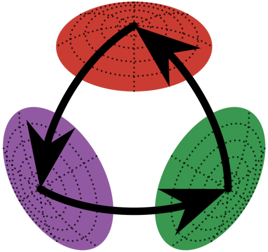
Groups and smooth geometry using LieGroups.jl



Ronny Bergmann
kellertuer

NTNU, Trondheim, Norway.

JuliaCon 2025,
Pittsburg, July 24, 2025.

NTNU
Norges teknisk-naturvitenskapelige universitet

Motivation

In a lot of applications, data or variables like for example

- rotation matrices,
- invertible matrices,
- rigid body motions: translation & rotation,

and many more, are **non-Euclidean**: For two rotation matrices $R_1, R_2 \in \mathbb{R}^{3\times 3}$ their sum $R_1 + R_2$ is **not** a rotation matrix.

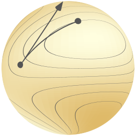
But. All 3 examples share a lot of structure

- they are smooth: elements have "a neighbourhood on a hyper surface"
- they have a group operation

Goals.

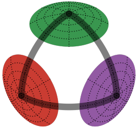
- an interface to define and work with these structures
- a library of these "groups with smoothness"

JuliaManifolds: Nonlinear data in Julia

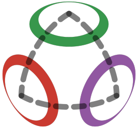


Nov. 2016 `Manopt.jl`
optimization algorithms on Riemannian manifolds

Jun 2019 first release `Manopt.jl v0.1`
same day: start of `Manifolds.jl`
to work with Riemannian manifolds in Julia

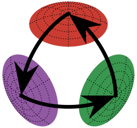


Nov 2019 `ManifoldsBase.jl`
an interface to work on and define
Riemannian manifolds.



Mar 2020 `Manifolds.jl v0.1`
which already contained a `GroupManifold`

Oct 2024 `LieGroups.jl` (kudos `yuehhua`)
an interface for and a library of Lie groups



"Manifolds in numerical computations with JuliaManifolds"
by Mateusz Baran, here @ JuliaCon 2025.

What is a manifold?

Informally. A manifold $\mathcal{M}$ is a set that locally "looks like" some $\mathbb{R}^d$ "around" every point. d is called the manifold dimension.

Example 1. Our earth, or a sphere, $\mathbb{S}^2 = \bigl\{p \in \mathbb{R}^3 \mid \|p\| = 1\bigr\}$ locally looks like $\mathbb{R}^2$, just take an atlas. **But** this works only locally.

Example 2. The set of $2D$ rotation matrices $R_\alpha = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}$ locally looks like a line, but (again) not globally, since $R_0 = R_{2\pi}$.

Example 3. The set of $3D$ rotation matrices $R \in \mathbb{R}^{3\times 3}$, i.e. with $R^\mathsf{T} R = I_3$ and $\det(R) = 1$, is locally isomorphic to $\mathbb{R}^3$.

one could use Euler angles, but they have their disadvantages.

From Manifolds to Lie groups

An operation $\cdot : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ is called (abelian) group operation if

1. $a \cdot b \in \mathcal{M}$ for all $a, b \in \mathcal{M}$
2. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all $a, b, c \in \mathcal{M}$
3. the exists a neutral element $e \in \mathcal{M}$,
   such that $e \cdot a = a \cdot e = a$ for all $a \in \mathcal{M}$
4. For $a \in \mathcal{M}$ there exists an inverse element a^{-1} s. t. $a \cdot a^{-1} = e$
5. the group is abelian if $a \cdot b = b \cdot a$

If both the group operation $\cdot$ and the map $a \mapsto a^{-1}$ are smooth,
then the pair $G = (\mathcal{M}, \cdot)$ is called a **Lie group**.

Often $a, b \in \mathcal{M} \subset \mathbb{R}^{n \times n}$ are matrices and $\cdot$ is the matrix multiplication
➜ a, b have to be invertible!

