Manifolds in Julia

Manifolds.jl and ManifoldsBase.jl

Seth Axen[a], Mateusz Baran[b,c], Ronny Bergmann[d]

[a]University of California, San Francisco, USA
[b]AGH University of Science and Technology, Kraków, Poland
[c]Cracow University of Technology, Kraków, Poland
[d]Technische Universität Chemnitz, Chemnitz, Germany

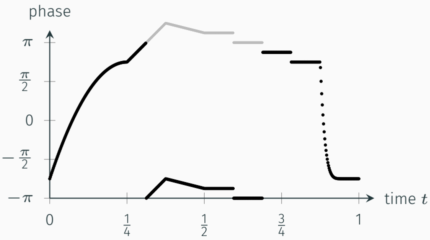
juliacon 2020

Lisbon, Portugal, Everywhere, July 27–31, 2020

Why Manifolds?

- cyclic data (phase, e.g. InSAR)
- spherical data (earth, directions)
- orientations
- diffusion tensors

→ non-linear spaces ☺ Riemannian manifolds



phase data
[Bergmann, Laus, Steidl, and Weinmann 2014]

Transferring properties, we provide methods for those data

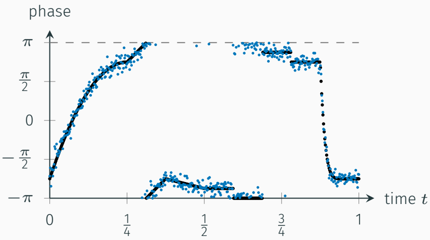
- statistics
- data processing, e.g. imaging
- optimization
- ...

- Implement methods generically for any manifold
- Make it easy to specialize methods using multiple dispatch

Why Manifolds?

- cyclic data (phase, e.g. InSAR)
- spherical data (earth, directions)
- orientations
- diffusion tensors

➔ non-linear spaces ☺ Riemannian manifolds



noisy phase data
[Bergmann, Laus, Steidl, and Weinmann 2014]

Transferring properties, we provide methods for those data

- statistics
- data processing, e.g. imaging
- optimization
- …

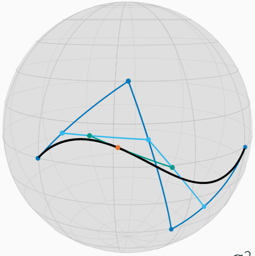
☑ Implement methods generically for any manifold

☑ Make it easy to specialize methods using multiple dispatch

Why Manifolds?

- cyclic data (phase, e.g. InSAR)
- spherical data (earth, directions)
- orientations
- diffusion tensors

➜ non-linear spaces ☺ Riemannian manifolds



a curve on $\mathbb{S}^2$
[Bergmann and Gousenbourger 2018]

Transferring properties, we provide methods for those data

- statistics
- data processing, e.g. imaging
- optimization
- ...

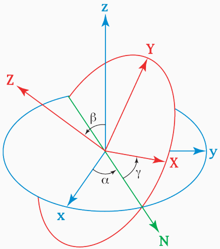
Implement methods generically for any manifold

Make it easy to specialize methods using multiple dispatch

Why Manifolds?

- cyclic data (phase, e.g. InSAR)
- spherical data (earth, directions)
- orientations
- diffusion tensors

➜ non-linear spaces ☺ Riemannian manifolds



Euler angles for orientations
W File:Euler.png

Transferring properties, we provide methods for those data

- statistics
- data processing, e.g. imaging
- optimization
- ...

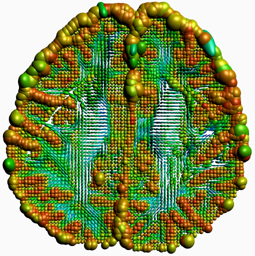
Implement methods generically for any manifold

Make it easy to specialize methods using multiple dispatch

Why Manifolds?

- cyclic data (phase, e.g. InSAR)
- spherical data (earth, directions)
- orientations
- diffusion tensors

⇒ non-linear spaces ☺ Riemannian manifolds



Diffusion tensors from DT-MRI
data: Camino project

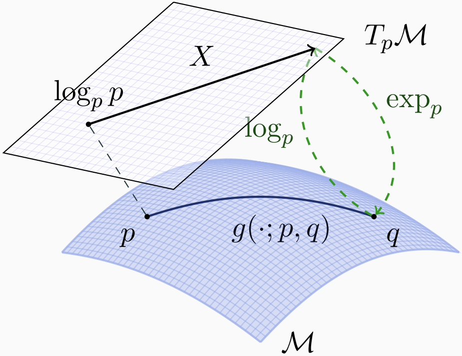
Transferring properties, we provide methods for those data

- statistics
- data processing, e.g. imaging
- optimization
- ...

