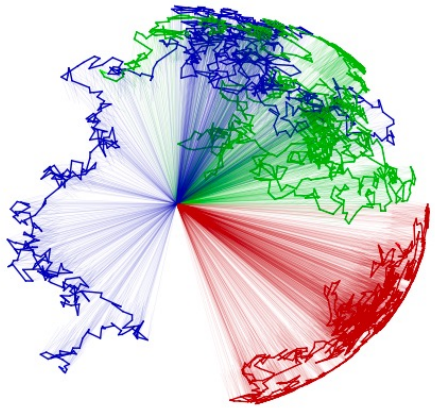
JAXGeometry and Hyperiax

Python Implementations of computational
differential geometry and tree traversal by JAX

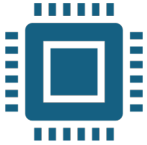
CIRM 2024



Hyperiax

Gefan Yang
University of Copenhagen
31 May, 2024

JAXGeometry



Differential geometry
+
Stochastic dynamics
+
JAX autodiff/JIT compiling
=
High performance optimized symbolic computational framework

Using JAX to reformulate the computation of geometric statistics
+
stochastic dynamics on manifold

Support:

Geometric statistics

Dynamics on groups

***Stochastic processes on manifolds**

JAXGeometry --manifolds

`Manifold()`
- Euclidean space: **`Euclidean()`**
- Heisenberg group: **`Heisenberg()`**
- LDDMM landmark manifold: **`Landmarks()`**

```python
class Manifold(object):
    """ Base manifold class """

    def __init__(self, g, ...):
        self.g = g  # Riemannian metric

    def chart(self):
        pass        # chart
```

`EmbeddedManifold()`
- 2D cylinder: **`Cylinder()`**
- 2D Ellipsoid: **`Ellipsoid()`**
- Hyperbolic plane: **`H2()`**
- 2D torus: **`Torus()`**

```python
class EmbeddedManifold(Manifold):
    """ Embedded manifold base class """

    def __init__(self, F, invF, ...):
        Manifold.__init__(self)

        self.F = F                  # embedding map
        self.invF = invF            # inverse of embedding map
        self.JF = jacfwdx(self.F)   # jacobian of F by autodiff
        self.invJF = jacfwdx(self.invF)

        # Riemannian metric by F
        self.g = lambda x: jnp.tensordot(self.JF(x),self.JF(x),(0,0))
```

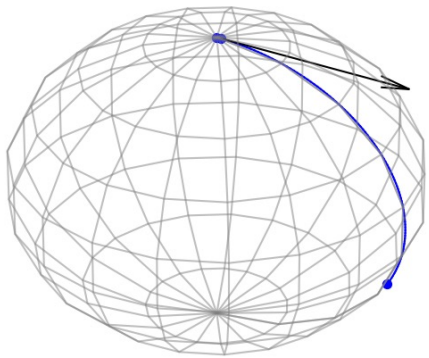
`LieGroup()`
- Symmetric positive definite matrices: **`SPDN()`**
- Special orthogonal group: **`SON()`**
- General linear group: **`GLN()`**

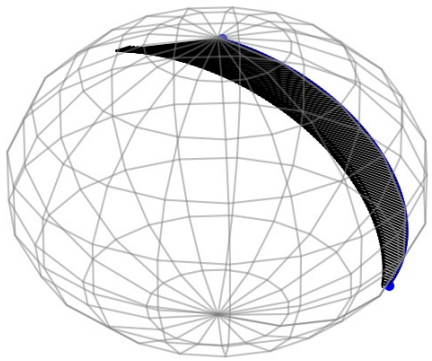
```python
class LieGroup(EmbeddedManifold):
    """ Base Lie Group class """

    def __init__(self, invariance, ...):
        EmbeddedManifold.__init__(self)

        self.invariance = invariance    # left or right invarient

    def Expm(self, g):
        pass        # exponential map

    def Logm(self, b):
        pass        # logarithm map
```

