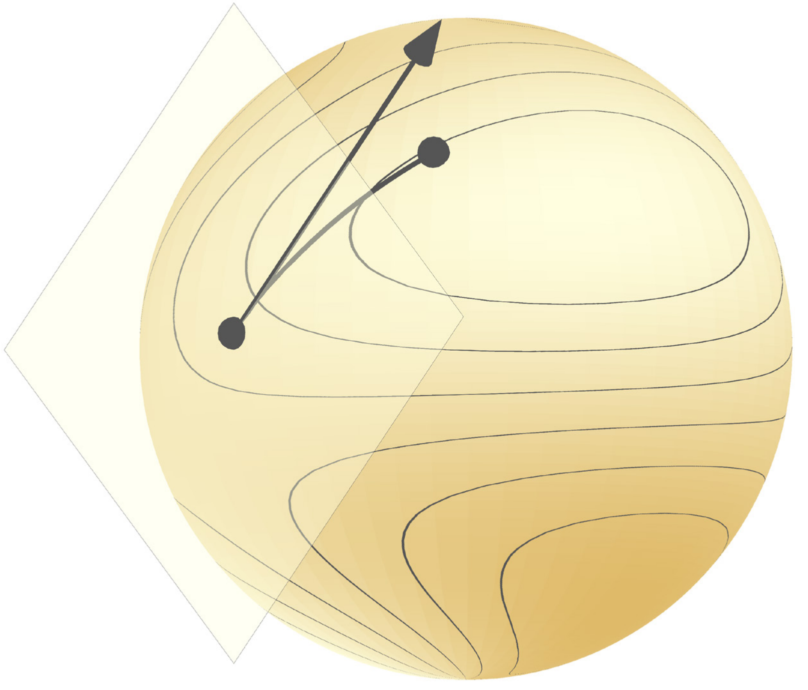
Riemannian optimization software and applications

TMS Workshop on

Foundations of Numerical Differential Geometry, May 7, 2024

Nicolas Boumal – chair of continuous optimization

Institute of Mathematics, EPFL



Step 0 in optimization

It starts with a set S and a function $f: S \to \mathbf{R}$. We want to compute:

$$\min_{x \in S} f(x)$$

These bare objects fully specify the problem.

Any additional structure on S and f may (and should) be exploited for algorithmic purposes but is not part of the problem.

Classical unconstrained optimization

The search space *is* a linear space, e.g., $S = \mathbf{R}^n$:

$$\min_{x \in \mathbf{R}^n} f(x)$$

We can *choose* to turn $\mathbf{R}^n$ into a Euclidean space: $\langle u, v \rangle = u^\top v$.

If f is differentiable, we have a gradient $\mathrm{grad} f$ and Hessian $\mathrm{Hess} f$.

We can build algorithms with them: gradient descent, Newton's...

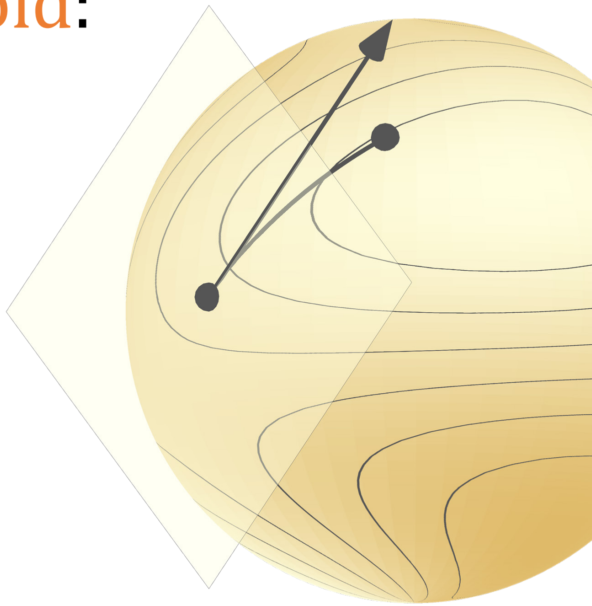
$$\langle \mathrm{grad} f(x), v \rangle = \mathrm{D}f(x)[v] = \lim_{t \to 0} \frac{f(x + tv) - f(x)}{t}$$