Implement methods generically for any manifold

Make it easy to specialize methods using multiple dispatch

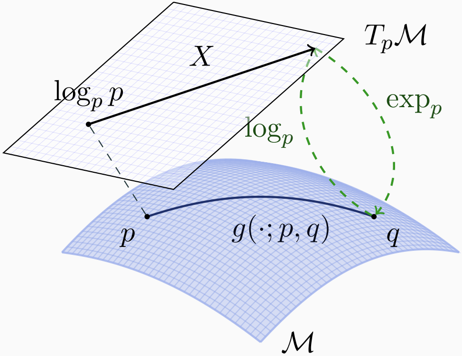
Background: A Riemannian manifold



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 tangential spaces.

[Absil, Mahony, and Sepulchre 2008]

Background: A Riemannian manifold



Geodesic $g(\cdot; p, q)$ shortest path (on $\mathcal{M}$) between $p, q \in \mathcal{M}$

Tangent space $\mathrm{T}_p\mathcal{M}$ at p, with inner product $\langle\cdot,\cdot\rangle_p$

Logarithmic map $\log_p q = \dot{g}(0; p, q)$ "speed towards q"

Exponential map $\exp_p X = g(1)$, where $g(0) = p$, $\dot{g}(0) = X$

Implementing a Riemannian manifold

In `Manifolds.jl` a manifold is a subtype of `Manifold{𝔽}`, $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}, \mathbb{H}\}$, that implements functions from `ManifoldsBase.jl` 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

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

➜ interface allows for generic algorithms for any `Manifold`:

`norm(M,p,X)`, `geodesic(M, p, X)` and `shortest_geodesic(M, p, q)` are available with the above implemented.

A manifold decorator

Properties are often implicitly given, like the Riemannian metric tensor.

The interface provides a decorator manifold acting semi-transparently, i.e. transparent for all functions specified not to be affected by this decorator.

Example.
`ValidationManifold(M)` performs (when applicable)

- `is_manifold_point(M, p)`
- `is_tangent_vector(M, p, X)`

before and after every basic function from the interface (`exp`, `log`, `inner`,...).

A different metric

Goal. Implement different Riemannian metric tensors for a manifold.

➜ transparent e.g. for `manifold_dimension(M)`

- existing implementation: default metric (transparent)
- other functions: implementation using parametric type

Example.

- `M = SymmetricPositiveDefinite(3)` has
- `MetricManifold(M, LinearAffineMetric)` as synonym,
- `MetricManifold(M, LogEuclidean)` is a second metric,
- `MetricManifold(M, LogCholesky)` is a metric providing an `exp`.

😊 `exp` defaults to a method numerically solving the ODE.

Embedded manifolds `AbstractEmbeddedManifold{𝔽,<:AbstractEmbeddingType}`

Goal. Model embedded manifold(s) of a manifold

😊 reuse functions (like `inner`) from embedding.

- different types via `AbstractEmbeddingType T`
- provide `embed`, `project` & `get_embedding`

Examples.

- `Sphere{N,𝔽} <: AbstractEmbeddedManifold {𝔽, DefaultIsometricEmbeddingType}` into `Euclidean(N+1)`, ➜ its `inner` is used
- `SymmetricMatrices{N,𝔽} <: AbstractEmbeddedManifold {𝔽, TransparentIsometricEmbedding}` into `Euclidean(N, N; field=𝔽)`, ➜ use its `exp` & `log`
- or use directly `EmbeddedManifold(Manifold, Embedding)`

Lie groups

`AbstractGroupManifold{𝔽, <:AbstractGroupOperation}`

Goal. Model manifolds that have a group structure

- a manifold with a smooth binary operator $\circ$, e.g. translation, multiplication, composition
- an `identity` element
- together with `MetricManifold`: left-, right- & bi-invariant metric

Examples.

