon \mathcal{C} \to \overline{\mathbb{R}}$ 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.

Musical Isomorphisms

[Lee 2003]

The dual space $\mathcal{T}_p^*\mathcal{M}$ of a tangent space $\mathcal{T}_p\mathcal{M}$ is called cotangent space.
We denote by $\langle\cdot\,,\cdot\rangle$ the duality pairing.

We define the musical isomorphisms

- $\flat\colon \mathcal{T}_p\mathcal{M} \ni X \mapsto X^\flat \in \mathcal{T}_p^*\mathcal{M}$ via $\langle X^\flat\,, Y\rangle = (X\,,Y)_p$ for all $Y \in \mathcal{T}_p\mathcal{M}$
- $\sharp\colon \mathcal{T}_p^*\mathcal{M} \ni \xi \mapsto \xi^\sharp \in \mathcal{T}_p\mathcal{M}$ via $(\xi^\sharp\,, Y)_p = \langle\xi\,, Y\rangle$ for all $Y \in \mathcal{T}_p\mathcal{M}$.

$\Rightarrow$ inner product and parallel transport on/between $\mathcal{T}_p^*\mathcal{M}$

The Euclidean Fenchel Conjugate

We define the Fenchel conjugate $f^*\colon \mathbb{R}^n \to \overline{\mathbb{R}}$ of $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$ by

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

- interpretation: maximize the distance of $\xi^{\mathrm{T}}x$ to f

$\Rightarrow$ extremum seeking problem on 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

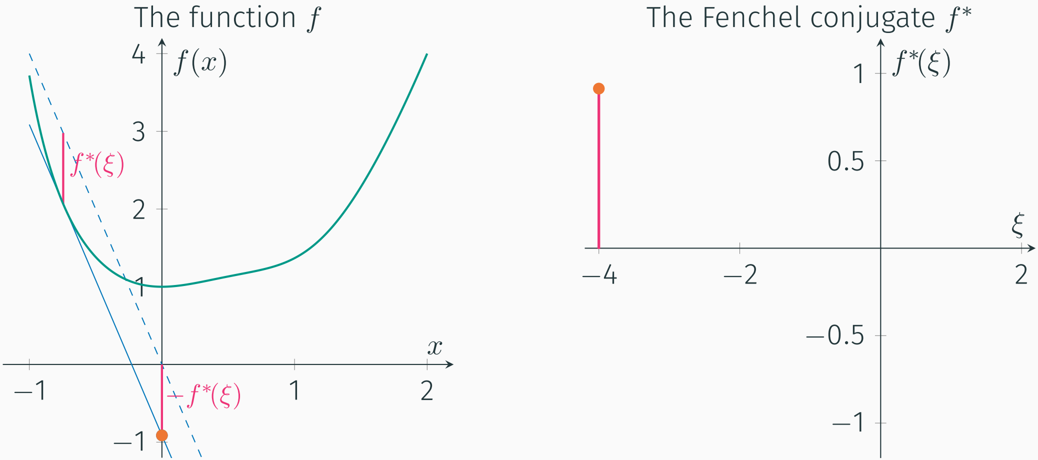
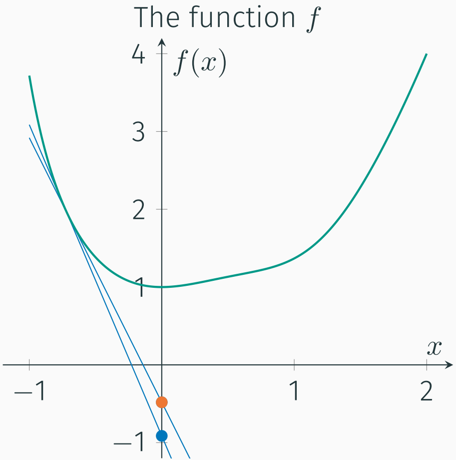


Illustration of the Fenchel Conjugate



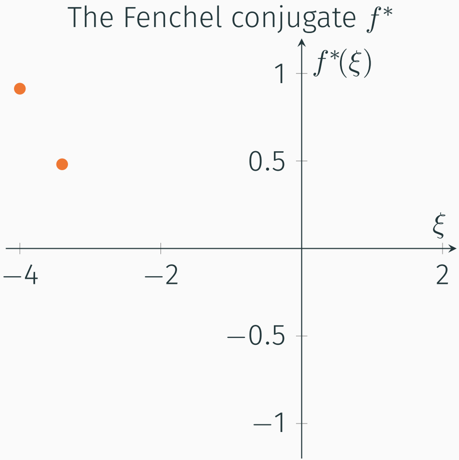
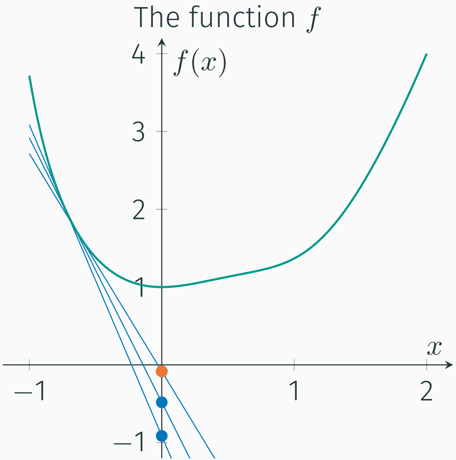