$$\mathrm{Hess} f(x)[v] = \mathrm{D}(\mathrm{grad} f)(x)[v] = \lim_{t \to 0} \frac{\mathrm{grad} f(x + tv) - \mathrm{grad} f(x)}{t}$$

Optimization on manifolds

We target applications where $S = \mathcal{M}$ *is* a smooth manifold:

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



We can *choose* to turn $\mathcal{M}$ into a Riemannian manifold.

If f is differentiable, we have a Riemannian gradient and Hessian.
We can build algorithms with them: gradient descent, Newton's...

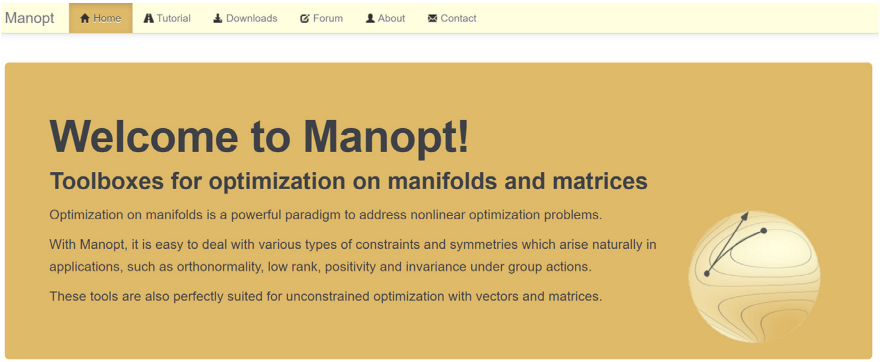
Manopt provides manifolds, solvers, tools

Manopt is a family of toolboxes for Riemannian optimization.

Go to manopt.org, pymanopt.org or manoptjl.org for code and help.

Matlab example for $\min_{\|x\|=1} x^\top A x$:

```
problem.M = spherefactory(n);
problem.cost = @(x) x'*A*x;
problem.egrad = @(x) 2*A*x;
x = trustregions(problem);
```