A short history

1823 Niels Henrik Abel (1802–1829)
introduces group theory to study
the solutions of algebraic equations

1854 Bernhard Riemann (1826–1866)
introduces differential geometry, especially
Riemannian manifolds, to study
intrinsic properties of surfaces

1870 Marius Sophus Lie (1842–1899)
introduces Lie groups to study
symmetries in differential equations

Tangent Spaces & the Lie Algebra

For a point $g \in G$ take

- a smooth curve $c(t)$ “running through” $c(0) = g$
- its derivative $\dot{c}(0)$: a “looking direction at” g
- collect all derivatives having the same value as $X = [\dot{c}(0)]$

X is called a tangent vector and collecting all possible (different values):
T_gG is the tangent space at g.

Special case.
At the identity $g = e$ we get $\mathfrak{g} := T_eG$ the so-called Lie algebra.

Technical Detour: Riemannian Manifolds

- Every tangent space is a d-dimensional vector space.
- We define an inner product ("measure angles") $\langle\cdot,\cdot\rangle_g$ for each $T_g\mathcal{M}$
- measure lengths using the induced norm $\|X\| = \sqrt{\langle X, X\rangle_g}$
- When $\langle\cdot,\cdot\rangle_g$ varries smoothly in g
- a Riemannian metric

A manifold $\mathcal{M}$ together with such a metric is called Riemannian manifold.

A bit technical, because we have to remember/store/implement
a whole family of inner products.

First code in `Manifolds.jl`

The set of n-by-n rotation matrices is a manifold called (simply) `Rotations(n)` in `Manifolds.jl`.

```julia
using Manifolds, LinearAlgebra

M = Rotations(3)
d = manifold_dimension(M)    # returns 3
g = rand(M); is_point(M, g)  # is true
                             # Lie group checks: the old way
e = one(g)                   # neutral: the identity matrix
is_point(M, e)               # true
is_point(M, g+e)             # false
is_point(M, g*e)             # true
X = zero_vector(M,e)         # from the TeG (the Lie algebra)
inner(M, e, X, X).           # norm(M, e, X)^2 -> yields zero
Y = rand(M; vector_at=e)     # a random vector from TeG.
```

First code in `LieGroups.jl`

The rotation matrices together with matrix multiplication are called the special orthogonal group $\mathrm{SO}(n)$.

```julia
using LieGroups, LinearAlgebra

G = SpecialOrthogonalGroup(3)
d = manifold_dimension(G)    # returns 3 as before
g = rand(G); is_point(G, g) # works as before
                             # Lie group checks: the new way
e = identity_element(G)      # new name: the identity
is_point(G, e)               # true
h = compose(G, g, e)         # the group operation
is_point(G, h)               # ...and we stay in G of course
𝔤 = LieAlgebra(G)            # does not explicitly store e
X = zero_vector(𝔤)           #
inner(𝔤, X, rand(𝔤))         # inner on 𝔤.
```

Technical Detour: Left-invariant vector fields

For the left group operation $\lambda_g(h) = g \cdot h$, consider its differential $D\lambda_g(h) \colon T_gG \to T_{gh}G$. `diff_left_compose(G,g,h,X)`

A vector field $V \colon G \to TG, \quad g \mapsto V(g) \in T_gG$ is called left-invariant if

$$D\lambda_g(h)[V(h)] = V(\lambda_g(h)) \quad \text{holds for all} \quad g, h \in G$$

➜ Knowing V at one point $V(e) = X \in T_eG$, we know it anywhere.

Example 1.
On $G = (\mathbb{R}, +)$ we have $\lambda_g(h) = g + h$ ➜ $D\lambda_g(h)[X] = X$.
⚠ yields constant vector fields $V(g) = X$; we can "attach X anywhere".