Illustration of the Fenchel Conjugate



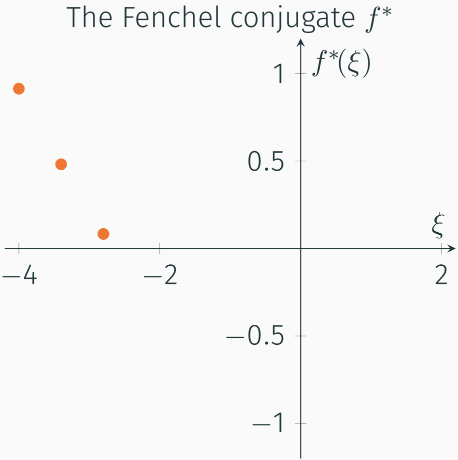
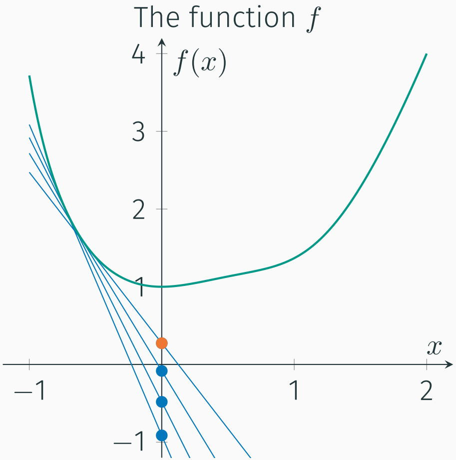


Illustration of the Fenchel Conjugate



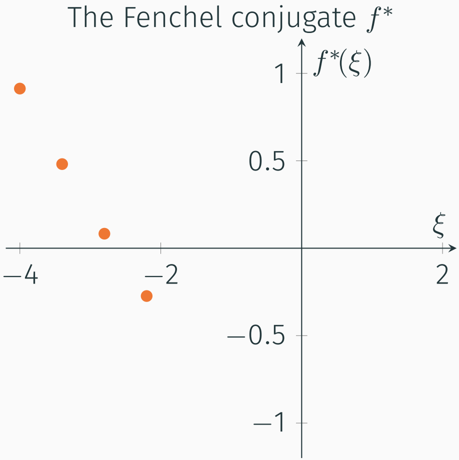
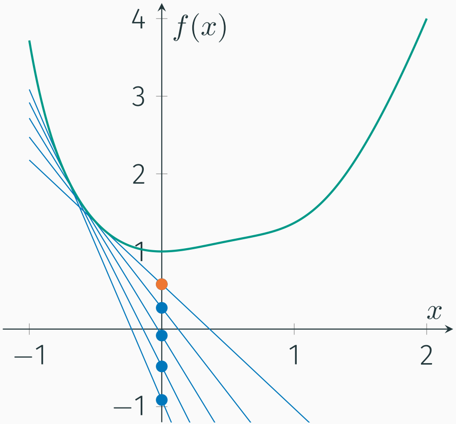


Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*

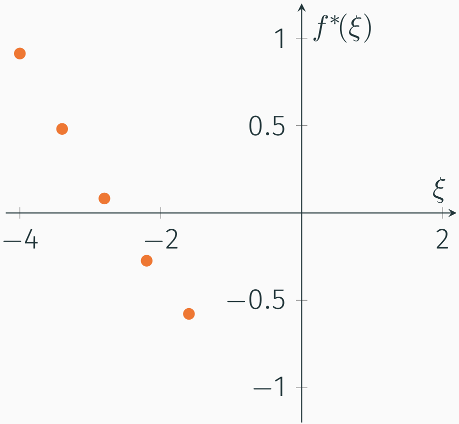
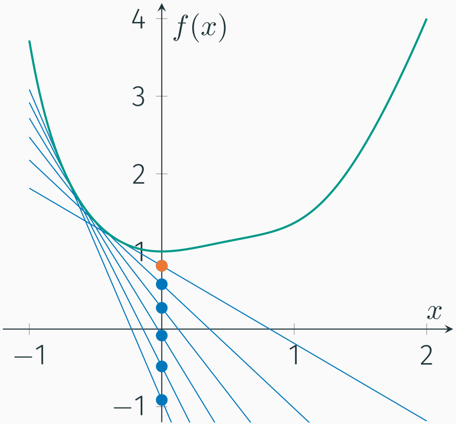


Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*

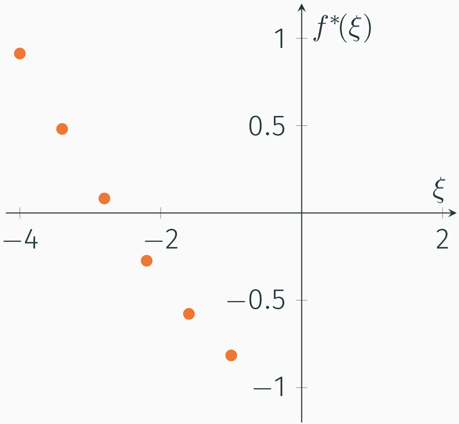
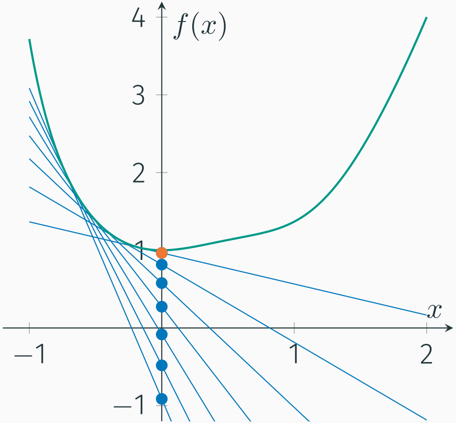


Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*

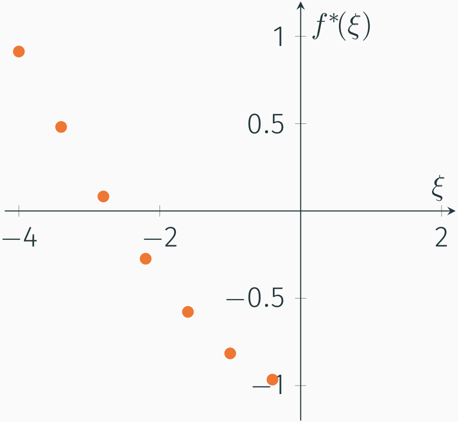
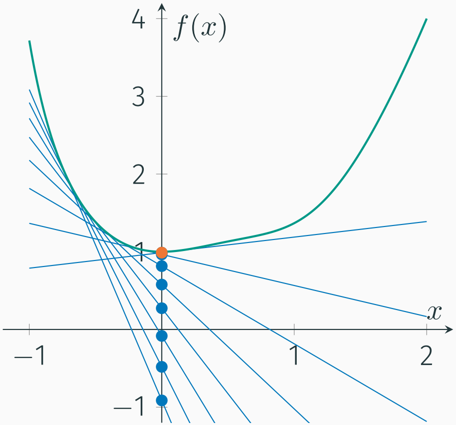


Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*

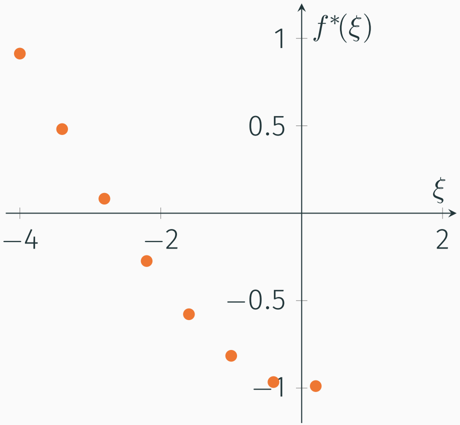
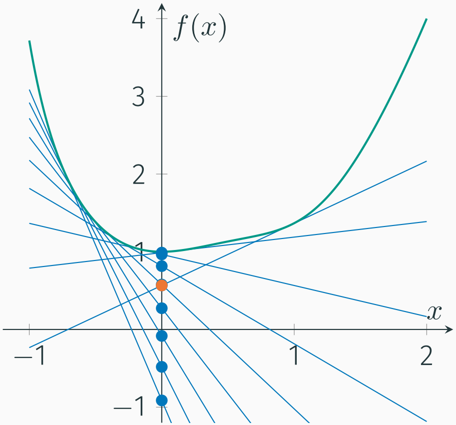


Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*

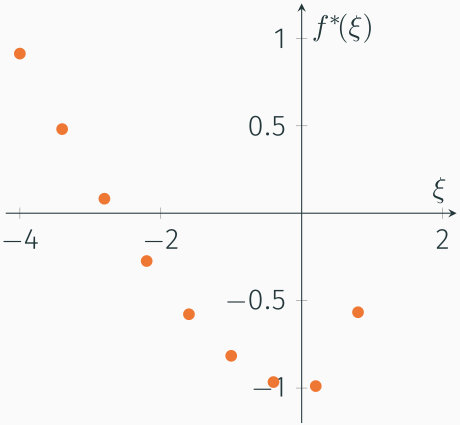
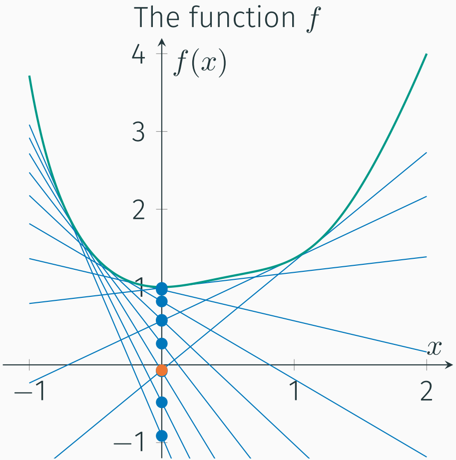


Illustration of the Fenchel Conjugate



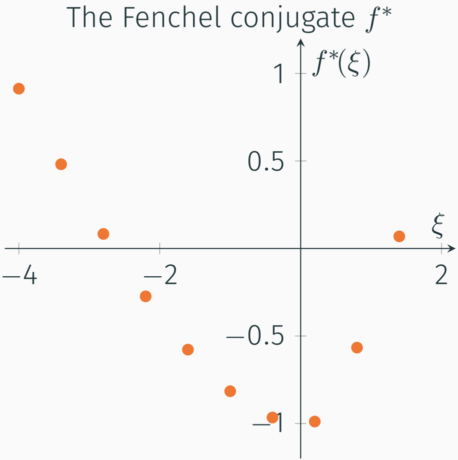
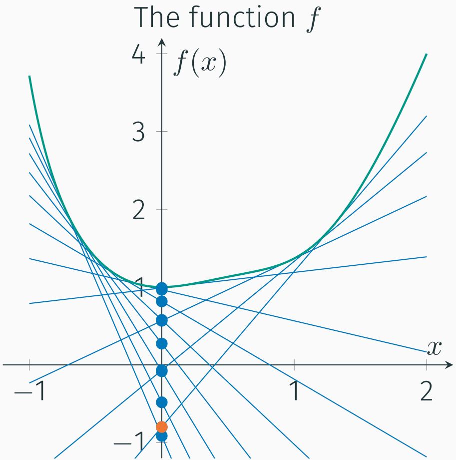