With Bamdev Mishra,
P.-A. Absil & R. Sepulchre

Lead by J. Townsend,
N. Koep & S. Weichwald

Lead by
Ronny Bergmann

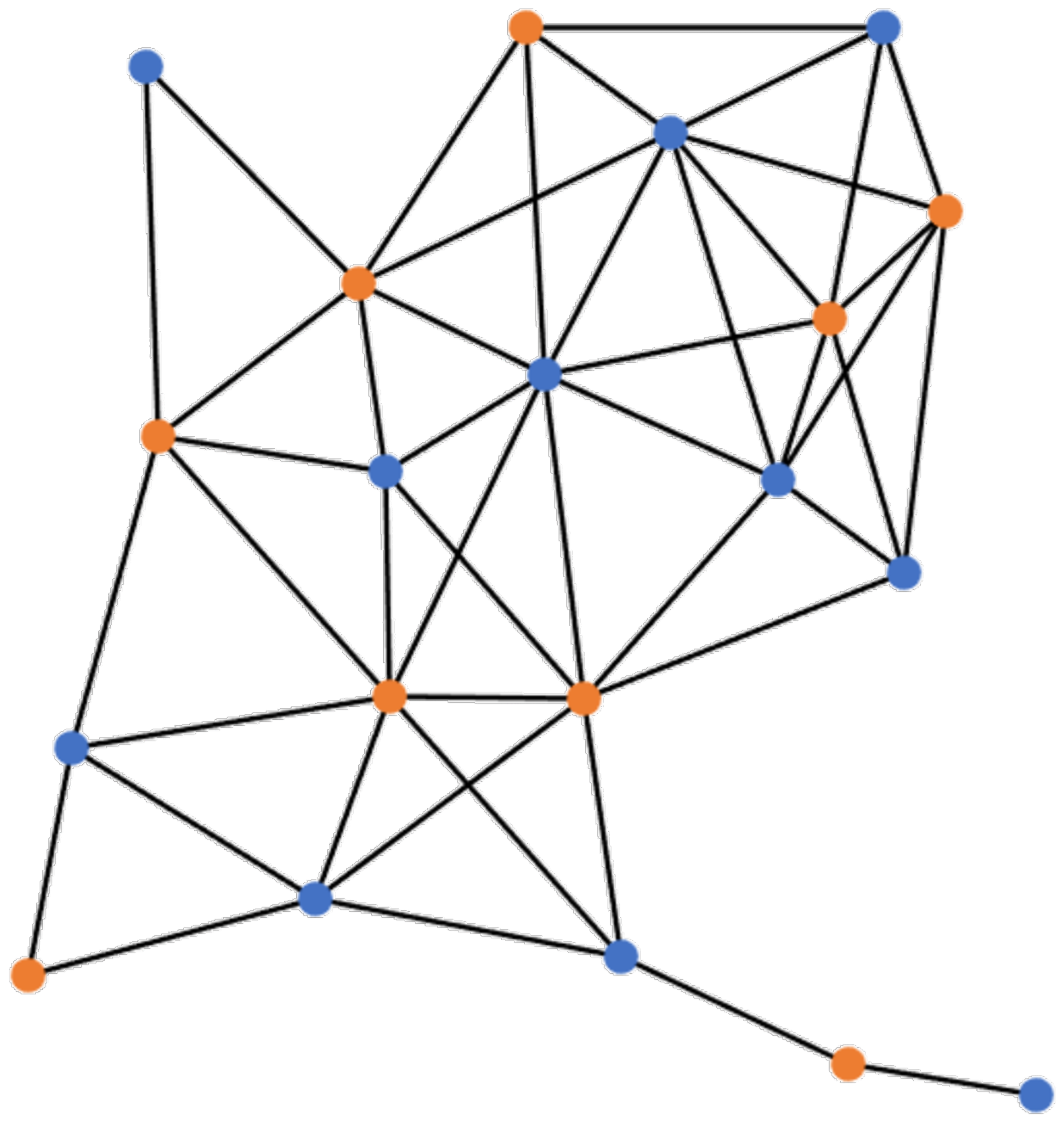
Example 1: Max-Cut

Input:

An undirected graph.

Output:

Vertex labels ($+1$, -1) so that as many edges as possible connect different labels.



Max-Cut

Input:

An undirected graph:

adjacency matrix A.

Output:

