

Manopt

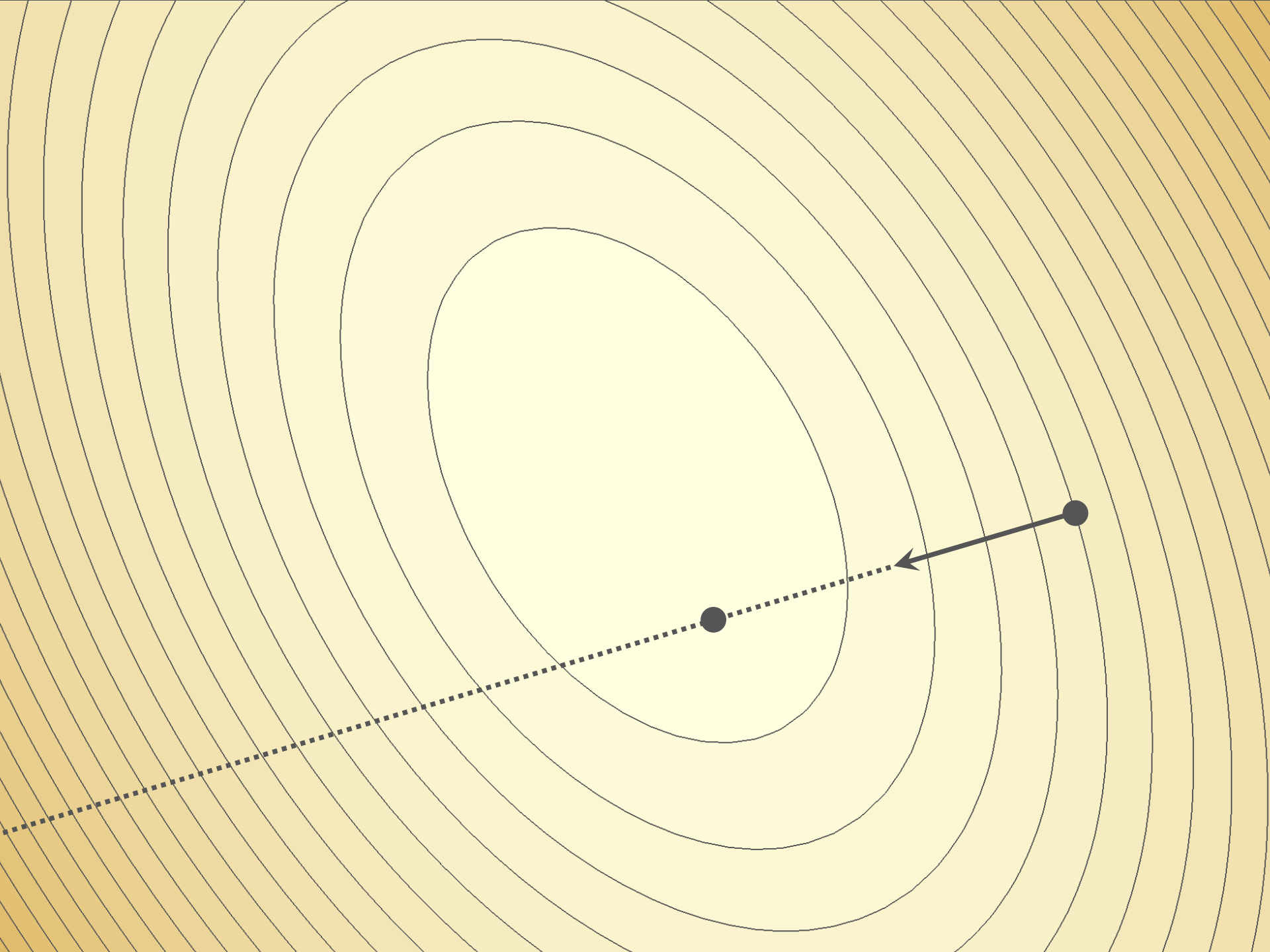
A Matlab toolbox to make optimization on manifolds feel as simple as unconstrained optimization