Illustration of the Fenchel Conjugate



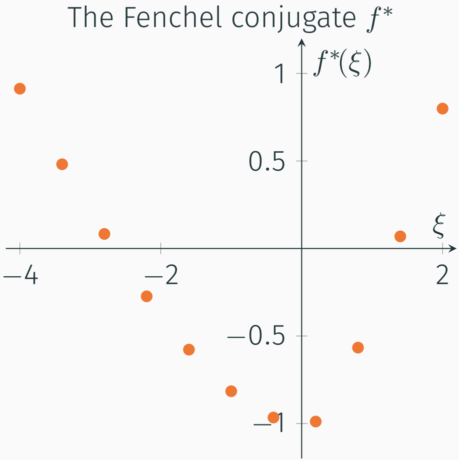
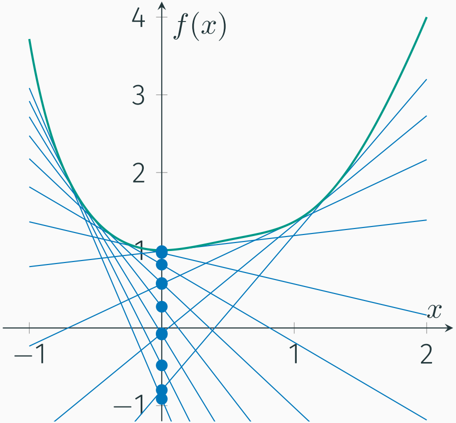
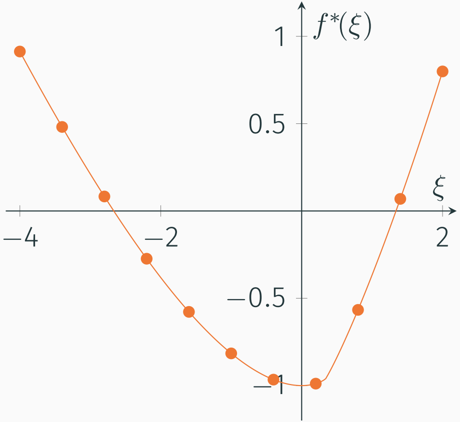


Illustration of the Fenchel Conjugate

The function f



The Fenchel conjugate f^*



Properties of the Fenchel Conjugate

[Rockafellar 1970]

- The Fenchel conjugate f^* is convex (even if f is not)
- If $f(x) \leq g(x)$ holds for all $x \in \mathbb{R}^n$ then $f^*(\xi) \geq g^*(\xi)$ holds for all $\xi \in \mathbb{R}^n$
- If $g(x) = f(x+b)$ for some $b \in \mathbb{R}$ holds for all $x \in \mathbb{R}^n$
  then $g^*(\xi) = f^*(\xi) - \xi^{\mathrm{T}} b$ holds for all $\xi \in \mathbb{R}^n$
- If $g(x) = \lambda f(x)$, for some $\lambda > 0$, holds for all $x \in \mathbb{R}^n$
  then $g^*(\xi) = \lambda f^*(\xi/\lambda)$ holds for all $\xi \in \mathbb{R}^n$
- f^{**} is the largest convex, lsc function with $f^{**} \leq f$
- especially the Fenchel–Moreau theorem:
  f convex, proper, lsc $\Rightarrow f^{**} = f$.

The Riemannian m–Fenchel Conjugate

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]
alternative approach: [Ahmadi Kakavandi and Amini 2010]

Idea: Introduce a point on $\mathcal{M}$ to "act as" 0.

Let $m \in \mathcal{C} \subset \mathcal{M}$ be given and $F\colon \mathcal{C} \to \overline{\mathbb{R}}$.
The m-Fenchel conjugate $F_m^*\colon \mathcal{T}_m^*\mathcal{M} \to \overline{\mathbb{R}}$ is defined by

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

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

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

$$F_{mm'}^{**}(p) = \sup_{\xi_{m'} \in \mathcal{T}_{m'}^*\mathcal{M}} \bigl\{\langle \xi_{m'} \,, \log_{m'} p\rangle - F_m^*(\mathsf{P}_{m\leftarrow m'}\xi_{m'})\bigr\}.$$

Properties of the m-Fenchel Conjugate

- F_m^* is convex on $\mathcal{T}_m^*\mathcal{M}$
- If $F(p) \leq G(p)$ holds for all $p \in \mathcal{C}$
  then $F_m^*(\xi_m) \geq G_m^*(\xi_m)$ holds for all $\xi_m \in \mathcal{T}_m^*\mathcal{M}$
- If $G(p) = F(p) + a$ for some $a \in \mathbb{R}$ holds for all $p \in \mathcal{C}$
  then $G_m^*(\xi_m) = F_m^*(\xi_m) - a$ holds for all $\xi_m \in \mathcal{T}_m^*\mathcal{M}$
- If $G(p) = \lambda F(p)$, for some $\lambda > 0$, holds for all $p \in \mathcal{C}$
  then $G_m^*(\xi_m) = \lambda F_m^*(\xi_m/\lambda)$ holds for all $\xi_m \in \mathcal{T}_m^*\mathcal{M}$
- It holds $F_{mm}^{**} \leq F$ on $\mathcal{C}$
- especially the Fenchel-Moreau theorem:
  If $F \circ \exp_m$ convex (on $\mathcal{T}_m\mathcal{M}$), proper, lsc, then $F_{mm}^{**} = F$ on $\mathcal{C}$.

Saddle Point Formulation

Let F be geodesically convex, $G \circ \exp_n$ be convex (on $\mathcal{T}_n\mathcal{N}$).

From

$$\min_{p\in\mathcal{C}} F(p) + G(\Lambda(p))$$

we derive the saddle point formulation for the n-Fenchel conjugate of G as

$$\min_{p\in\mathcal{C}} \max_{\xi_n\in\mathcal{T}_n^*\mathcal{N}} \langle \xi_n \,, \log_n \Lambda(p)\rangle + F(p) - G_n^*(\xi_n).$$

For Optimality Conditions and the Dual Prolem: What's Λ^*?