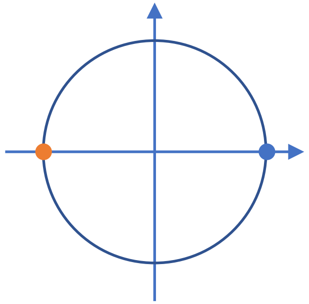
Vertex labels $x_i \in \{+1, -1\}$

so that as many edges

as possible connect

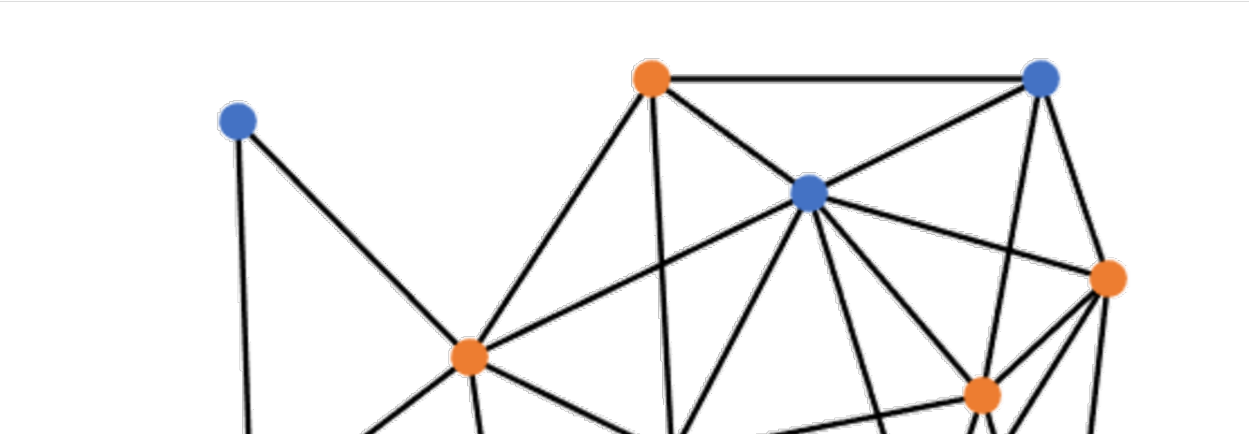
different labels.

$$\min_{x_1,\ldots,x_n} \sum_{ij} a_{ij} x_i x_j \quad \text{s.t.} \quad x_i \in \{\pm 1\}$$

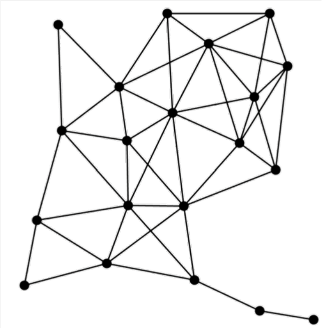


Relax the dimension:

Let x_i be unit-norm in $\mathbf{R}^p$.



Max-Cut via relaxation to spheres in Manopt



With adjacency matrix $A \in \mathbf{R}^{n\times n}$, want:

$$\min_{x_1,\ldots,x_n \in \mathbf{R}^p} \sum_{ij} a_{ij} x_i^\top x_j \quad \text{s.t.} \quad \|x_i\| = 1 \ \forall i$$

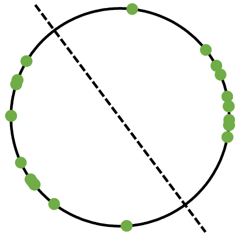
The manifold is a product of n spheres:

$$\mathcal{M} = \{x \in \mathbf{R}^p : \|x\| = 1\}^n$$

$$\equiv \left\{X \in \mathbf{R}^{p\times n} : \left\|X_{:,i}\right\| = 1 \ \forall i\right\}$$

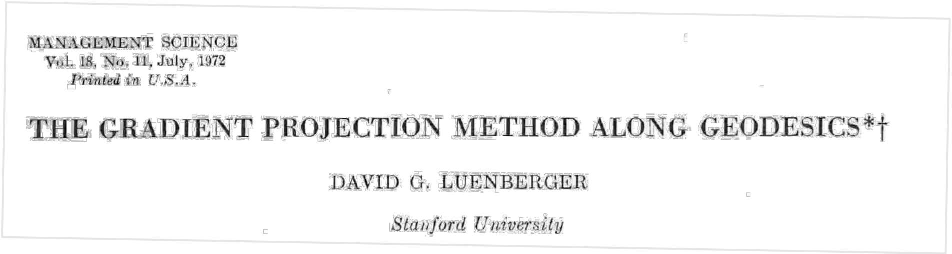
Called the oblique manifold.

```
data = load('graph20.mat');
A = data.A; n = data.n;

p = 2;
problem.M = obliquefactory(p, n);
problem.cost = @(X) sum((X*A) .* X, 'all');
problem.egrad = @(X) 2*X*A;
problem.ehess = @(X, Xdot) 2*Xdot*A;

X = trustregions(problem);

s = sign(X'*randn(p, 1));
    %random rounding
```



Fifty years

Proposed by Luenberger in 1972.