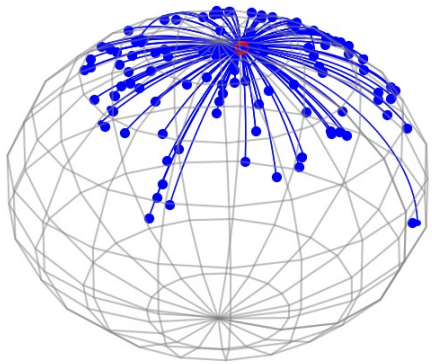
JAXGeometry
--manifolds

On a Riemannian manifold, we can compute:

- Geodesics: `jaxgeometry.Riemannian.geodesic`
- Curvature: `jaxgeometry.Riemannian.curvature`
- Parallel transport: `jaxgeometry.Riemannian.parallel_transport`
- Fréchet mean: `jaxgeometry.statistics.Frechet_mean`
- *Stochastic processes: `jaxgeometry.stochastics`

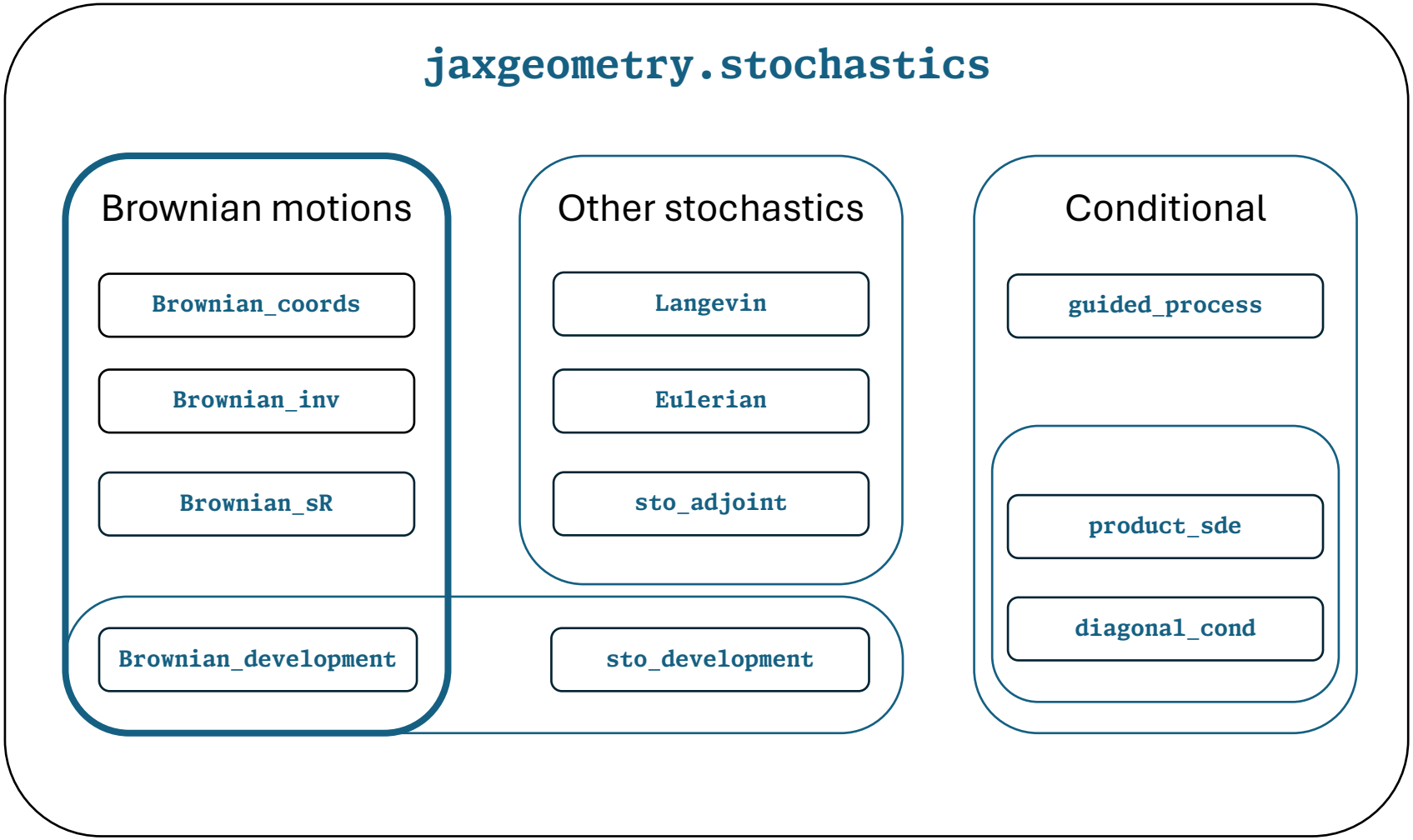




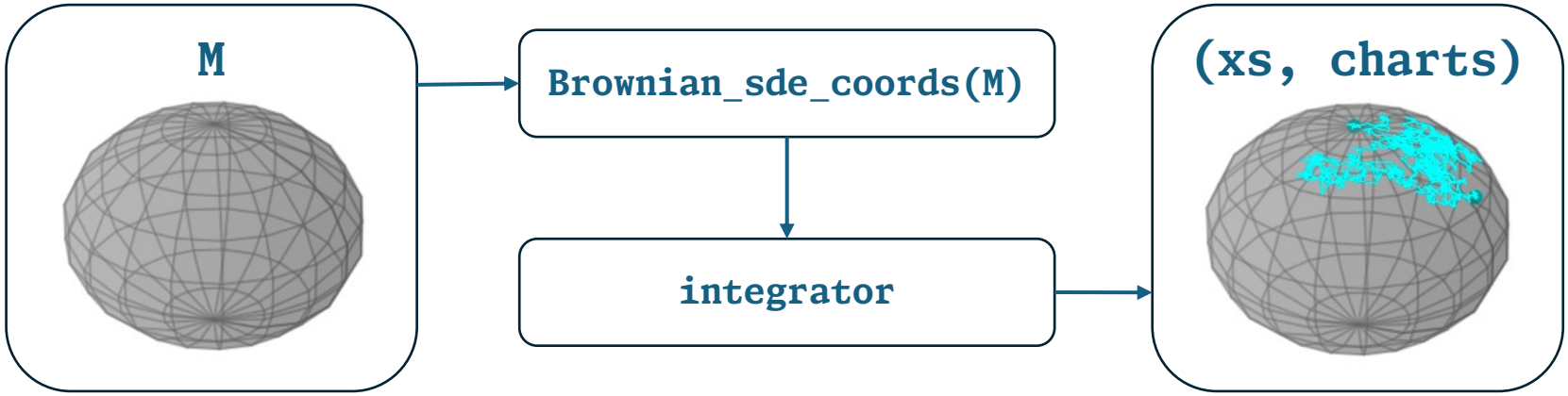


JAXGeometry
--Stochastic processes

With a defined metric, we can simulate stochastic processes on manifolds