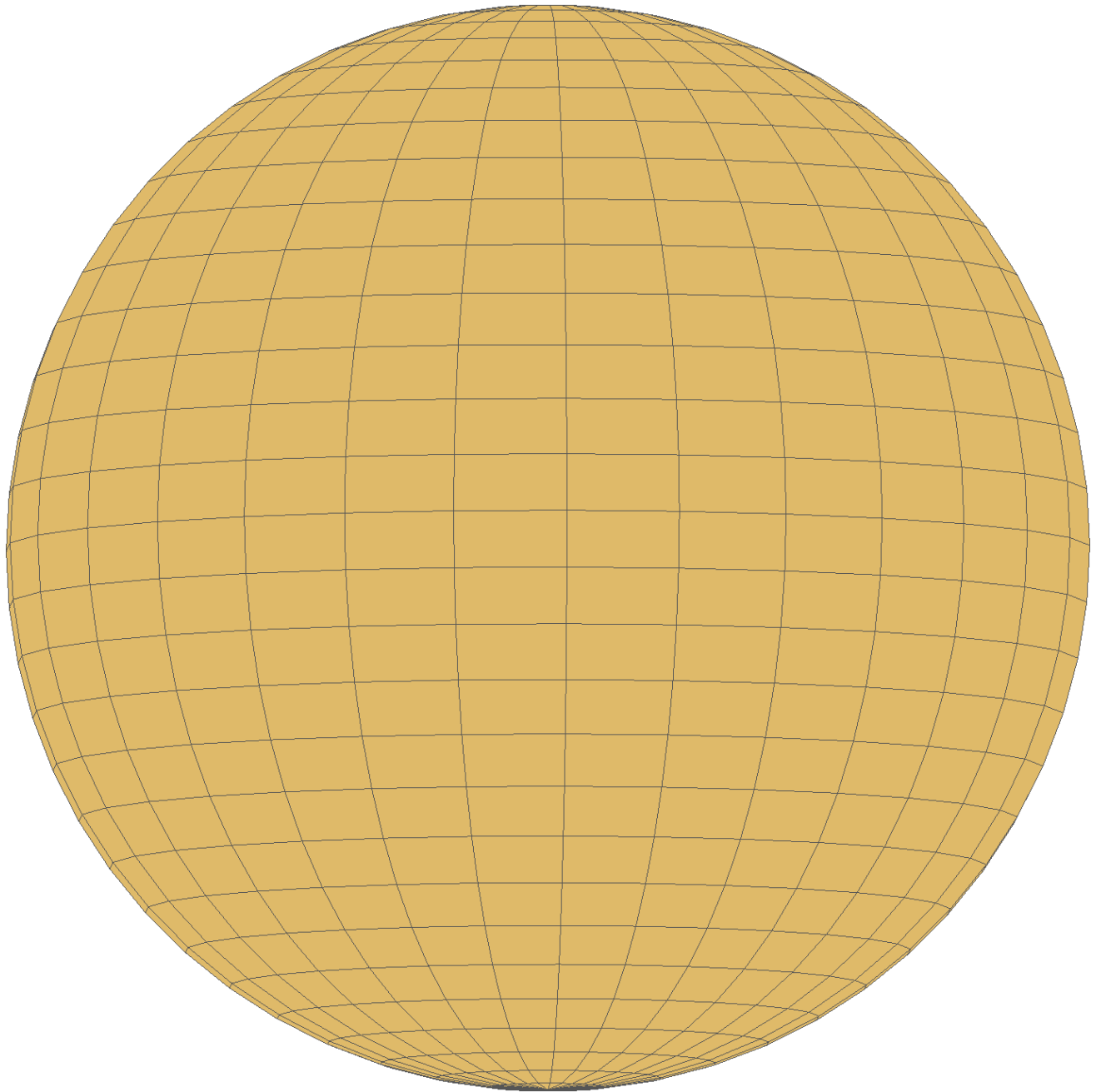
A project of the RANSO group

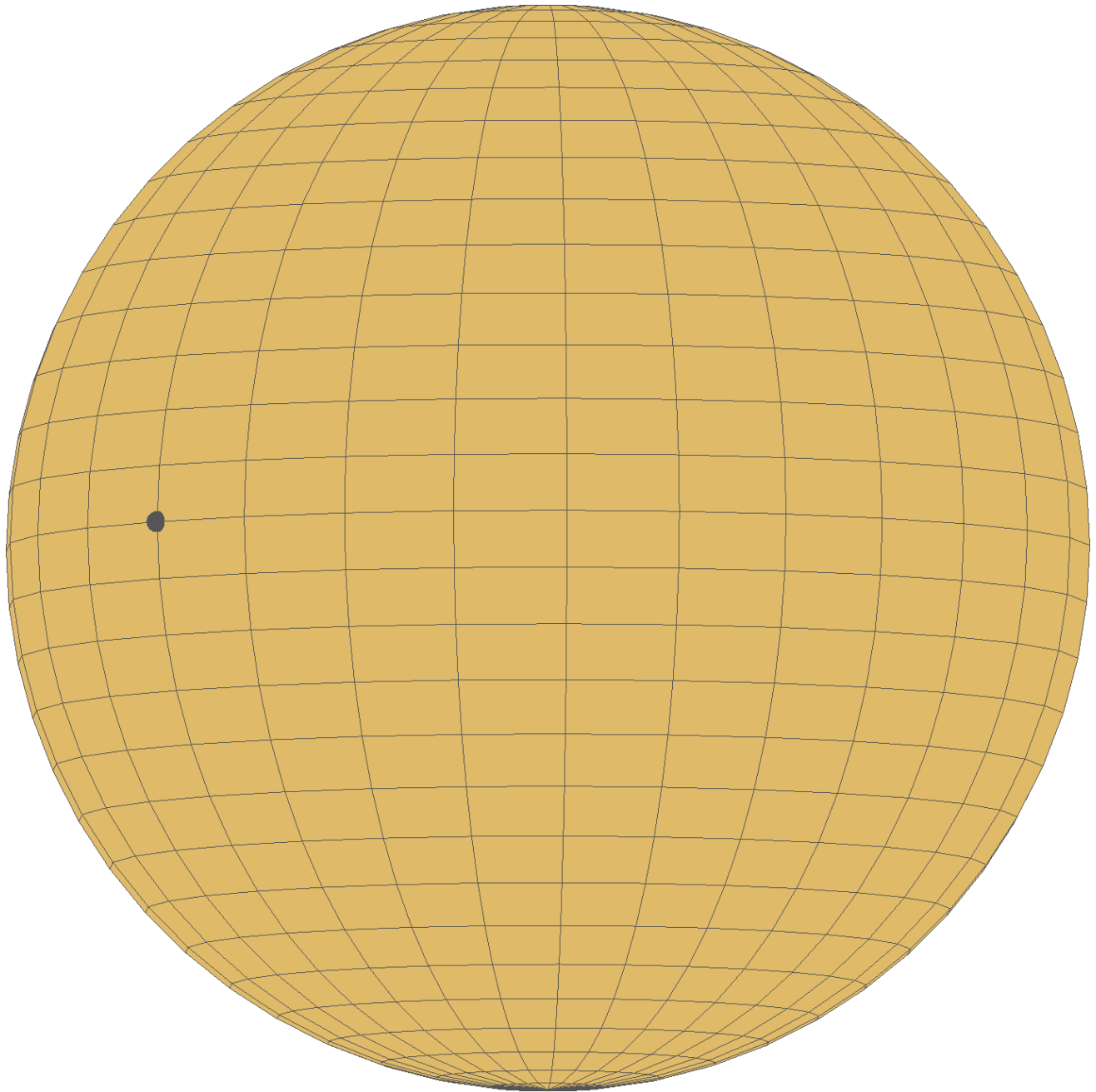