MANAGEMENT SCIENCE
Vol. 18, No. 11, July, 1972
Printed in U.S.A.

THE GRADIENT PROJECTION METHOD ALONG GEODESICS*†

DAVID G. LUENBERGER

Stanford University

Practical since the 1990s
with numerical linear algebra.

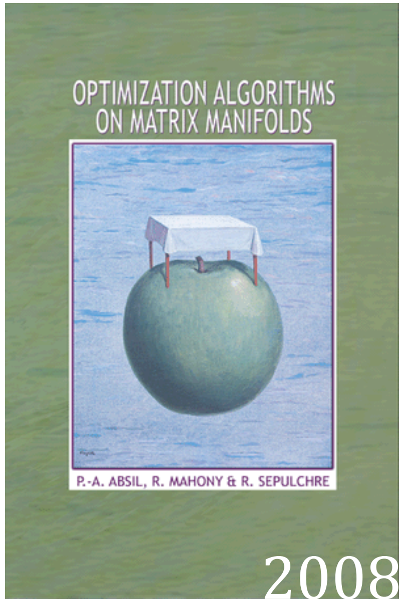
SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 303–353

© 1998 Society for Industrial and Applied Mathematics

THE GEOMETRY OF ALGORITHMS WITH ORTHOGONALITY CONSTRAINTS*

ALAN EDELMAN†, TOMÁS A. ARIAS‡, AND STEVEN T. SMITH§

Popularized in the 2010s
by Absil, Mahony & Sepulchre's book.



2008

Becoming mainstream now.

How do manifolds arise in optimization?

Linear spaces

$\mathbf{R}^n, \mathbf{R}^{n\times m}$

Orthonormality

Spheres, Stiefel, rotations, Grassmann

Positivity

Simplex, positive definite matrices

Rank

Matrices, tensors

Symmetry

Quotient manifolds

Lifts/parameterizations

arXiv:2207.03512, with E. Levin & J. Kileel

Products

$\mathcal{M} \times \mathcal{N}$

How do you “put” a manifold

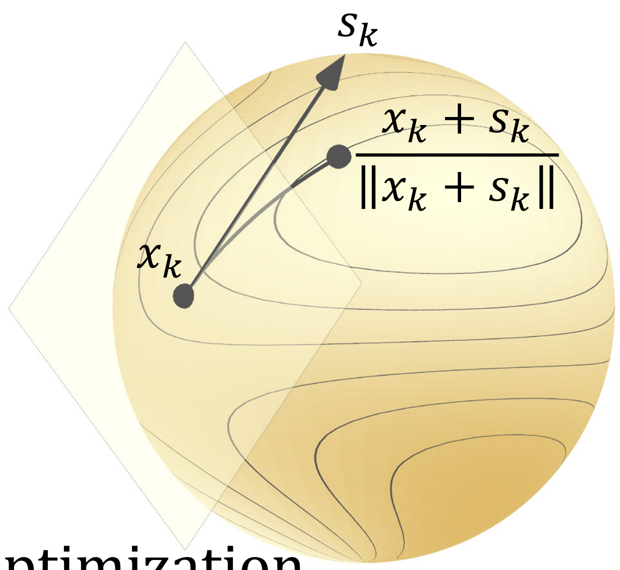
and those other tools

in a computer?

TMS Workshop on

Foundations of Numerical Differential Geometry

What do we need?



$$\min_x f(x)$$

	Euclidean optimization	Riemannian optimization
Basic step:	$x_{k+1} = x_k + s_k$	$x_{k+1} = R_{x_k}(s_k)$ (retraction)
Gradient descent:	$s_k = -\alpha_k \mathrm{grad} f(x_k)$	same, with Riemannian gradient
Newton's method:	$\mathrm{Hess} f(x_k)[s_k] = -\mathrm{grad} f(x_k)$	and Riemannian Hessian.