- `TranslationGroup(n)` is $\mathbb{R}^n$ with translation action
- `SpecialOrthogonal{n} <: GroupManifold{Rotations{n},MultiplicationOperation}`
- `SpecialEuclidean(n)` is a `SemidirectProductGroup`
- or directly `GroupManifold(Manifold, Operation)`

Build more manifolds

Given Riemannian manifolds $\mathcal{M}, \mathcal{M}_1, \ldots, \mathcal{M}_N$ you can build

- the `ProductManifold`: $\mathcal{N} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_N$
  points are tuples $p = (p_1, \ldots, p_N)$, where $p_i \in \mathcal{M}_i$
  Example. `N = ProductManifold(M1, M2)` or `N = M1×M2`
- the `PowerManifold`: $\mathcal{N} = \mathcal{M}^{n_1 \times n_2}$
  points are (nested) arrays $p = (p_{i,j})_{i,j=1}^{n_1,n_2}$, where $p_{i,j} \in \mathcal{M}$
  Example. `N = PowerManifold(M, 5, 6)` or `N = M^(5, 6)`
- the `TangentBundle`: $\mathcal{N} = T\mathcal{M} = \dot{\bigcup}_{p\in\mathcal{M}} T_p\mathcal{M}$
  points are tuples $p = (q, X)$, where $X \in T_q\mathcal{M}$
  Example. `N = TangentBundle(M)`
  or more generally `VectorBundleFibers`

😊 easy access/modification: `p[N, i]`

Statistics

The mean $\frac{1}{n}\sum_{k=1}^{n} x_i$ can be phrased as $\operatorname*{arg\,min}_y \sum_{i=1}^{n} \|x_i - y\|_2^2$

😊 replace norm of difference by distance

➜ no closed form but a smooth optimization problem.

- `mean(M, x[, weights, method])` to compute the (weighted) mean, where `method` is a gradient descent, geodesic interpolation or an extrinsic estimator
- `var(M, x, weights, m=mean(M, x, w))` variance of the data (in $T_m\mathcal{M}$)
- similarly available `std`, `kurtosis`, `skewness`, `moment`

A `median` is given by any $\operatorname*{arg\,min}_y \sum_{i=1}^{n} d_{\mathcal{M}}(x_i, y)$

➜ nonsmooth optimization problem on $\mathcal{M}$

➜ method: `CyclicProximalPointEstimation` [Bačák 2014]

Statistics

The mean $\frac{1}{n}\sum_{k=1}^{n} x_i$ can be phrased on a manifold as $\arg\min_y \sum_{i=1}^{n} d_{\mathcal{M}}(x_i, y)^2$

😊 replace norm of difference by distance

➜ no closed form but a smooth optimization problem.

- `mean(M, x[, weights, method])` to compute the (weighted) mean, where `method` is a gradient descent, geodesic interpolation or an extrinsic estimator
- `var(M, x, weights, m=mean(M, x, w))` variance of the data (in $T_m\mathcal{M}$)
- similarly available `std`, `kurtosis`, `skewness`, `moment`

A `median` is given by any $\arg\min_y \sum_{i=1}^{n} d_{\mathcal{M}}(x_i, y)$

➜ nonsmooth optimization problem on $\mathcal{M}$

➜ method: `CyclicProximalPointEstimation` [Bačák 2014]

Bases in tangent spaces

A tangent vector $X \in T_p\mathcal{M}$ is often neither a vector nor of dimension $\dim_{\mathcal{M}}$.

➔ use an `AbstractBasis` for tangent spaces, e.g.

- `DefaultBasis` for any basis
- `DefaultOrthogonalBasis`, `DefaultOrthonormalBasis` w. r. t. $\langle\cdot,\cdot\rangle_p$
- `ProjectedOrthonormalBasis` from the embedding
- `DiagonalizingOrthonormalBasis` diagonalizes the curvature tensor

☺ do not store the basis explicitly, but provide an iterator.

➔ to store them explicitly use `get_basis(M, p, basis)` to get a `CachedBasis`.

Then use `coords = get_coordinates(M, p, X, basis)`
and its inverse `X = get_vector(M, p, coords, basis)`

Available basic manifolds

Currently the following manifolds are available

- Centered matrices*
- Cholesky space
- Circle*
- Euclidean*,†,‡
- Fixed-rank matrices*
- Generalized Stiefel*
- Generalized Grassmann*
- Grassmann*
- Hyperbolic space
- Lorentzian Manifold
- Multinomial matrices
- Oblique manifold*
- Probability simplex
- Rotations
- Skew-symmetric matrices*
- (Array) Sphere*
- Symmetric matrices*
- Symmetric positive definite
- Torus
- Unit-norm symmetric matrices*
- … your favourite manifold?

*also available as complex-valued manifold.

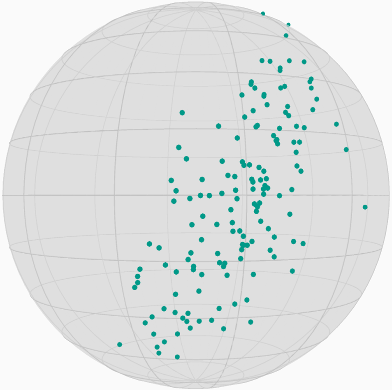
†also available as quaternion-valued manifold.