Approach. Linearization: [Valkonen 2014]

$$\Lambda(p) \approx \exp_{\Lambda(m)} D\Lambda(m)[\log_m p]$$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

Input: $p^{(0)} \in \mathbb{R}^d$, $\xi^{(0)} \in \mathbb{R}^d$, and parameters σ, τ, $\theta > 0$

$k \leftarrow 0$

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

while not converged **do**

$\quad \xi^{(k+1)} \leftarrow \operatorname{prox}_{\tau G^*}\big(\xi^{(k)} + \tau(\quad \Lambda(\bar{p}^{(k)})) \ \big)$

$\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Big(p^{(k)} \quad \big(-\sigma \quad \Lambda \quad {}^{*}\,\xi^{(k+1)}\big)^{\sharp} \quad \Big)$

$\quad \bar{p}^{(k+1)} \leftarrow p^{(k+1)} + \theta(p^{(k+1)} - p^{(k)})$

$\quad k \leftarrow k + 1$

end while

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

Input: $m, p^{(0)} \in \mathcal{C} \subset \mathcal{M}, n = \Lambda(m), \xi^{(0)} \in \mathbb{R}^d$, and parameters $\sigma, \tau, \theta > 0$

$k \leftarrow 0$

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

while not converged **do**

$\quad \xi^{(k+1)} \leftarrow \operatorname{prox}_{\tau G^*}\bigl(\xi^{(k)} + \tau(\quad \Lambda(\bar{p}^{(k)}))\ \bigr)$

$\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Bigl(p^{(k)} \quad \bigl(-\sigma \quad \Lambda \quad * \ \xi^{(k+1)}\bigr)^\sharp \quad \Bigr)$

$\quad \bar{p}^{(k+1)} \leftarrow p^{(k+1)} + \theta(p^{(k+1)} - p^{(k)})$

$\quad k \leftarrow k + 1$

end while

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

$k \leftarrow 0$

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

while not converged **do**

$\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\bigl(\xi_n^{(k)} + \tau(\qquad \Lambda(\bar{p}^{(k)}))\ \bigr)$

$\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Bigl(p^{(k)} \qquad \bigl(-\sigma \quad \Lambda \qquad {}^{*}\ \xi_n^{(k+1)}\bigr)^{\sharp} \qquad \Bigr)$

$\quad \bar{p}^{(k+1)} \leftarrow p^{(k+1)} + \theta(p^{(k+1)} - p^{(k)})$

$\quad k \leftarrow k + 1$

end while

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

$k \leftarrow 0$

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

while not converged **do**

$\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\bigl(\xi_n^{(k)} + \tau\bigl(\log_n \Lambda(\bar{p}^{(k)})\bigr)^\flat\bigr)$

$\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Bigl(p^{(k)} \qquad \bigl(-\sigma \quad \Lambda \qquad * \; \xi_n^{(k+1)}\bigr)^\sharp \qquad \Bigr)$

$\quad \bar{p}^{(k+1)} \leftarrow p^{(k+1)} + \theta(p^{(k+1)} - p^{(k)})$

$\quad k \leftarrow k + 1$

end while

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

1: $k \leftarrow 0$

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

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

4: $\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\big(\xi_n^{(k)} + \tau\big(\log_n \Lambda(\bar{p}^{(k)})\big)^\flat\big)$

5: $\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Big(p^{(k)} \qquad \big(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}]\big)^\sharp \qquad \Big)$

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

7: $\quad k \leftarrow k + 1$

8: **end while**

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

1: $k \leftarrow 0$

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

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

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

5: $p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Big(p^{(k)} \quad \big(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}]\big)^\sharp \Big)$

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

7: $k \leftarrow k + 1$

8: **end while**

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

1: $k \leftarrow 0$

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

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

4: $\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\bigl(\xi_n^{(k)} + \tau\bigl(\log_n \Lambda(\bar{p}^{(k)})\bigr)^\flat\bigr)$

5: $\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Bigl(p^{(k)} + \mathrm{P}_{p^{(k)}\leftarrow m}\bigl(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}]\bigr)^\sharp\Bigr)$

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

7: $\quad k \leftarrow k + 1$

8: **end while**

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

1: $k \leftarrow 0$

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

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

4: $\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\big(\xi_n^{(k)} + \tau\big(\log_n \Lambda(\bar{p}^{(k)})\big)^\flat\big)$

5: $\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Big(\exp_{p^{(k)}}\Big(\mathrm{P}_{p^{(k)}\leftarrow m}\big(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}]\big)^\sharp\Big)\Big)$

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

7: $\quad k \leftarrow k + 1$

8: **end while**

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

1: $k \leftarrow 0$

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

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

4: $\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\bigl(\xi_n^{(k)} + \tau\bigl(\log_n \Lambda(\bar{p}^{(k)})\bigr)^\flat\bigr)$

5: $\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Bigl(\exp_{p^{(k)}}\Bigl(\mathsf{P}_{p^{(k)}\leftarrow m}\bigl(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}]\bigr)^\sharp\Bigr)\Bigr)$

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

7: $\quad k \leftarrow k + 1$

8: **end while**

Output: $p^{(k)}$

The Exact Riemannian Chambolle–Pock Algorithm (eRCPA)

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

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

$k \leftarrow 0$

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

while not converged **do**

$\quad \xi_n^{(k+1)} \leftarrow \operatorname{prox}_{\tau G_n^*}\Big(\xi_n^{(k)} + \tau\big(\log_n \Lambda(\bar{p}^{(k)})\big)^\flat\Big)$

$\quad p^{(k+1)} \leftarrow \operatorname{prox}_{\sigma F}\Big(\exp_{p^{(k)}}\Big(\mathrm{P}_{p^{(k)} \leftarrow m}\big(-\sigma D\Lambda(m)^*[\xi_n^{(k+1)}]\big)^\sharp\Big)\Big)$

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

$\quad k \leftarrow k + 1$

end while

Output: $p^{(k)}$

Generalizations & Variants of the RCPA

Classically

[Chambolle and Pock 2011]