(Fancier algorithms involve more substantial differences, especially in analysis.)

These are the foundations.

Hess f

Connections $\nabla, \frac{\mathrm{D}}{\mathrm{d}t}$

grad f

Riemannian metric $\langle u, v \rangle_x$

Vector fields

Retractions

$\mathrm{D}F(x)[v]$

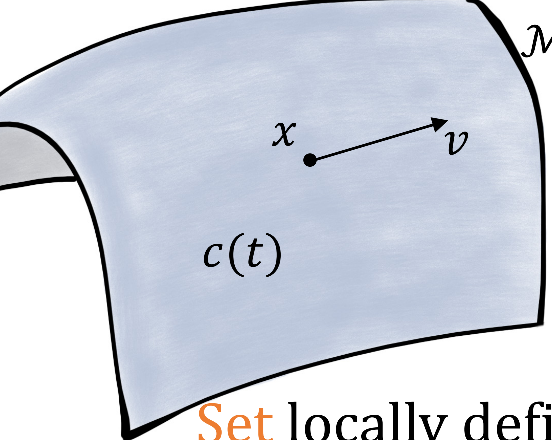
Tangent bundle $\mathrm{T}\mathcal{M}$

What is a smooth function?

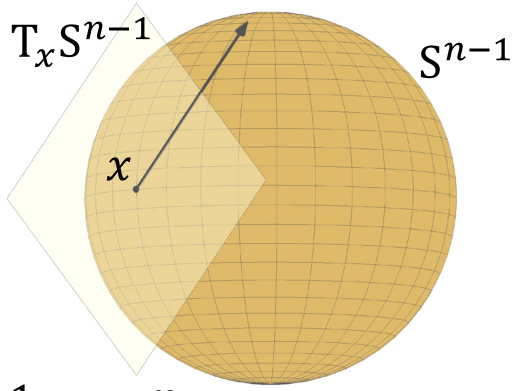
What is a tangent vector?

What is a smooth set?

This crash course:
Riemannian submanifolds
of linear spaces.



Submanifolds of $\mathbf{R}^n$



Set locally defined by (good) equations:

$$\mathcal{M} = \{x \in \mathbf{R}^n : h(x) = 0\}$$

Tangent space at x is $\ker \mathrm{D}h(x)$

Interpretations:

1. Linearize $h(x+v) \approx h(x) + \mathrm{D}h(x)[v]$
2. Curves: $c(0) = x \;\; \Rightarrow \;\; c'(0) \in \mathrm{T}_x\mathcal{M}$

Functions: $f = \bar{f}|_{\mathcal{M}}$ smooth iff $\bar{f}$ smooth

Derivative: $\mathrm{D}f(x)[v] = (f \circ c)'(0) = \mathrm{D}\bar{f}(x)[v]$

Example: the unit sphere S^{n-1} in $\mathbf{R}^n$

$$h(x) = x^\top x - 1$$

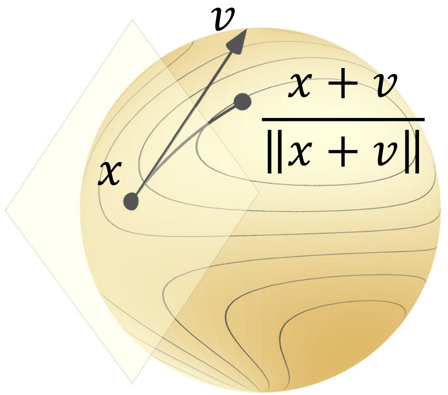
$$\mathrm{D}h(x)[v] = v^\top x + x^\top v$$

$$\mathrm{T}_x\mathrm{S}^{n-1} = \{v \in \mathbf{R}^n : x^\top v = 0\}$$

Any smooth $\bar{f}$ on $\mathbf{R}^n$ is still smooth if you restrict it to S^{n-1}. All smooth f are so.