‡can also be used for numbers, vectors, matrices, tensors,…

Example: A PCA on the sphere $\mathbb{S}^2$

Compute a principal component analysis (PCA) for a `Vector` `pts` of points on $\mathbb{S}^2$ by computing a PCA in the tangent space of the mean m.

```
using Manifolds, MultivariateStats
M = Sphere(2)
```



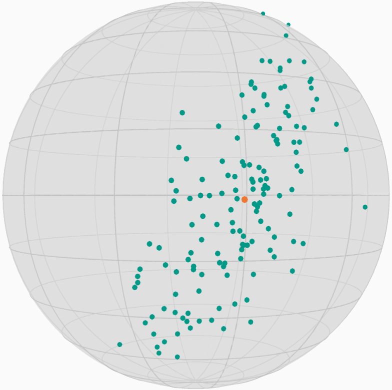
a set of points on $\mathbb{S}^2$

Example: A PCA on the sphere $\mathbb{S}^2$

- Compute a principal component analysis (PCA) for a `Vector` `pts` of points on $\mathbb{S}^2$ by computing a PCA in the tangent space of the mean m.

```
using Manifolds, MultivariateStats
M = Sphere(2)

m = mean(M, pts)
```



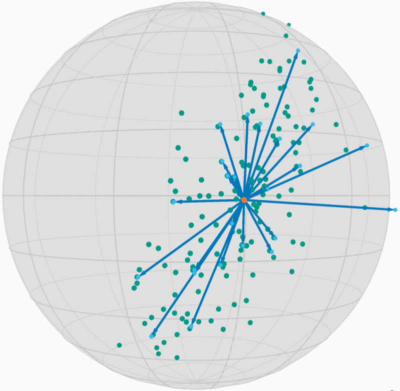
a set of points on $\mathbb{S}^2$ and its mean

Example: A PCA on the sphere $\mathbb{S}^2$

Compute a principal component analysis (PCA) for a `Vector` `pts` of points on $\mathbb{S}^2$ by computing a PCA in the tangent space of the mean m.

```
using Manifolds, MultivariateStats
M = Sphere(2)

m = mean(M, pts)
logs = log.(Ref(M), Ref(m), pts)
```



logarithmic maps of the points into $T_m\mathbb{S}^2$

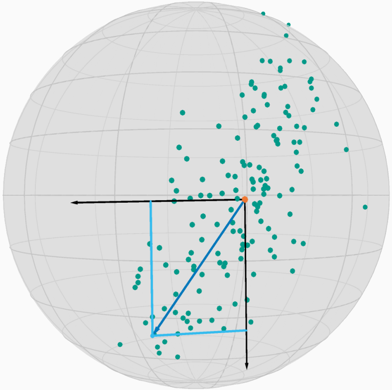
Example: A PCA on the sphere $\mathbb{S}^2$

Compute a principal component analysis (PCA) for a `Vector` `pts` of points on $\mathbb{S}^2$ by computing a PCA in the tangent space of the mean m.

```
using Manifolds, MultivariateStats
M = Sphere(2)

m = mean(M, pts)
logs = log.(Ref(M), Ref(m), pts)

basis = DefaultOrthonormalBasis()
coords = map(X -> get_coordinates(M, m, X, basis), logs)
coords_red = reduce(hcat, coords)
```



a tangent vector in coordinates of a basis

Example: A PCA on the sphere $\mathbb{S}^2$

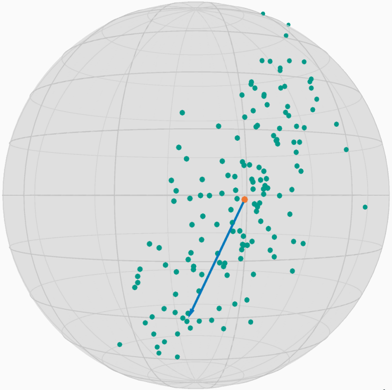
Compute a principal component analysis (PCA) for a `Vector` `pts` of points on $\mathbb{S}^2$ by computing a PCA in the tangent space of the mean m.

```
using Manifolds, MultivariateStats
M = Sphere(2)

m = mean(M, pts)
logs = log.(Ref(M), Ref(m), pts)

basis = DefaultOrthonormalBasis()
coords = map(X -> get_coordinates(M, m, X, basis), logs)
coords_red = reduce(hcat, coords)

z = zeros(manifold_dimension(M))
model = fit(PCA, coords_red; maxoutdim=1, mean=z)
X = get_vector(M, m, reconstruct(model, [1.0]), basis)
```



PCA as a tangent vector X (scaled by $\frac{1}{2}$)

Example: A PCA on the sphere $\mathbb{S}^2$

- Compute a principal component analysis (PCA) for a `Vector` `pts` of points on $\mathbb{S}^2$ by computing a PCA in the tangent space of the mean m.