- change $\sigma = \sigma_k$, $\tau = \tau_k$, $\theta = \theta_k$ during the iterations
- introduce an acceleration γ
- relax dual $\bar{\xi}$ instead of primal $\bar{p}$ (switches lines 4 and 5)

Furthermore we

[RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

- introduce the lRCPA: linearize Λ, too, i. e.

$$\log_n \Lambda(\bar{p}^{(k)}) \quad \to \quad \mathsf{P}_{n\leftarrow\Lambda(m)} D\Lambda(m)[\log_m \bar{p}^{(k)}]$$

- choose $n \neq \Lambda(m)$ introduces a parallel transport

$$D\Lambda(m)^*[\xi_n^{(k+1)}] \quad \to \quad D\Lambda(m)^*[\mathsf{P}_{\Lambda(m)\leftarrow n}\xi_n^{(k+1)}]$$

- change $m = m^{(k)}$, $n = n^{(k)}$ during the iterations

Convergence of lRPCA

Theorem. [RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

Let $\mathcal{M}, \mathcal{N}$ be Hadamard, F be geodesically convex, and $G_n = G \circ \exp_n$ be convex. Assume that the linearized problem

$$\min_{p \in \mathcal{M}} \max_{\xi_n \in \mathcal{T}_n^* \mathcal{N}} \langle (D\Lambda(m))^* [\xi_n] \, , \log_m p \rangle + F(p) - G_n^*(\xi_n).$$

has a saddle point $(\widehat{p}, \widehat{\xi_n})$.
Choose σ, τ such that

$$\sigma\tau < \|D\Lambda(m)\|^2$$

and additionally a technical assumption holds. Then

1. the sequence $(p^{(k)}, \xi_n^{(k)})$ remains bounded,
2. there exists a saddle-point (p', ξ_n') such that $p^{(k)} \to p'$ and $\xi_n^{(k)} \to \xi_n'$.

2. Implementation in Manopt.jl

Implementing a Riemannian manifold

`MyManifold <: Manifold{𝔽}`

`ManifoldsBase.jl` introduces a manifold type with its field $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}, \mathbb{H}\}$ as parameter to provide an interface for implementing functions like

- `inner(M, p, X, Y)` for angles between tangent vectors,
- `exp(M, p, X)` and `log(M, p, q)`,
- more general: `retract(M, p, X, m)`, where `m` is a retraction method
- moving tangents: `vector_transport_to(M, p, X, q, t)`, where `t` is a transport method (e.g. `ParallelTransport()`)

for your manifold, which is a subtype of `Manifold`.

☺ mutating version `exp!(M, q, p, X)` works in place in `q`

➔ basis for generic algorithms working on any `Manifold`:
`norm(M,p,X)`, `geodesic(M, p, X)` and `shortest_geodesic(M, p, q)` are available with the above implemented.

`Manifolds.jl` – A Library of Manifolds

`Manifolds.jl` is a library of manifolds with a set tools to implement new ones.

Core feature: decorators for manifolds

- `M = SymmetricPositiveDefinite(n)` behaves as
- `MetricManifold(M, LinearAffineMetric())`
- `MetricManifold(M, LogEuclidean())` behaves as `M`, despite for `exp`, `log`, `dist`, `inner`.

⇒ semitransparent decorator pattern

Similarly: `(Abstract)EmbeddedManifold` & `GroupManifold(M, GroupAction)`

`Manifolds.jl` – A Library of Manifolds

`Manifolds.jl` is a library of manifolds with a set tools to implement new ones.

Core feature: decorators for manifolds

- `M = SymmetricPositiveDefinite(n)` behaves as
- `MetricManifold(M, LinearAffineMetric())`
- `MetricManifold(M, LogEuclidean())` behaves as `M`, despite for `exp`, `log`, `dist`, `inner`.

⇒ semitransparent decorator pattern

Similarly: `(Abstract)EmbeddedManifold` & `GroupManifold(M, GroupAction)`

We cover for example

- Euclidean
- Elliptope & Spectrahedron
- Fixed-rank Matrices
- (Generalized) Stiefel
- (Generalized) Grassmann
- Hyperbolic space
- Lorentz space
- Probability simplex
- Rotation
- Skew- & symmetric matrices
- (Array)Sphere & Circle
- Symm. Pos. Def.
- Torus
- …

`Manifolds.jl` – A Library of Manifolds

`Manifolds.jl` is a library of manifolds with a set tools to implement new ones.

Core feature: decorators for manifolds

- `M = SymmetricPositiveDefinite(n)` behaves as
- `MetricManifold(M, LinearAffineMetric())`
- `MetricManifold(M, LogEuclidean())` behaves as `M`, despite for `exp, log, dist, inner`.

⇒ semitransparent decorator pattern

Similarly: `(Abstract)EmbeddedManifold` & `GroupManifold(M, GroupAction)`

And the following constructors

- `PowerManifold(M,n1,n2, ... )` or short `M^(n1,n2, ... )`
- `ProductManifold(M,N, ... )` or short `M×N× ...`
- `TangentBundle(M)`
- `(Co)TangentSpaceAt(M,p)`

`Manopt.jl` – A framework for Optimization on Manifolds

`Manopt.jl` provides a unified framework for optimization on manifolds as well as a unified set of algorithms based on `ManifoldsBase.jl`, and hence for all manifolds from `Manifolds.jl`.

An algorithm usually has a high level interface, like

`gradient_descent(M, F, ∇F, x0)`

with usually a lot more keyword options.

Example. Use a certain retraction in gradient descent.

```
xOpt = gradient_descent(M, F, ∇F, x0;
    retraction_method = PolarRetraction(),
)
```

The Manopt family: `manoptjl.org`

`Manopt` in Matlab [N. Boumal et. al.]
`manopt.org`

`pymanopt` in Python [J. Townsend, N. Koep, S. Weichwald]
`pymanopt.org`

The Solver Framework

Internally an algorithm is based on a `Problem p` and `Options o`.
The problem usually stores the manifold in `p.M`.