Differentiate as usual, only on $\mathrm{T}_x\mathrm{S}^{n-1}$.

Retractions, gradients and Hessians



A retraction "smoothly" generates a curve

$$c(t) = R_x(tv)$$

such that $c(0) = x$ and $c'(0) = v$.

The Riemannian gradient of $f\colon \mathcal{M} \to \mathbf{R}$ at x is a tangent vector:

$$\mathrm{grad} f(x) = \mathrm{Proj}_x\left(\mathrm{grad} \bar{f}(x)\right)$$

$\mathrm{Hess} f(x)[v] = \mathrm{Proj}_x(\mathrm{D}\,\mathrm{grad} f(x)[v])$

Example on a sphere:

$$R_x(tv) = \frac{x + tv}{\|x + tv\|}$$

Inner product on $\mathbf{R}^n$: $\langle u, v\rangle = u^\top v$

Same inner product on each tangent space.

Let $\bar{f}(x) = \frac{1}{2}x^\top A x$. Then $\mathrm{grad} \bar{f}(x) = Ax$.

So $\mathrm{grad} f(x) = (I_n - xx^\top)Ax$

$\mathrm{Hess} f(x)[v] = \mathrm{Proj}_x(Av - (x^\top A x)v)$

In code, a manifold is a bunch of functions

Example: stripped down and simplified `spherefactory`

```
function M = spherefactory(n)
    M.name = @() sprintf('Sphere S^%d', n-1);
    M.dim = @() n-1;
    M.inner = @(x, u, v) u'*v;
    M.norm = @(x, u) norm(u);
    M.dist = @(x, y) real(2*asin(.5*norm(x - y)));
```

```
M.exp = @exponential;
M.retr = @(x, u) (x+u)/norm(x+u);
M.invretr = @inverse_retraction;
M.log = @logarithm;
M.hash = @(x) ['z' hashmd5(x)];
M.rand = @() normalize(randn(n, 1));
));
```

```
function M = spherefactory(n)
  M.inner = @(x, u, v) u'*v;
  M.proj = @(x, u) u - x*(x'*u);
  M.egrad2rgrad = M.proj;
  M.ehess2rhess = @(x, egrad, ehess, u) ...
                  M.proj(x, ehess - (x'*egrad)*u);
  M.retr = @(x, u) (x+u)/norm(x+u);
```

Example 2: Synchronization

See this paper: arxiv.org/abs/2312.10794

$\varphi(t) = e^{\beta t}$

$\max f(X) = \sum_{ij} \varphi\left(x_i^\top x_j\right)$

$\|x_1\| = \cdots = \|x_n\| = 1$

Let's go to Matlab.

A MATHEMATICAL PERSPECTIVE ON TRANSFORMERS

BORJAN GESHKOVSKI, CYRIL LETROUIT, YURY POLYANSKIY, AND PHILIPPE RIGOLLET

Remark 3.7. *Let us briefly sketch the particle version of the Wasserstein gradient flow* (3.8). *When* $\mu(t) = \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i(t)}$, *the interaction energy* (3.5) *takes the form*

$$\mathsf{E}_\beta(X) = \frac{1}{2\beta n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} e^{\beta\langle x_i, x_j\rangle}$$

where $X = (x_1, \ldots, x_n) \in (\mathbb{S}^{d-1})^n$. *Denoting by* ∇_X *the gradient associated to the standard Riemannian metric on* $(\mathbb{S}^{d-1})^n$, *we get the dynamics*

$$\dot{X}(t) = n\nabla_X \mathsf{E}_\beta(X(t)). \tag{3.11}$$

Indeed, the gradient on $(\mathbb{S}^{d-1})^n$ *is simply* $\nabla = (\partial_1, \ldots, \partial_n)$ *where* ∂_i *is the gradient in* $\mathbb{S}^{d-1}$ *acting on the i-th copy in* $(\mathbb{S}^{d-1})^n$. *Therefore*

$$\partial_i \mathsf{E}_\beta(X(t)) = \frac{1}{\beta n^2} \sum_{j=1}^{n} \mathbf{P}_{x_i(t)}\left(e^{\beta\langle x_i(t), x_j(t)\rangle} \beta x_j(t)\right) = \frac{1}{n}\dot{x}_i(t)$$