Nicolas Boumal and Bamdev Mishra

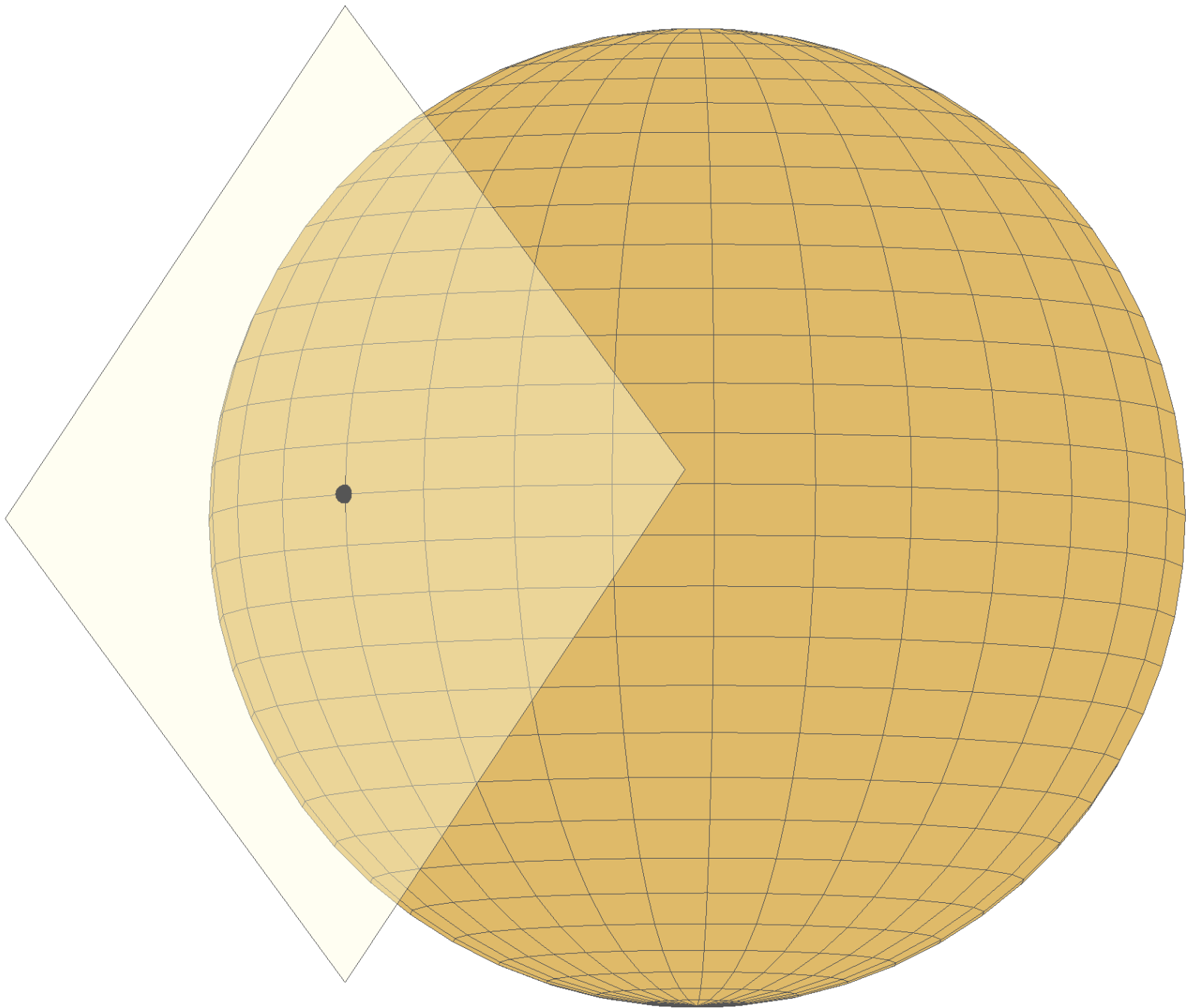
P.-A. Absil, Y. Nesterov and R. Sepulchre