Example.

- `GradientProblem p` is a problem having a `p.cost` and a `p.gradient`
- `GradientDescentOptions` store a current iterate, current gradient, a retraction method to use,

and the implementation requires

- `intialize_solver!(p,o)` to initialize values within `o`
- `step_solver!(p,o,i)` to implement the `i`th step
- `get_solver_result(o)` which returns the resulting minimizer.

Stopping Criteria – The `stopping_criterion=` keyword

A `StoppingCriterion` is a functor: a `struct` that is also a function `(problem, options, iteration) -> Boolean`, for example

- `stopAfterIteration(n)`, `stopAfter(time)`
- `stopWhenChangeLess(eps)`
- or more specific `stopWhenTrustRegionIsExceeded`
- and for multiple criteria: `stopWhenAny`, `stopWhenAll`

Example. Stop when $\|\nabla F(x^{(k)})\|$ is less then 10^{-6}

```
m = gradient_descent(M, F, ∇F, x0;
    stopping_criterion = stopWhenGradientNormLess(1e-6),
)
```

Similarly: `Linesearches`, e.g. as `stepsize = ArmijoLinesearch()`.

Debug & Record

Both `DebugOptions(o,A)`, `RecordOptions(o,A)` act as if they where just the `Options o` (decorator pattern), but execute additional print/store after every step.

Examples.

```
o = gradient_descent(M, F, ∇F, x0;
    debug = [:Iteration," | ", :x," | ", :Change," | ", :Cost,"\n",
        50, :Stop],
    record = [:Iteration, :Change, :Cost, :∇],
)
```

- use keywords for `:Iteration` display and the `:Stop`ing reason
- use keywords (`:Cost`) or fields (`:x`) of `Options o`
- `debug=` can be interleaved with strings and a number
- Similarly: Record certain values (or fields of the `Options`)

Available Solvers

- Conjugate Gradient Descent
- Cyclic Proximal Point
- Douglas-Rachford
- Gradient Descent
- Nelder Mead
- Particle Swarm
- Subgradient Method
- Riemannian Trust Regions

😊 high level interface, a default stopping criterion, debug, record, …

3. Numerical Example

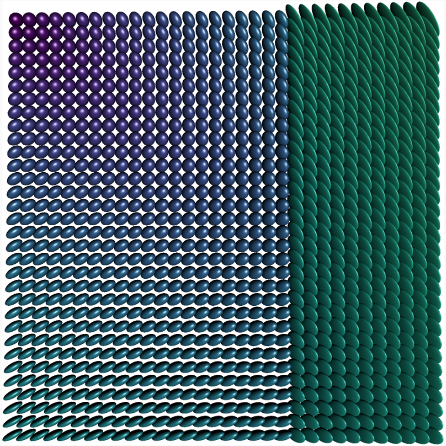
An lRCPA example

Let e.g. `F` be the cost, and with `S=SymmetricPositiveDefinite(3)` we set `M=S^(32,32)` and `N=TangentBundle(M^(32,32,2))`.

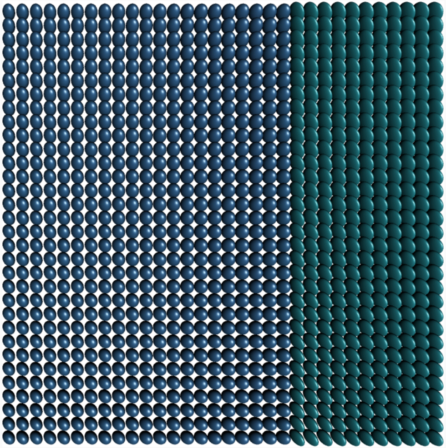
Then, having defined the remaining differentials, proximal maps and parameters we call

```
o = ChambollePock(M, N, F, x0, ξ0, m, n, prox_F, prox_G_dual, DΛ, AdjDΛ;
    primal_stepsize = σ, dual_stepsize = τ, relaxation = θ, acceleration = γ,
    relax = :dual,
    debug = [:Iteration," | ", :Cost, "\n", 10 , :Stop],
    record = [:Iteration, :Cost ],
    stoppingCriterion = sC,
    variant = :linearized,
)
x = get_solver_result(o)
```

Numerical Example for a $\mathcal{P}(3)$-valued Image



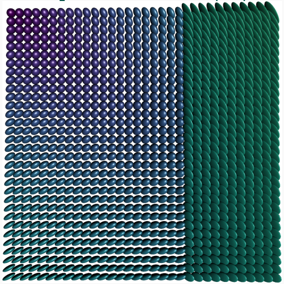
$\mathcal{P}(3)$-valued data.



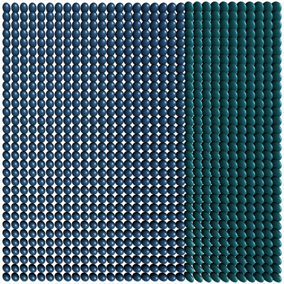
anisotropic TV, $\alpha = 6$.

- in each pixel we have a symmetric positive definite matrix
- Applications: denoising/inpainting e.g. of DT-MRI data

Numerical Example for a $\mathcal{P}(3)$-valued Image



$\mathcal{P}(3)$-valued data.



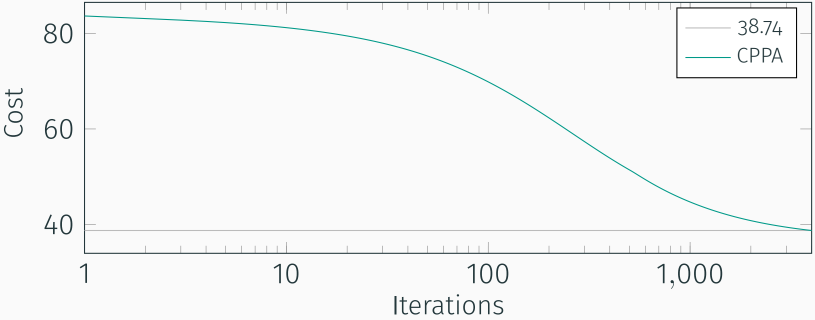
anisotropic TV, $\alpha = 6$.

