NTNU

Norwegian University of Science and Technology

Manifolds.jl

Ronny Bergmann

joint work with
Seth D. Axen, Mateusz Baran.

Workshop in Geometry and Geometric Integration,
Bergen, September 7, 2022

Motivation

- abstract definition of manifolds and properties thereon
  e. g. different metrics, Lie groups, embeddings

$\Rightarrow$ implement abstract algorithms for generic manifolds / Lie Groups / ...

- easy to implement own manifolds & easy to use
- well-documented and well-tested
- fast.

Why Julia?

- high-level language, properly typed
- multiple dispatch (cf. `f(x)`, `f(x::Number)`, `f(x::Int)`)
- just-in-time compilation, solves two-language problem
- I like the language.

Defining a Manifold: Types and Dispatch

`ManifoldsBase.jl` defines a common interface for Riemannian manifolds.

- a manifold is an abstract type `AbstractManifold{𝔽}` with parameter field $\mathbb{F}$.
- concrete manifolds: subtype containing dimension/size information
  Examples. (from `Manifolds.jl`) `Euclidean{Tuple{3,3},ℝ}` or `Sphere{2,ℂ}`

😊 Easy constructors `M1 = ℝ^(3,3)` and `M2 = Sphere(2,ℂ)`

Points `p` and (co-)tangent vectors `ξ,X` are usually not typed specifically

😊 works with arbitrary `AbstractArray` types, e. g. `StaticArrays`

- they are subtyped os `ManifoldPoint` or `TVector` for different representations.
  Example. on `M3 = Hyperbolic(2)`:
  - arrays `p` are equivalent to using `HyperboloidPoint(p)`
  - further representations: `PoincareBallPoint` and `PoincareHalfSpacePoint`
  - for tangent vectors like `PoincareBallTVector`: vector operations also defined

Implementing a Riemannian Manifold

`ManifoldsBase.jl` uses a `AbstractManifold{𝔽}` with type parameter $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}, \mathbb{H}\}$ to provide an interface for implementing functions like

- `inner(M, p, X, Y)` for the Riemannian metric $(X, Y)_p$
- `exp(M, p, X)` and `log(M, p, q)`,
- more general: `retract(M, p, X, m)`, where `m` is a retraction method
- similarly: `parallel_transport(M, p, X, q)` and `vector_transport_to(M, p, X, q, m)`

for your manifold `M` a subtype of the abstract manifold `Manifold{𝔽}`.

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

➔ basis for generic algorithms working on any `Manifold` and generic functions like `norm(M,p,X)`, `geodesic(M, p, X)` and `shortest_geodesic(M, p, q)`

🔗 juliamanifolds.github.io/ManifoldsBase.jl/

Decorating a Manifold: Adding Features using Traits

A manifold can be extended with features/properties using traits (THTT), e. g.

- `MetricManifold{𝔽,AbstractManifold{𝔽},AbstractMetric}`
  - implement a second metric `MyMetric <: AbstractMetric` for a manifold
  - metric-unrelated functions (like `dimension(M)`) are just "passed on"
    ☺ no need to reimplement them
- `EmbeddedManifold{𝔽,AbstractManifold,AbstractManifold}`
  - implement embedding-specific `embed!` and `project!` functions
  - for an `IsIsometricEmbeddedManifold` (use `inner` from embedding)
  - for an `IsEmbeddedSubmanifold` (use also `exp!`, `log!`, `geodesic` from embedding)
- `GroupManifold{𝔽,AbstractManifold{𝔽},AbstractGroupAction}`
  - models Lie groups, e. g. `Rotations(n)` vs. `SpecialOrthogonal(n)`
  - additional functions like `exp_lie(G,X)` and `log_lie(G,p)` or `Identity(G)`
  - again: unrelated functions "passed down" to the internal manifold

Manifolds.jl – A library of manifolds in Julia

`Manifolds.jl` is based on the `ManifoldsBase.jl` interface. [Axen, Baran, RB, and Rzecki 2021]

Features.

- different metrics
- Lie groups
- Build manifolds using
  - Product manifold $\mathcal{M}_1 \times \mathcal{M}_2$
  - Power manifold $\mathcal{M}^{n\times m}$
  - Tangent bundle
- Embedded manifolds
- perform statistics
- well-documented, including formulae and references
- well-tested, $>98\,\%$ code cov.