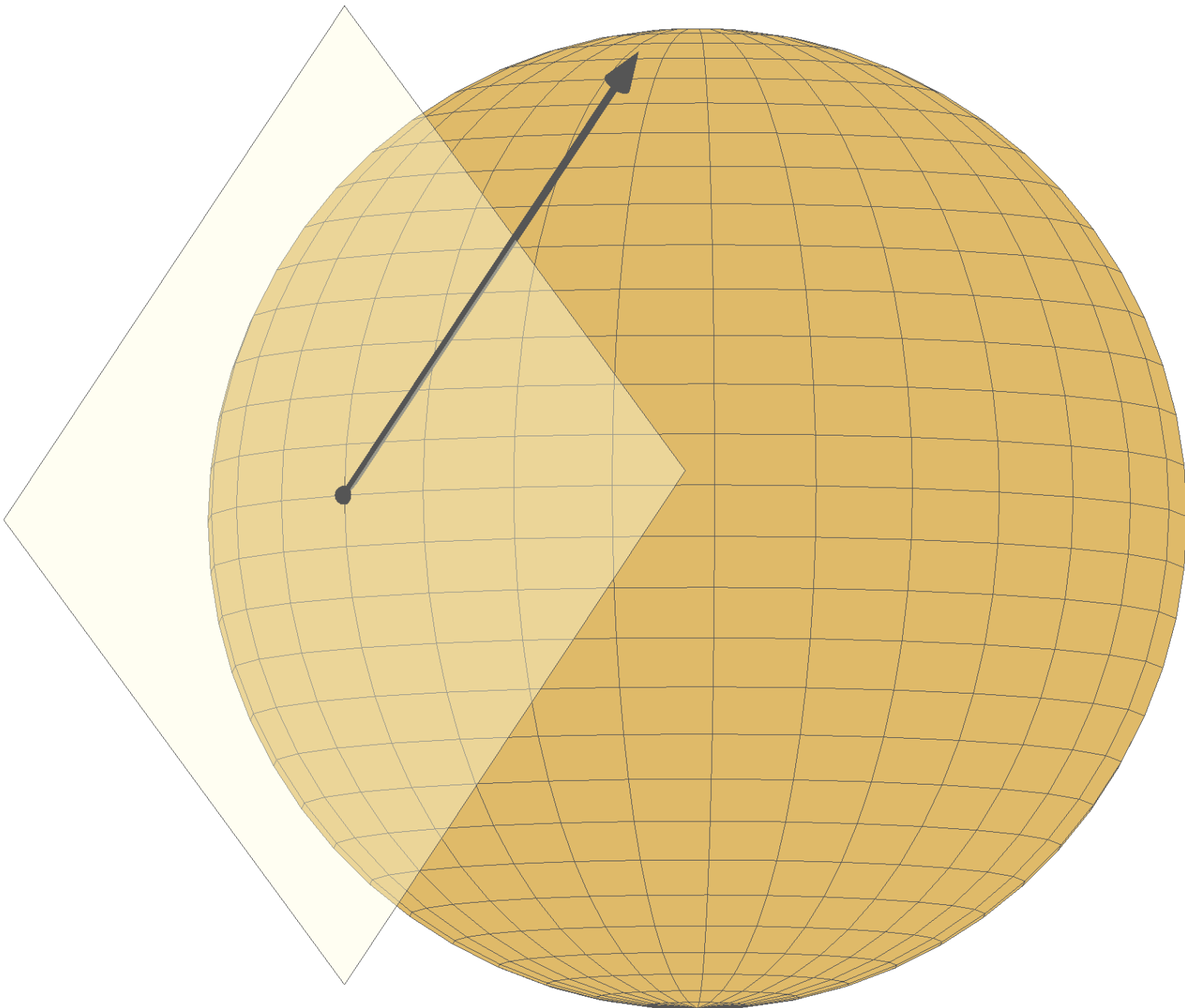
What is the minimal framework you need for steepest descent optimization?