Example 2.
On $G = (\mathrm{SO}(n), \cdot)$ we have $\lambda_g(h) = gh$ ➜ $D\lambda_g(h)[X] = gX$.
⚠ For $V(e) = X$ we have $V(g) = gX \in T_gG$

Model (nearly) everything on the Lie algebra $\mathfrak{g}$

We saw

- $X \in \mathfrak{g}$ implies $D\lambda_g(e)[X] \in T_gG$
- Knowing X and g is enough, since for $Y = gX \in T_gG$ we have $g^{-1}Y = X \in \mathfrak{g}$

Given a metric $\langle \cdot, \cdot \rangle$ on $\mathfrak{g}$
Use this idea to introduce the so-called left invariant metric

$$\langle Y, Z \rangle_g = \langle g^{-1}Y, g^{-1}Z \rangle \quad \text{for} \quad X, Y \in T_gG$$

is a smoothly varying metric on G.

even easier: just store elements X from $\mathfrak{g}$ to avoid the group op. with g^{-1}

The Lie group exponential

Motivation. Generalise the idea to take a tangent vector ("direction") $X \in \mathfrak{g}$ back ("down to") the Lie group. Or: "walk that way".

Definition.(Hilgert, Neeb, 2012, Def. 9.2.2) `exp(G,X)`

The (Lie group) exponential function $\exp_G \colon \mathfrak{g} \to G$ is defined as

$$\exp_G(X) = \gamma_X(1),$$

where γ_X is the unique curve that solves the initial value problem

$$\dot{\gamma}(t) = \gamma(t)X, \qquad \gamma_X(0) = e, \quad \dot{\gamma}_X(0) = X.$$

Example 1. On $G = (\mathbb{R}, +)$ we obtain $exp_G(X) = X$

Example 2. On the circle we obtain the complex exponential $X \mapsto \mathrm{e}^{\mathrm{i}X}$

Example 3. On $G = (\mathrm{SO}(n), \cdot)$ we obtain the matrix exponential e^X

Be careful with the name exp

There are several things called the exponential

Lie group (function) `exp(G,X), exp!(G,g,X)`

the map $\exp_G\colon \mathfrak{g} \to G$ from the last page,

Idea: "Start walking" from e

Lie group (map) `exp(G,g,X), exp!(G,h,g,X)`

Interpret $X \in \mathfrak{g}$ as $gX \in T_gG$ and compute (due to chain rule)

$exp_g(X) = g \exp_G(X)$

Idea: "Start walking" from g

Riemannian manifold (map) `exp(M,g,X), exp!(M,h,g,X)`

On the `M=base_manifold(G)` follow the geodesic w. r t. the Riemannian metric.

Idea: Follow the "straightest" curve from g in direction X.

…and of course the "classical" exponential and matrix exponential.

Example I (cont.): Special orthogonal group $\mathrm{SO}(3)$

```julia
using LieGroups, LinearAlgebra, Rotations
SO3 = SpecialOrthogonalGroup(3) # 3d Rotations w/ matrix mult.
g = [1.0 0.0 0.0; 0.0 1.0 0.0; 0.0 0.0 1.0]
h = RotZ(π/4)                   # 45 degrees in XY plane

is_point.(Ref(SO3), [g,h])      # returns [true, true]
k = compose(SO3, g, h)
compose!(SO3, k, inv(SO3, g), k)# in-place of k; avoid allocs
isapprox(SO3, k, h)             # we inverted the first compose

so3 = LieAlgebra(SO3); X = [0 0.3 0; -0.3 0 0; 0 0 0]
is_point(so3, X)                  # same as is_vector(SO3,e,X)
l = exp(SO3, X); is_point(SO3, l) # so3 -> SO3
Y = log(SO3, l); isapprox(so3,X,Y)# ...and back
is_point(so3, X+Y)                # so3 is a vector space
```

Group actions and (semidirect) product Lie groups

A group action describes how a Lie group G acts on some manifold $\mathcal{M}$:

$$\sigma : G \times \mathcal{M} \to \mathcal{M}, \qquad q = \sigma(g, p) \in \mathcal{M}$$

Example. For $G = \mathrm{SO}(3)$, $\mathcal{M} = \mathbb{R}^3$ we have $\sigma(R, p) = Rp$.
Here the group action describes how vectors in $\mathbb{R}^3$ are actually rotated.