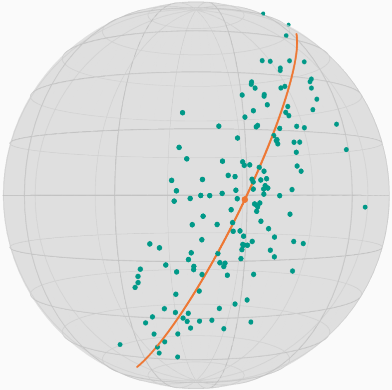
```julia
using Manifolds, MultivariateStats
M = Sphere(2)

m = mean(M, pts)
logs = log.(Ref(M), Ref(m), pts)

basis = DefaultOrthonormalBasis()
coords = map(X -> get_coordinates(M, m, X, basis), logs)
coords_red = reduce(hcat, coords)

z = zeros(manifold_dimension(M))
model = fit(PCA, coords_red; maxoutdim=1, mean=z)
X = get_vector(M, m, reconstruct(model, [1.0]), basis)

geodesic(M, m, X, range(-1.0, 1.0, length=101))
```



principal component as geodesic on $\mathbb{S}^2$

Manopt.jl: Optimization on manifolds

Build upon `ManifoldsBase.jl` to solve

$$\operatorname*{arg\,min}_{p \in \mathcal{M}} F(p)$$

using

- a `Problem p` describing function, gradient, Hessian,...
- `Options o` specifying a solver settings and state
- call `solve(p, o)`, which includes `StoppingCriterion` calls

➔ implement your own solver within the solver framework

- `initialize_solver!(p, o)`
- `step_solver!(p, o, i)`

The Manopt family: manoptjl.org

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

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

Manopt.jl: Available solvers

- cyclic proximal point
- gradient descent
- conjugate gradient descent
- subgradient method
- Nelder–Mead
- Douglas–Rachford
- Riemannian trust regions
- ☺ all with a high level interface

Example.

Compute the mean of a `pts` vector of `n` points on `M`.

```
F(y) = sum(1/(2*n) * distance.(Ref(M), pts, Ref(y)).^2)
∇F(y) = sum(1/n*∇distance.(Ref(M), pts, Ref(y)))

xMean = gradient_descent(M, F, ∇F, pts[1];
    debug = [:Iteration, " | ", :x, " | ", :Change, " | ", :Cost, "\n",
        :Stop, 10]
)
```

Summary & Outlook

`ManifoldsBase.jl` is a flexible lightweight interface for manifolds.

`Manifolds.jl`

- provides a library of basic manifolds
- provides tools for manifolds, for example statistics
- embeddings, metrics and group manifolds with a decorator pattern

`Manopt.jl` provides optimization tools on manifolds based on `ManifoldsBase.jl`

What's next?

- automatic differentiation & Zygote
- a generic way to implement distributions
- more abstract manifolds (quotient manifold, projective space)
- more manifolds… maybe add your favourite manifold?

Literature

- Absil, P.-A., R. Mahony, and R. Sepulchre (2008). *Optimization Algorithms on Matrix Manifolds.* Princeton University Press. DOI: 10.1515/9781400830244.
- Bačák, M. (2014). "Computing medians and means in Hadamard spaces". In: *SIAM Journal on Optimization* 24.3, pp. 1542–1566. DOI: 10.1137/140953393.
- Bergmann, R. and P.-Y. Gousenbourger (2018). *A variational model for data fitting on manifolds by minimizing the acceleration of a Bézier curve.* arXiv: 1807.10090.
- Bergmann, R., F. Laus, G. Steidl, and A. Weinmann (2014). "Second order differences of cyclic data and applications in variational denoising". In: *SIAM Journal on Imaging Sciences* 7.4, pp. 2916–2953. DOI: 10.1137/140969993.
- do Carmo, Manfredo P. (1992). *Riemannian Geometry.* Mathematics: Theory & Applications. Birkhäuser Boston, Inc., Boston, MA. ISBN: 0-8176-3490-8.

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

ronnybergmann.net/talks/2020-JuliaCon-Manifolds.pdf