Software, book, lectures, slides

Manopt software packages

manopt.org pymanopt.org manoptjl.org

Matlab with Bamdev Mishra, P.-A. Absil, R. Sepulchre++

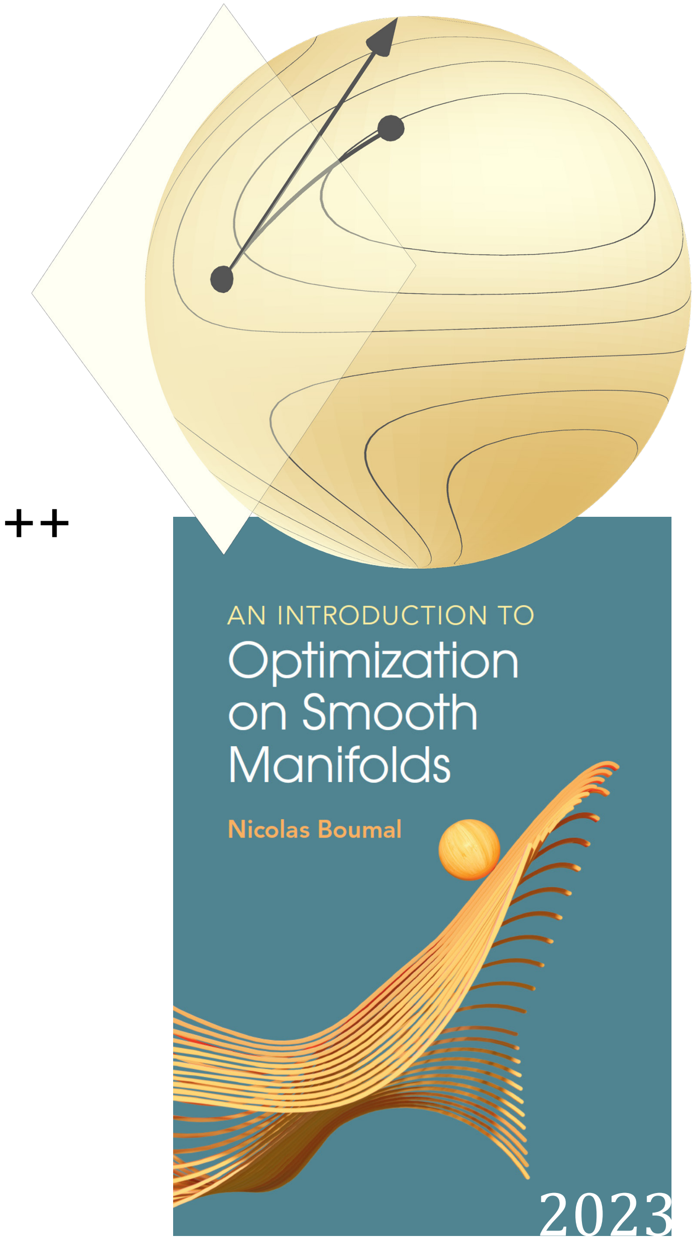
Julia by Ronny Bergmann++

Python by James Townsend, Niklas Koep and Sebastian Weichwald++

Book (pdf, lecture material, videos) and tutorial slides

nicolasboumal.net/book

nicolasboumal.net/SIAMOP23



Many thanks to Cambridge University Press, who agreed for me to keep the preprint freely available online.