Product Lie groups. `G×H`

A (direct) product group $G \times H$ works on tuples elementwise

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1 \star g_2, h_1 \diamond h_2)$$

Semidirect product Lie groups. `G⋉H`

On a (left) semidirect product group $(G, \star) \ltimes (H, \diamond)$ the first (left) group acts on the second

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1 \star g_2, h_1 \diamond \sigma_{g_1}(h_2))$$

(in $\ltimes = \ltimes_\sigma$ the action is implicit; analogously: a right semidirect product $\rtimes$)

Example II: Special Euclidean group $\mathrm{SE}(n)$

Rigid body motions $\mathrm{SE}(3) = (\mathrm{SO}(3), \cdot) \ltimes (\mathbb{R}^3, +)$. Group operation: $(R, t) \circ (S, u) = (RS, t + Ru)$.[1]

```
using LieGroups, LinearAlgebra, RecursiveArrayTools, Rotations
SE3 = SpecialEuclideanGroup(3) # or use SO3 ⋉ T3 from before

g = ArrayPartition(RotZ(π/3)*RotY(π/4), [1.0, 2.0])
h = ArrayPartition(RotX(π/6), [0.0, 1.0])
gh = compose(SE3, g, h)                #
e = identity_element(SE3, typeof(g)) # 2nd arg: representation.
                                      # def. without: hom coord.

X = log(SE3, gh) # inverse: gh = exp(SE3,X)
se3 = LieAlgebra(SE3)
c = vee(se3, X) # repr. as vector, coeffs in a basis of 𝔰𝔢(3)
```

[1]obtained also from default matrix product in homogeneous coordinates.

Checks along the way: `ValidationLieGroup`

- default: neither input nor output are checked

⚠ hard to see "where things go wrong"

💡 use `ValidationLieGroup(G)`

Ansatz. Wrap the Lie group `G` as `ValidationLieGroup(G)`
➜ every function call is "enhanced" by checks `is_point`/`is_vector` on corresponding inputs/outputs

Keyword arguments.

- `error=:error` how to "report" errors in the checks
  change to `:warn` or `:info`
- `ignore_contexts=[:input]` to e. g. not validate inputs
- `ignore_functions=Dict(exp => :All)` to exclude certain function (& their contexts) from validation

Functions available in `LieGroups.jl`

.

Lie group G

- `adjoint(G,g,X)`
- `compose(G,g,h)` and `inv(G,g)`
- `conjugate(G,g,h)`
- `exp(G,g,X)`, `exp(G,X)`
- `log(G,g,h)`, `log(G,g)`
- `inv_left_compose(G,g,h)`, `inv_right_compose(G,g,h)`
- differentials of conjugate, inv, compose (left & right arg)
- `jacobian_conjugate(G, g, h)`
- `identity_element(G)`

Lie algebra $\mathfrak{g}$

- `base_lie_group(`$\mathfrak{g}$`)`
- `lie_bracket(`$\mathfrak{g}$`, X, Y)`
- `get_coordinates(`$\mathfrak{g}$`,X)` (vee)
- `get_vector(`$\mathfrak{g}$`,c)` (hat)
- `inner(`$\mathfrak{g}$`,X,Y)`
- `zero_vector(`$\mathfrak{g}$`,X,Y)`

all also in-place: `f!(G, ret, args...)`

suitable ones automatically "pass through" to `Manifolds.jl`

Notable differences to `GroupManifolds`

In a nutshell.

`GroupManifolds` equipped a manifolds with a group operation

`LieGroups` use a manifold internally

On `LieGroups`

- the Lie group exponential is more prominent

  previously called `exp_lie` / `exp_inv`
- naming was simplified and unified
- `LieAlgebra` its own type / vector space

  nearly no need to allocate an identity
- more efficient (power/product) Lie groups
- a generic implementation of semidirect product Lie groups
- more consistent default: left invariant vector fields

see tutorials/transition/ for a complete list.