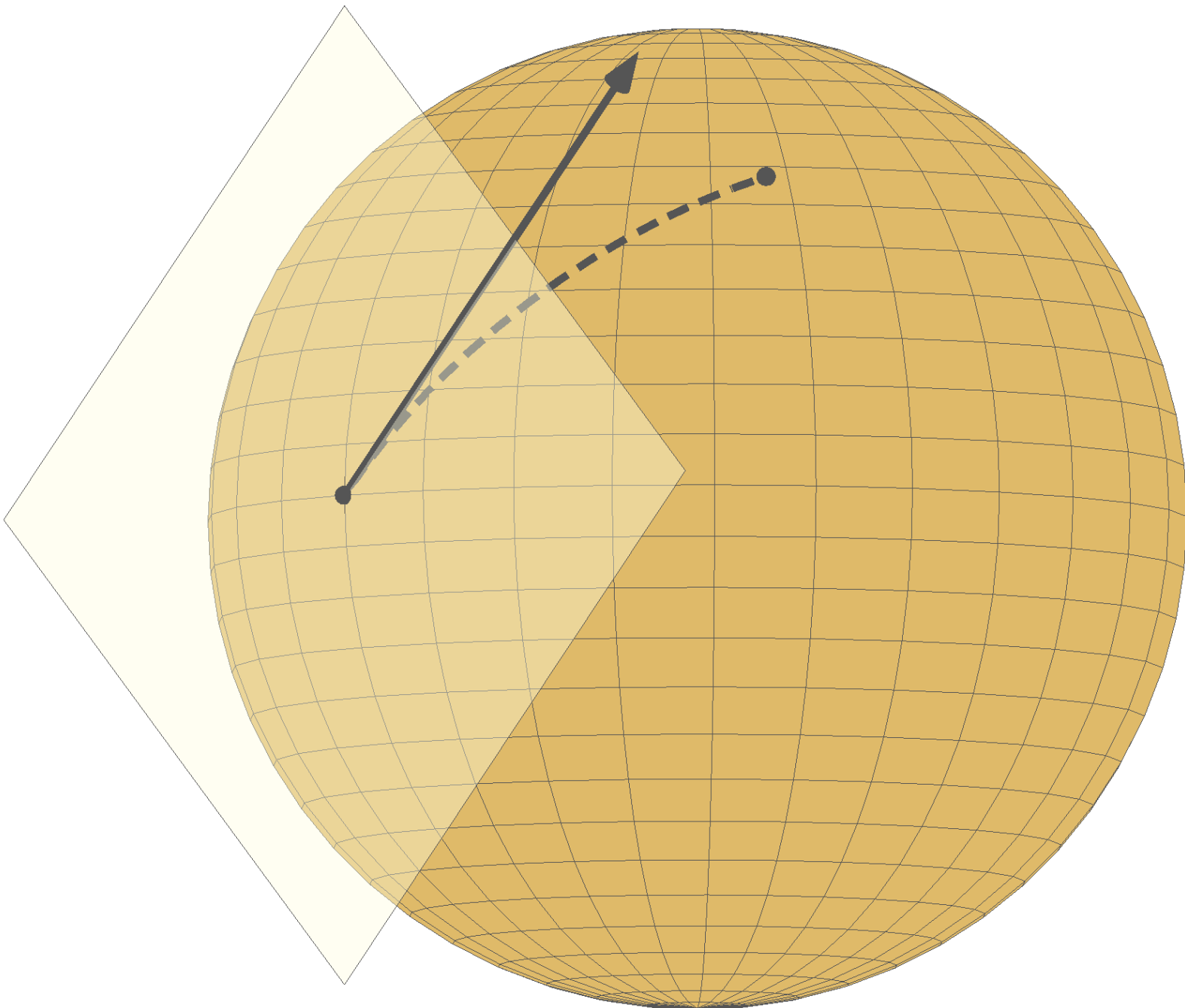












To optimize, we only need the search space to be a Riemannian manifold

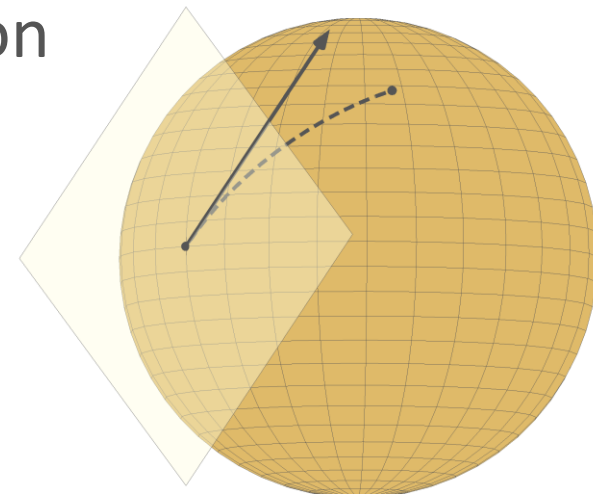
$$\min_{x \in M} f(x)$$

We need...

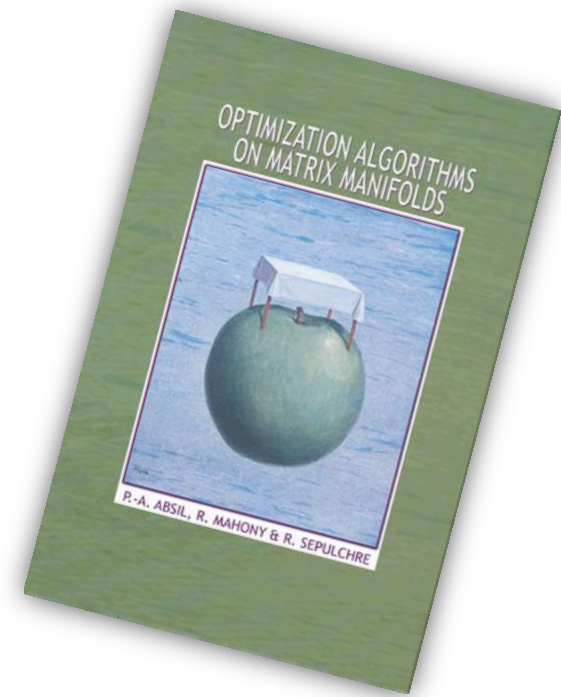
A notion of directions along which we can move
tangent space, tangent vector

A notion of steepest descent direction
inner product, gradient

A means of moving along a direction
Geodesics, retractions



The theory is mature at this point.



What's been missing is matching software.

Manopt

A Matlab toolbox to make optimization on manifolds feel as simple as unconstrained optimization

With generic solvers,
a library of manifolds and
diagnostics tools

Low-rank matrix completion

$$\min_{X \in M} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2$$

$$M = \{X \in \mathbb{R}^{m \times n} : \text{rank}(X) = r\}$$

Find a **matrix X of rank r** which matches A as well as possible on a subset of entries.

Independent component analysis

$$\min_{X \in M} \sum_{i=1, \dots, N} \|\text{offdiag}(X^T C_i X)\|^2$$

$$M = \{X \in \mathbb{R}^{n \times p} : \text{ddiag}(X^T X) = I_p\}$$

Find a demixing **matrix X with unit-norm columns** which simultaneously diagonalizes given C_i 's as well as possible.

Distance matrix completion

$$\min_{X \in M} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2$$

$$M = \{X \in \mathbb{R}^{n \times n} : q_1, \dots, q_n \in \mathbb{R}^p \text{ and } X_{ij} = \|q_i - q_j\|^2\}$$

Find a **Euclidean distance matrix** X which matches A as well as possible on a subset of entries.

Estimation of rotations

$$\min_{R_1, \dots, R_N \in M} \sum_{(i,j) \in \Omega} \|R_i R_j^T - H_{ij}\|^2$$

$$M = \{Q \in \mathbb{R}^{n \times n} : Q^T Q = I \text{ and } \det(Q) = +1\}$$

Find **rotation matrices** R_i which match measurements of relative rotations $H_{ij} \approx R_i R_j^T$ as well as possible.