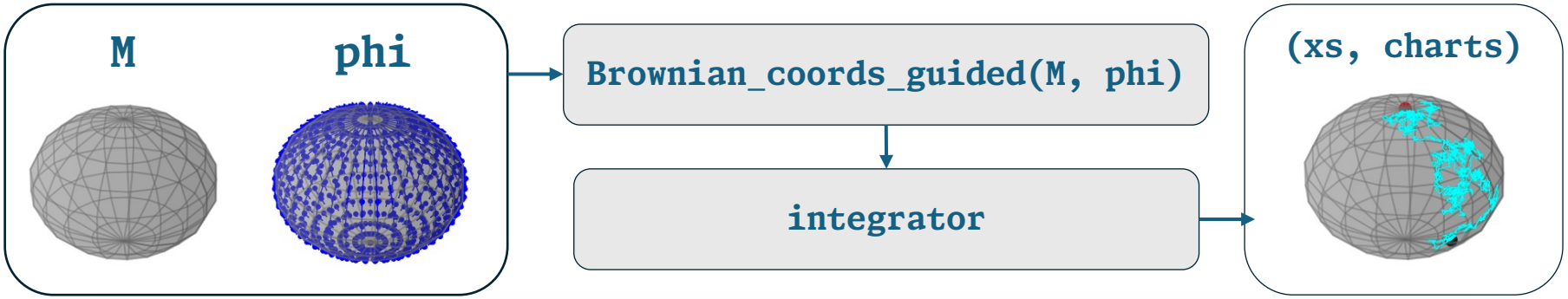


JAXGeometry
--Brownian motion in coordinates




```python
from jaxgeometry.stochastics import Brownian_coords
# set up the Brownian coordinates SDE
Brownian_coords.initialize(M)

# specify the time steps and discrete Wiener process for numerical simulation
_dts = dts(n_steps=1000)
_dWs = (M.dim, _dts)
(ts, xs, charts) = M.Brownian_coords(x, _dts, _dWs)
```

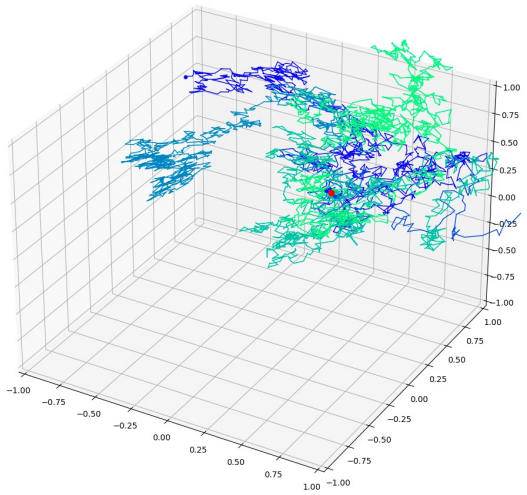
JAXGeometry
--Guided process [1]



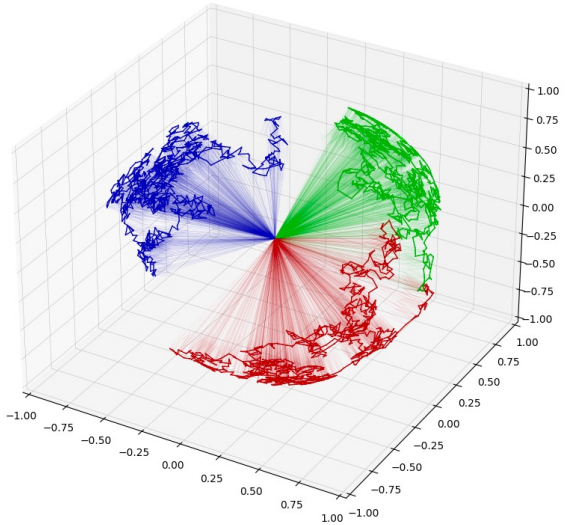

```python
from jaxgeometry.stochastics.guided_process import get_guided
# guided vector field
phi = ...

# get bridge SDE by Delyon/Hu's guided proposal
(Brownian_coords_guided, ...) = get_guided(M, phi, ...)

# specify the time steps and discrete Wiener process for numerical simulation
_dts = dts(n_steps=1000)
_dWs = dWs(M.dim, _dts)
(ts, xs, charts, ...) = Brownian_coords_guided(x, _dts, _dWs, ...)
```

[1] Delyon, B., & Hu, Y. (2006). Simulation of conditioned diffusion and application to parameter estimation. *Stochastic Processes and Their Applications*, *116*(11), 1660-1675. https://doi.org/10.1016/j.spa.2006.04.004