Approach. CPPA as benchmark

[Bačák 2014; RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

	CPPA	PDRA	lRCPA
parameters	$\lambda_k = \frac{4}{k}$	$\lambda = 0.58$ $\beta = 0.93$	$\sigma = \tau = 0.4$ $\gamma = 0.2$, $m = I$
iterations	4000		
runtime	1235 s.		

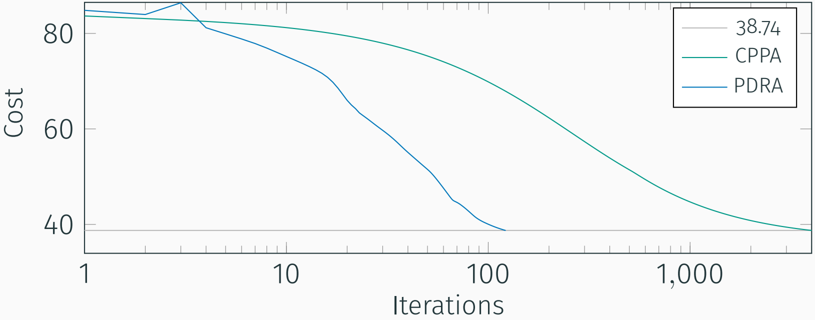
Numerical Example for a $\mathcal{P}(3)$-valued Image



Approach. CPPA as benchmark [Bačák 2014; RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

	CPPA	PDRA	lRCPA
parameters	$\lambda_k = \frac{4}{k}$	$\lambda = 0.58$ $\beta = 0.93$	$\sigma = \tau = 0.4$ $\gamma = 0.2,\ m = I$
iterations	4000		
runtime	1235 s.		

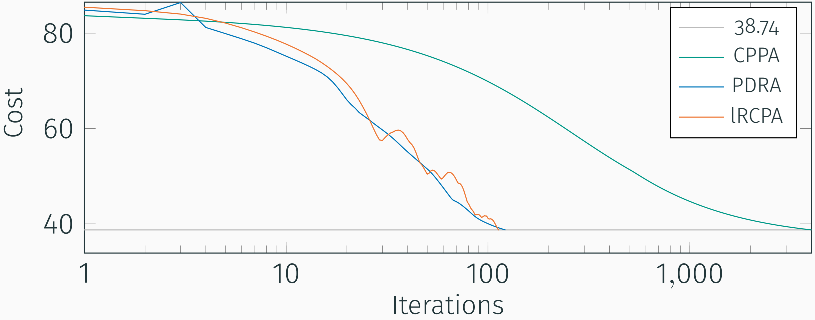
Numerical Example for a $\mathcal{P}(3)$-valued Image



Approach. CPPA as benchmark [Bačák 2014; RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

	CPPA	PDRA	lRCPA
parameters	$\lambda_k = \frac{4}{k}$	$\lambda = 0.58$ $\beta = 0.93$	$\sigma = \tau = 0.4$ $\gamma = 0.2,\ m = I$
iterations	4000	122	
runtime	1235 s.	380 s.	

Numerical Example for a $\mathcal{P}(3)$-valued Image



Approach. CPPA as benchmark [Bačák 2014; RB, Herzog, Silva Louzeiro, Tenbrinck, and Vidal-Núñez 2020]

	CPPA	PDRA	lRCPA
parameters	$\lambda_k = \frac{4}{k}$	$\lambda = 0.58$ $\beta = 0.93$	$\sigma = \tau = 0.4$ $\gamma = 0.2,\ m = I$
iterations	4000	122	**113**
runtime	1235 s.	380 s.	**96.1 s.**

4. Summary & Outlook

Summary

- Variational Methods for manifold valued data
- Splitting methods for efficient minimization
- A Riemannian Chambolle–Pock Algorithm
- implementation and examples with `Manopt.jl` in Julia.

What's next?

- derive a Fenchel duality which "works" with G (geodesically) convex
- Combine ML techniques with `ManifoldsBase.jl`
- Benchmark of manifold packages (together with S. Axen, M. Baran, K. Rzecki)
- constrained optimization problems and algorithms

Selected References

- Ahmadi Kakavandi, B. and M. Amini (Nov. 2010). "Duality and subdifferential for convex functions on complete metric spaces". In: *Nonlinear Analysis: Theory, Methods & Applications* 73.10, pp. 3450–3455. DOI: 10.1016/j.na.2010.07.033.
- RB, R. Herzog, M. Silva Louzeiro, D. Tenbrinck, and J. Vidal-Núñez (2020). *Fenchel duality theory and a primal-dual algorithm on Riemannian manifolds*. accepted for publication in Foundations of Computational Mathematics. arXiv: 1908.02022.
- RB, J. Persch, and G. Steidl (2016). "A parallel Douglas Rachford algorithm for minimizing ROF-like functionals on images with values in symmetric Hadamard manifolds". In: *SIAM Journal on Imaging Sciences* 9.4, pp. 901–937. DOI: 10.1137/15M1052858.
- Chambolle, A. and T. Pock (2011). "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of Mathematical Imaging and Vision* 40.1, pp. 120–145. ISSN: 0924-9907. DOI: 10.1007/s10851-010-0251-1.

- Manopt.jl: `https://manoptjl.org`
- Manifolds.jl `https://juliamanifolds.github.io/Manifolds.jl/stable/`