Example code for dominant eigenvectors

$$\max_x \frac{x^T A x}{x^T x}$$

Example code for dominant eigenvectors

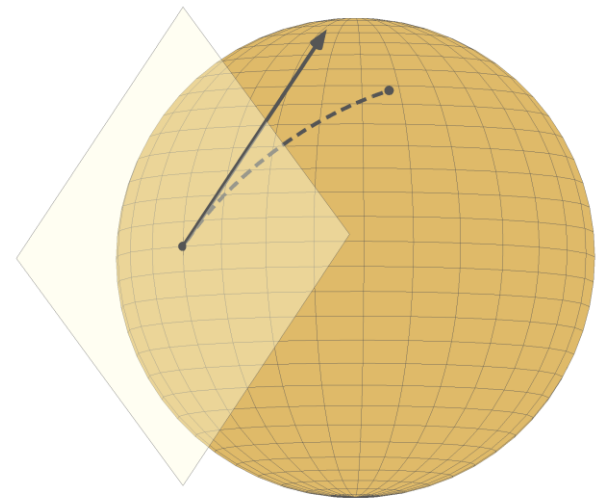
$$\max_{\|x\|=1} x^T A x$$

$$M = \{x \in \mathbb{R}^n : x^T x = 1\}$$

$$f(x) = x^T A x$$

$$\nabla f(x) = 2Ax$$

$$\text{grad } f(x) = (I - xx^T) \nabla f(x)$$



```
import manopt.solvers.trustregions.*;
import manopt.manifolds.sphere.*;
import manopt.tools.*;

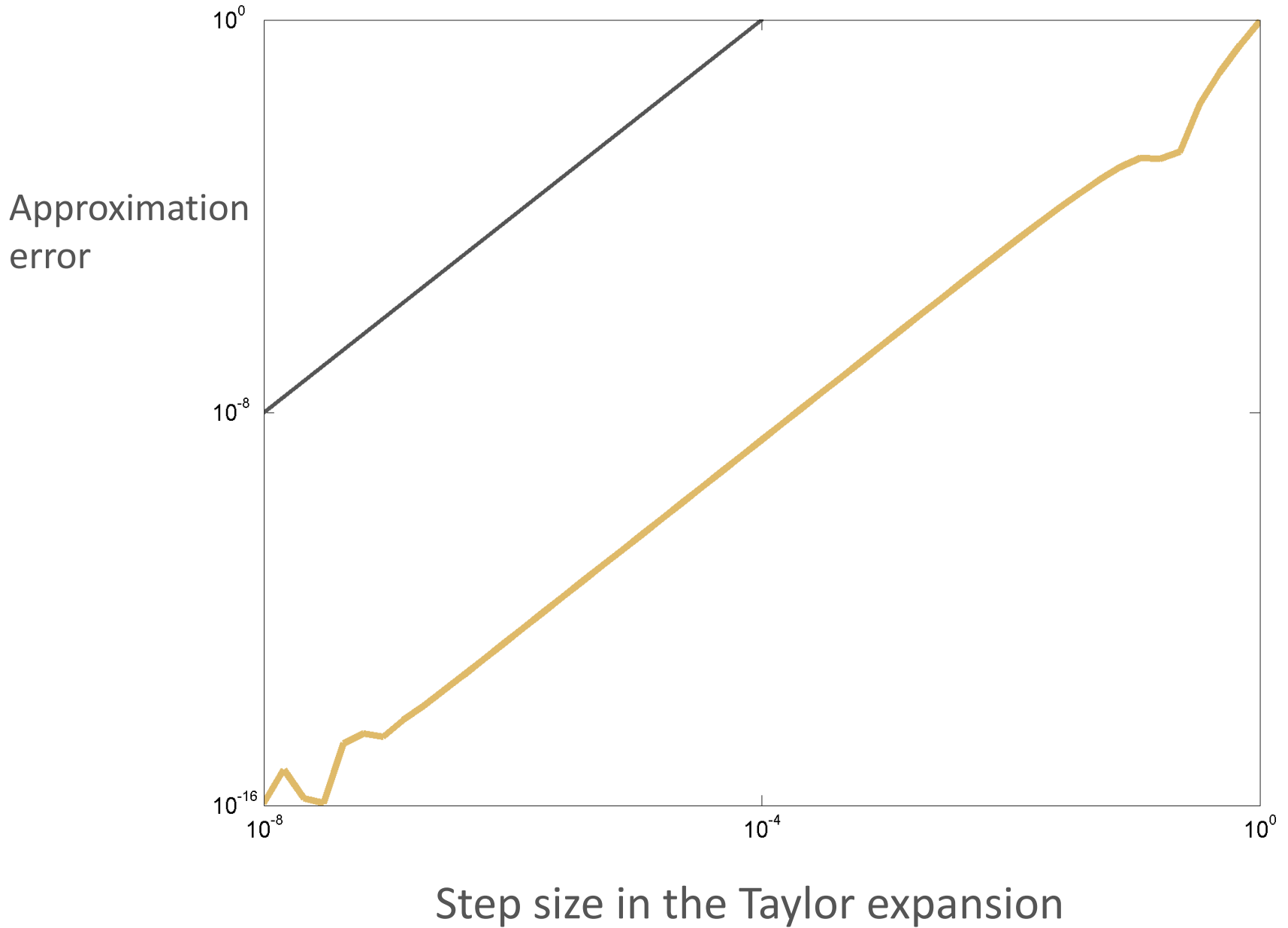
% Generate the problem data.
n = 1000;
A = randn(n);
A = .5*(A+A');

% Create the problem structure.
manifold = spherefactory(n);
problem.M = manifold;

% Define the problem cost function and its gradient.
problem.cost = @(x) -x'*(A*x);
problem.grad = @(x) manifold.egrad2rgrad(x, -2*A*x);

% Numerically check gradient consistency.
checkgradient(problem);
```

Gradient check



```
import manopt.solvers.trustregions.*;
import manopt.manifolds.sphere.*;
import manopt.tools.*;

% Generate the problem data.
n = 1000;
A = randn(n);
A = .5*(A+A');

% Create the problem structure.
manifold = spherefactory(n);
problem.M = manifold;

% Define the problem cost function and its gradient.
problem.cost = @(x) -x'*(A*x);
problem.grad = @(x) manifold.egrad2rgrad(x, -2*A*x);

% Numerically check gradient consistency.
checkgradient(problem);

% Solve.
[x xcost info] = trustregions(problem);
```