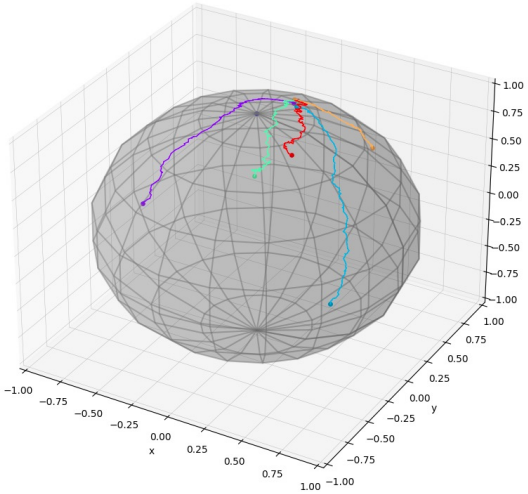
JAXGeometry
--Other Brownian processes



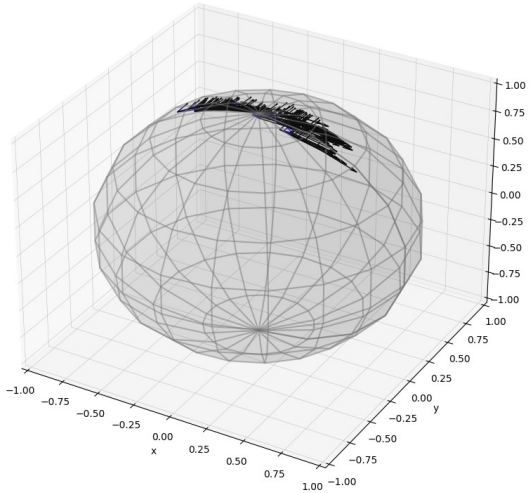
Brownian motion on Heisenberg groups
with `Brownian_sR` for sub-Riemannian manifold



Brownian motion on $SO(3)$
with `Brownian_inv` for invariant metrics



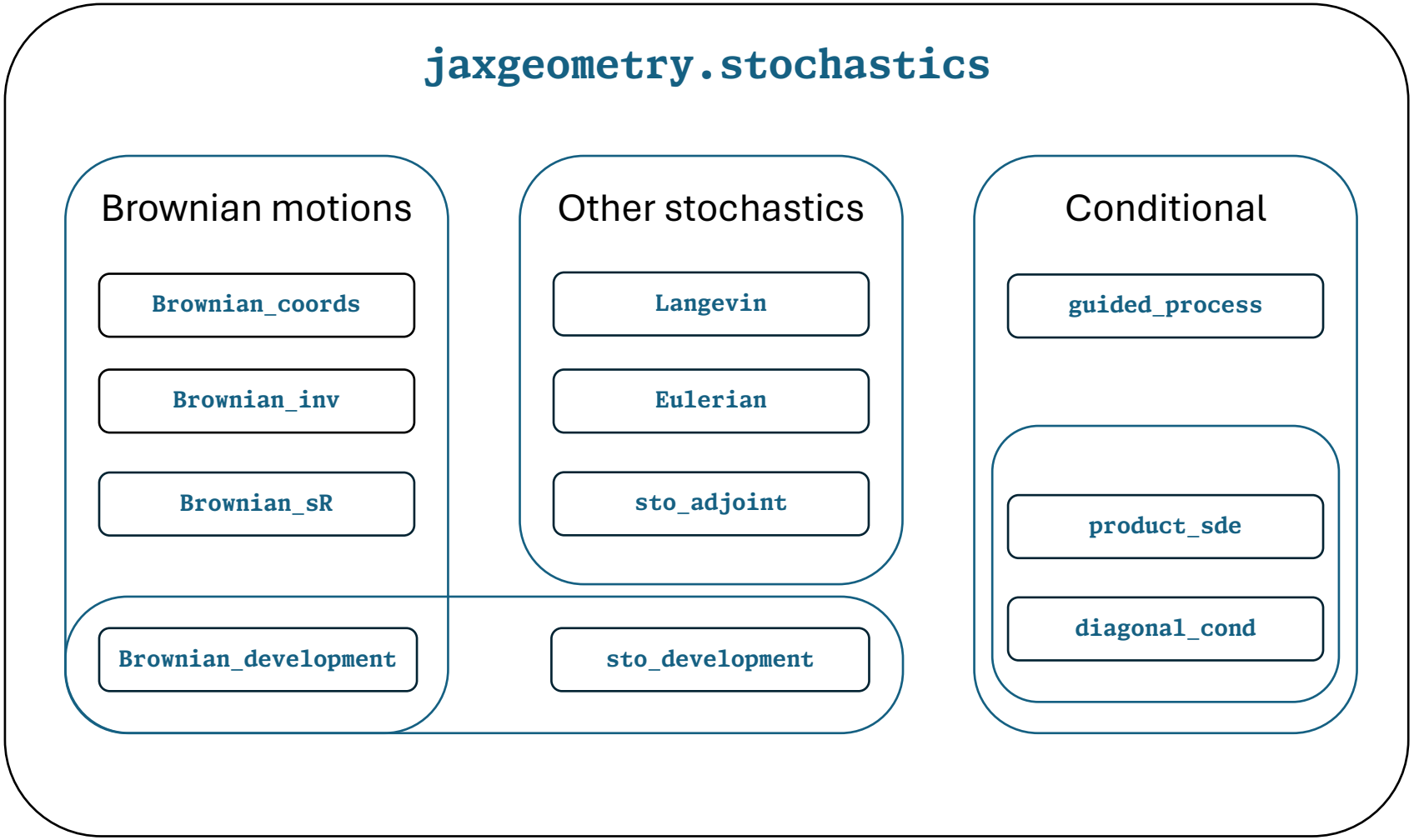
Mutiple conditional Brownians
with `diagonal_cond` on the product manifold