Manifolds. For example

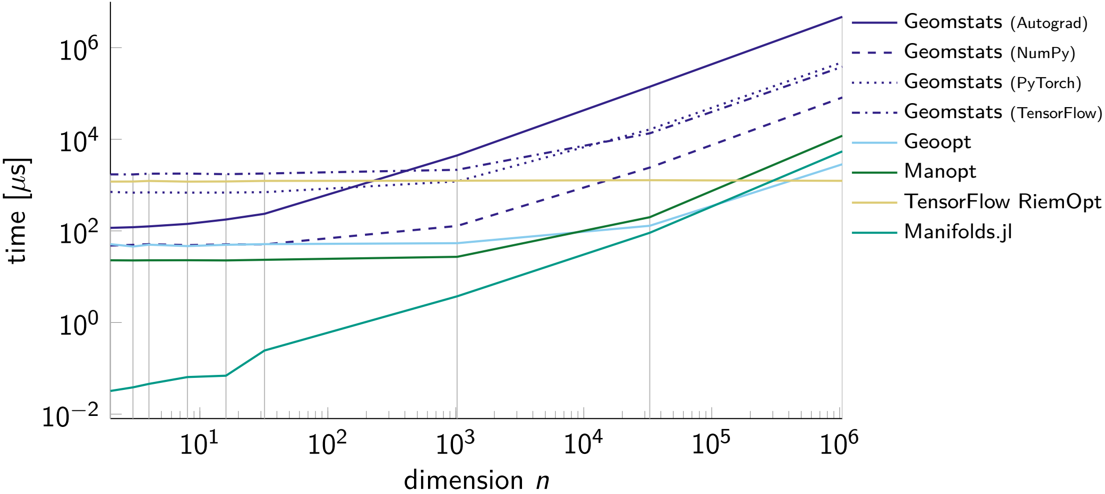
- (unit) Sphere, Circle & Torus
- Fixed Rank Matrices
- (Generalized) Stiefel & Grassmann
- Hyperbolic space
- Rotations, O(n), SO(n), SU(n)
- Several further Lie groups
- Symmetric positive definite matrices
- Symplectic & Symplectic Stiefel
- …

juliamanifolds.github.io/Manifolds.jl/

JuliaCon 2020 youtu.be/md-FnDGCh9M

Preview: Benchmark of the logarithmic map on $\mathbb{H}^n$

For $n = 2, 3, 2^2, 2^3, 2^4, 2^5, 2^{10}, 2^{15}, 2^{20}$ we compare



➔ For $n > 2^{16}$: PyTorch & TensorFlow based packages faster.

…we could maybe try using `LazyArrays.jl` in Julia.

Summary

`ManifoldsBase.jl` is an abstract interface for Manifolds

- ...to define/implement manifolds
- ...to implement generic algorithms for arbitrary manifolds

https://juliamanifolds.github.io/ManifoldsBase.jl/

`Manifolds.jl` is a a library of manifolds implemented using the interface

https://juliamanifolds.github.io/Manifolds.jl/

Further Packages

`Manopt.jl` – optimization algorithms for functions $f\colon \mathcal{M} \to \mathbb{R}$

https://manoptjl.org

`ManifoldsDiffEq.jl` – combines `ManifoldsBase.jl` and `OrdinaryDiffEq.jl` to provide solvers for differential equations on Riemannian manifolds

https://juliamanifolds.github.io/ManifoldDiffEq.jl/

References

- Axen, S. D., M. Baran, RB, and K. Rzecki (2021). *Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds*. arXiv: 2106.08777.
- Boumal, N., B. Mishra, P.-A. Absil, and R. Sepulchre (2014). "Manopt, a Matlab toolbox for optimization on manifolds". In: *The Journal of Machine Learning Research* 15, pp. 1455–1459. URL: https://www.jmlr.org/papers/v15/boumal14a.html.
- Kochurov, M., R. Karimov, and S. Kozlukov (2020). *Geoopt: Riemannian Optimization in PyTorch*. arXiv: 2005.02819.
- Miolane, N. et al. (2020). "Geomstats: A Python Package for Riemannian Geometry in Machine Learning". In: *Journal of Machine Learning Research* 21.223, pp. 1–9. URL: http://jmlr.org/papers/v21/19-027.html.
- Smirnov, O. (2021). *TensorFlow RiemOpt: a library for optimization on Riemannian manifolds*. arXiv: 2105.13921.

ronnybergmann.net/talks/2022-Geo2Int-Manifoldsjl.pdf