				f: 1.571531e+000	grad : 4.456216e+001	
REJ TR-	k: 1	num_inner: 1		f: 1.571531e+000	grad : 4.456216e+001	negative curvature
acc	k: 2	num_inner: 1		f: -2.147351e+001	grad : 3.053440e+001	negative curvature
acc	k: 3	num_inner: 2		f: -3.066561e+001	grad : 3.142679e+001	negative curvature
acc	k: 4	num_inner: 2		f: -3.683374e+001	grad : 2.125506e+001	exceeded trust region
acc	k: 5	num_inner: 3		f: -4.007868e+001	grad : 1.389614e+001	exceeded trust region
acc	k: 6	num_inner: 4		f: -4.237276e+001	grad : 9.687523e+000	exceeded trust region
acc	k: 7	num_inner: 6		f: -4.356244e+001	grad : 5.142297e+000	exceeded trust region
acc	k: 8	num_inner: 8		f: -4.412433e+001	grad : 2.860465e+000	exceeded trust region
acc	k: 9	num_inner: 20		f: -4.438540e+001	grad : 3.893763e-001	reached target residual-kappa
acc	k: 10	num_inner: 20		f: -4.442759e+001	grad : 4.116374e-002	reached target residual-kappa
acc	k: 11	num_inner: 24		f: -4.442790e+001	grad : 1.443240e-003	reached target residual-theta
acc	k: 12	num_inner: 39		f: -4.442790e+001	grad : 1.790137e-006	reached target residual-theta
acc	k: 13	num_inner: 50		f: -4.442790e+001	grad : 3.992606e-010	dimension exceeded

Gradient norm tolerance reached.

Total time is 2.966843 [s] (excludes statsfun)

```
import manopt.solvers.trustregions.*;
import manopt.manifolds.sphere.*;
import manopt.tools.*;

% Generate the problem data.
n = 1000;
A = randn(n);
A = .5*(A+A');

% Create the problem structure.
manifold = spherefactory(n);
problem.M = manifold;

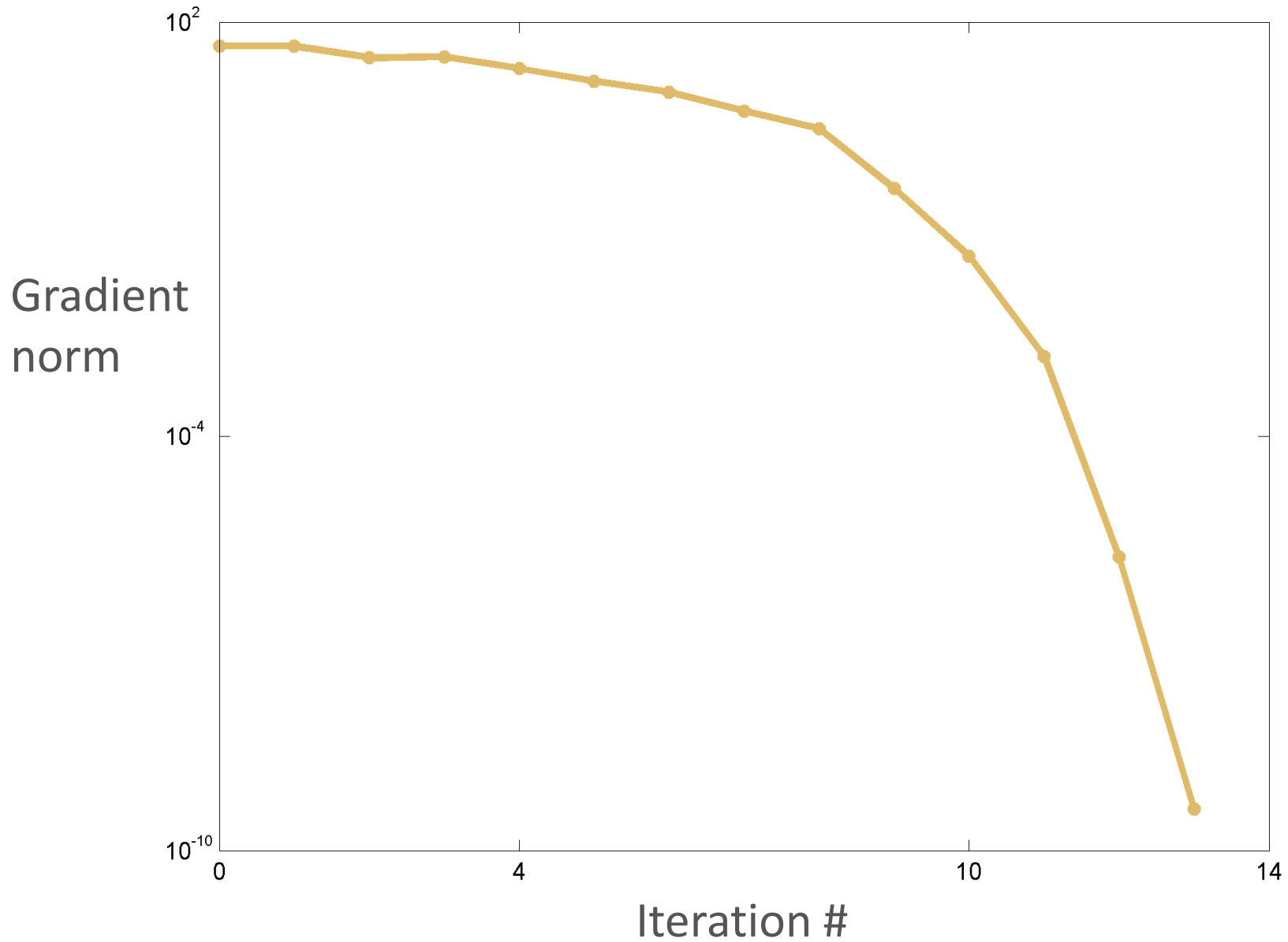
% Define the problem cost function and its gradient.
problem.cost = @(x) -x'*(A*x);
problem.grad = @(x) manifold.egrad2rgrad(x, -2*A*x);

% Numerically check gradient consistency.
checkgradient(problem);

% Solve.
[x xcost info] = trustregions(problem);

% Display some statistics.
semilogy([info.iter], [info.gradnorm], '-');
```

Convergence of the trust-regions method



Riemannian optimization is...