Available Lie groups

Meta Lie groups. To build Lie groups from existing ones

- `PowerLieGroup(G, n)` or `G^n`
- `ProductLieGroup(G1, G2)` or `G1 × G2`
- `LeftSemidirectProductGroup()` or `G1 ⋉ G2`
- `RightSemidirectProductGroup()` or `G1 ⋊ G2`

Lie groups.

- `CircleGroup()`, 3 variants: $\mathbb{R}$, embedded in $\mathbb{C}$ or $\mathbb{R}^2$
- `GeneralLinearGroup(n; field=ℝ)` and `HeisenbergGroup(n)`
- `OrthogonalGroup(n)` and `UnitaryGroup(n)`
- `SpecialEuclideanGroup(n; variant=:left)` or `:right`
- `SpecialLinearGroup(n; field=ℝ)` or $\mathbb{C}$
- `SpecialOrthogonalGroup(n)` and `SpecialUnitaryGroup(n)`
- `SymplecticGroup(n)` and `TranslationGroup(n; field=ℝ)`

Summary

We gave a short introduction to Lie Groups and `LieGroups.jl`.

The package provides

Interfaces to work with and define

- Lie groups & group operations
- Lie algebras
- group actions
- ➜ directly work on abstract Lie groups or define your own

A library of Lie groups

- well-documented with formulae and literature
- based on `Manifolds.jl`
- efficiently implemented

Links & References

LieGroups.jl documentation: juliamanifolds.github.io/LieGroups.jl/

References.

- Axen, S. D.; M. Baran; RB; K. Rzecki (2023). “Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds”. *ACM Transactions on Mathematical Software* 49.4. DOI: 10.1145/3618296.
- RB (2022). “Manopt.jl: Optimization on manifolds in Julia”. *Journal of Open Source Software* 7.70, p. 3866. DOI: 10.21105/joss.03866.
- Hilgert, J.; K.-H. Neeb (2012). *Structure and Geometry of Lie Groups*. Springer Monographs in Mathematics. DOI: 10.1007/978-0-387-84794-8.

...and special thanks to Michael Goerz for

- DocumenterInterLinks.jl
- DocumenterCitations.jl

ronnybergmann.net/talks/2025-LieGroups-JuliaCon.pdf or

Links & References

`LieGroups.jl` documentation: juliamanifolds.github.io/LieGroups.jl/

References.

- Axen, S. D.; M. Baran; RB; K. Rzecki (2023). "Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds". *ACM Transactions on Mathematical Software* 49.4. DOI: 10.1145/3618296.
- RB (2022). "Manopt.jl: Optimization on manifolds in Julia". *Journal of Open Source Software* 7.70, p. 3866. DOI: 10.21105/joss.03866.
- Hilgert, J.; K.-H. Neeb (2012). *Structure and Geometry of Lie Groups*. Springer Monographs in Mathematics. DOI: 10.1007/978-0-387-84794-8.

...and special thanks to Michael Goerz for

- `DocumenterInterLinks.jl`
- `DocumenterCitations.jl`

Links & References

`LieGroups.jl` documentation: juliamanifolds.github.io/LieGroups.jl/

References.

Axen, S. D.; M. Baran; RB; K. Rzecki (2023). “Manifolds.jl: An Extensible Julia Framework for Data Analysis on Manifolds”. *ACM Transactions on Mathematical Software* 49.4. DOI: 10.1145/3618296.

RB (2022). “Manopt.jl: Optimization on manifolds in Julia”. *Journal of Open Source Software* 7.70, p. 3866. DOI: 10.21105/joss.03866.

Hilgert, J.; K.-H. Neeb (2012). *Structure and Geometry of Lie Groups*. Springer Monographs in Mathematics. DOI: 10.1007/978-0-387-84794-8.

…and special thanks to Michael Goerz for

- `DocumenterInterLinks.jl`
- `DocumenterCitations.jl`

ronnybergmann.net/talks/2025-LieGroups-JuliaCon.pdf or