Well-understood

Theory is available for many algorithms

Useful

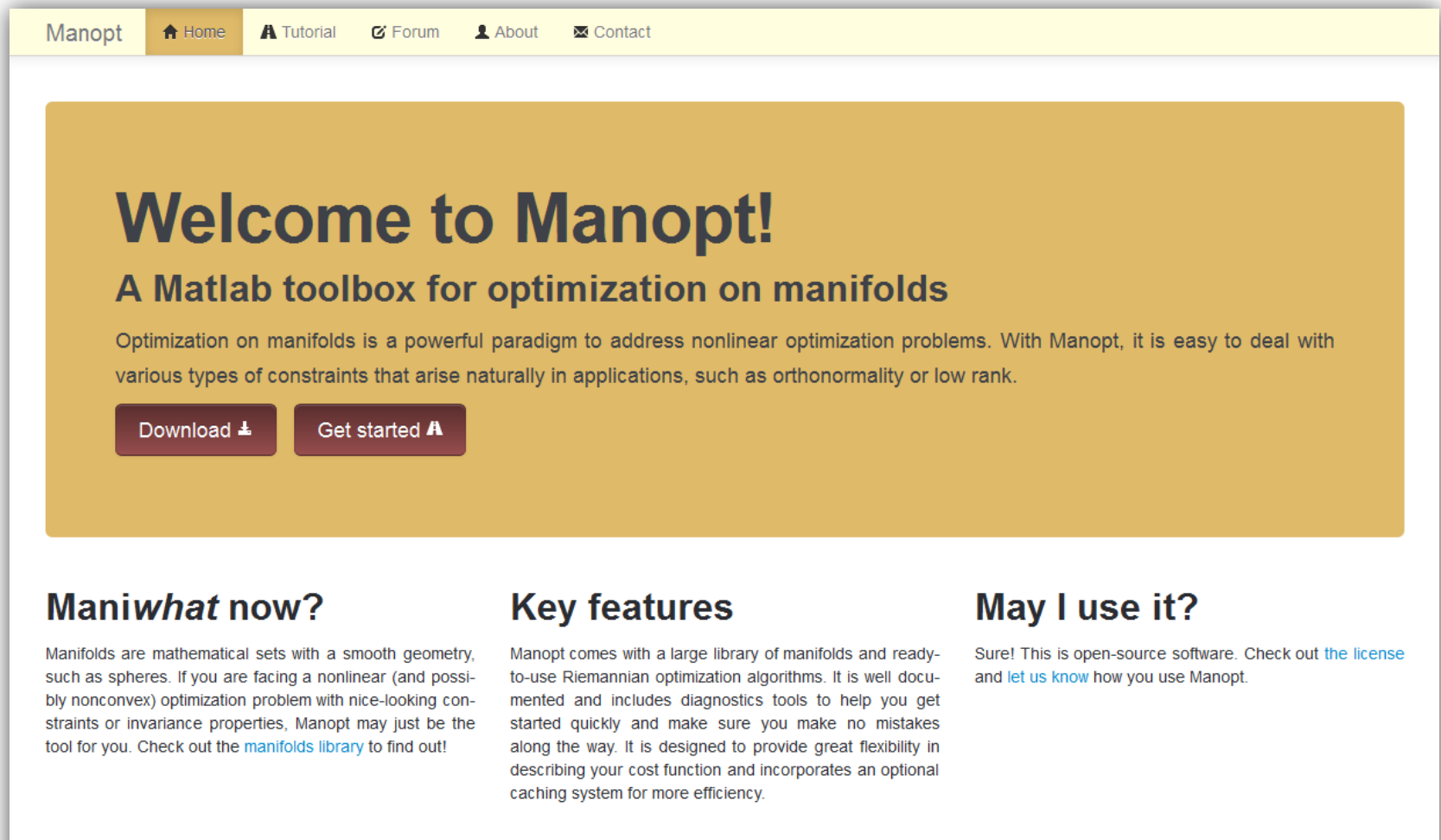
We covered a few fashionable problems

Easy

With Manopt, you simply provide the cost

Manopt is open source and documented

www.manopt.org



The image shows a screenshot of the Manopt website homepage. The page has a yellow navigation bar at the top with the Manopt logo and links for Home, Tutorial, Forum, About, and Contact. The main content area has a large orange background with the heading "Welcome to Manopt!" and a sub-heading "A Matlab toolbox for optimization on manifolds". Below this is a paragraph of text and two buttons: "Download" and "Get started". The page is divided into three columns: "Maniwhat now?", "Key features", and "May I use it?".


Manopt


Home Tutorial Forum About Contact

Welcome to Manopt!

A Matlab toolbox for optimization on manifolds

Optimization on manifolds is a powerful paradigm to address nonlinear optimization problems. With Manopt, it is easy to deal with various types of constraints that arise naturally in applications, such as orthonormality or low rank.

Download 

Get started 

Maniwhat now?

Manifolds are mathematical sets with a smooth geometry, such as spheres. If you are facing a nonlinear (and possibly nonconvex) optimization problem with nice-looking constraints or invariance properties, Manopt may just be the tool for you. Check out the [manifolds library](#) to find out!

Key features

Manopt comes with a large library of manifolds and ready-to-use Riemannian optimization algorithms. It is well documented and includes diagnostics tools to help you get started quickly and make sure you make no mistakes along the way. It is designed to provide great flexibility in describing your cost function and incorporates an optional caching system for more efficiency.

May I use it?

Sure! This is open-source software. Check out [the license](#) and [let us know](#) how you use Manopt.