Brownian motion of the frame bundle
with `sto_development`

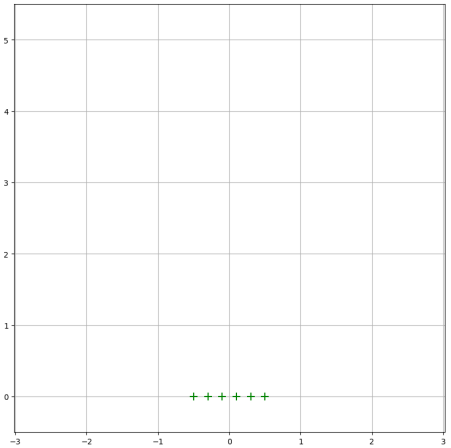
JAXGeometry
--Stochastic processes

With a defined metric, we can simulate stochastic processes on manifolds



JAXGeometry
--Landmarks and LDDMM

Euclidean space



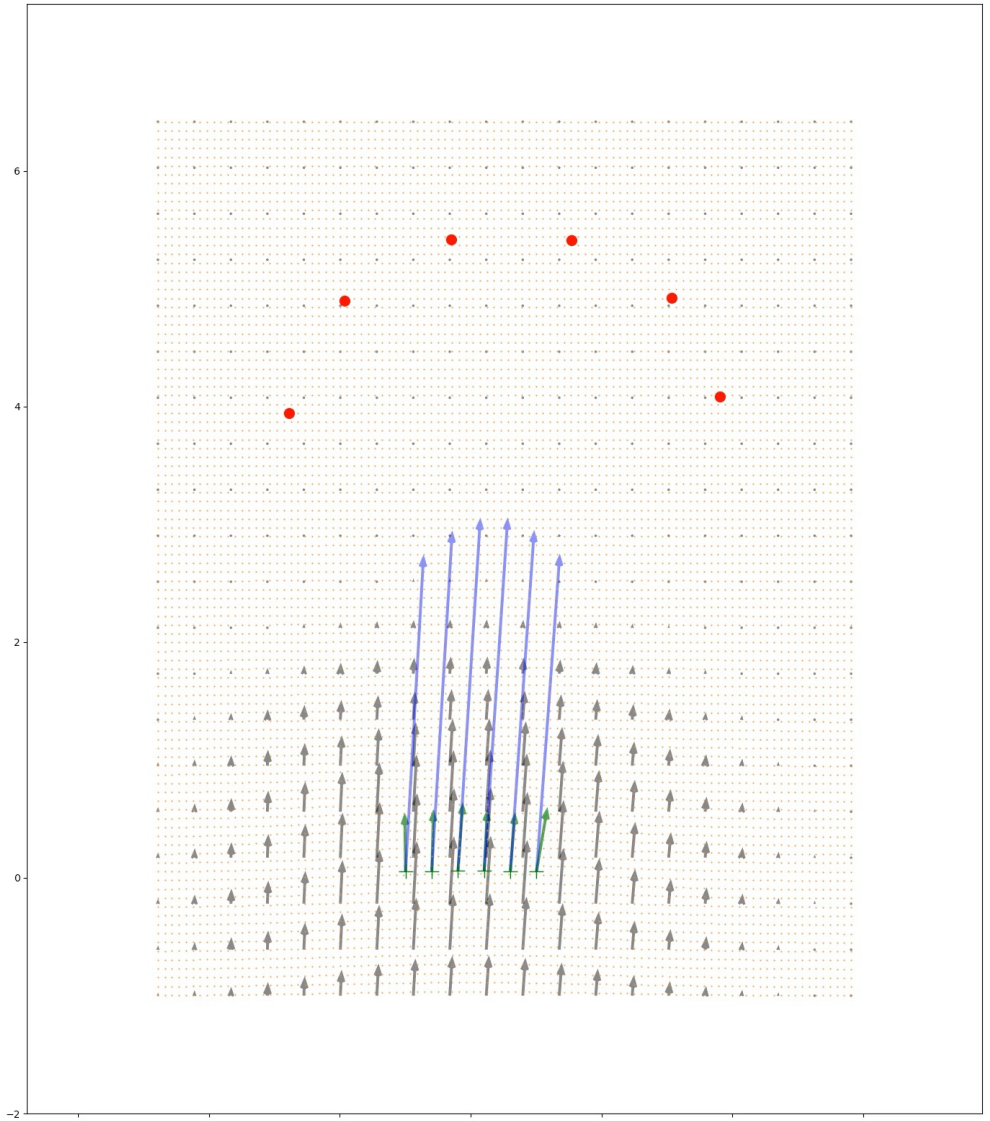
+

kernel-defined metric

$$g_{ij} = K^{-1}(\mathbf{x}_i, \mathbf{x}_j)$$

||

LDDMM manifold

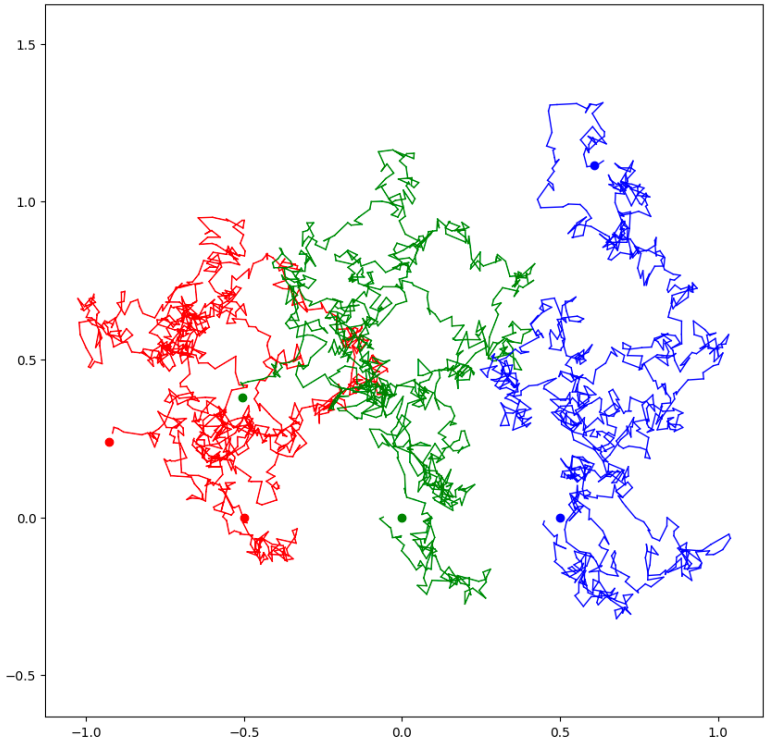


JAXGeometry
--Landmarks and LDDMM

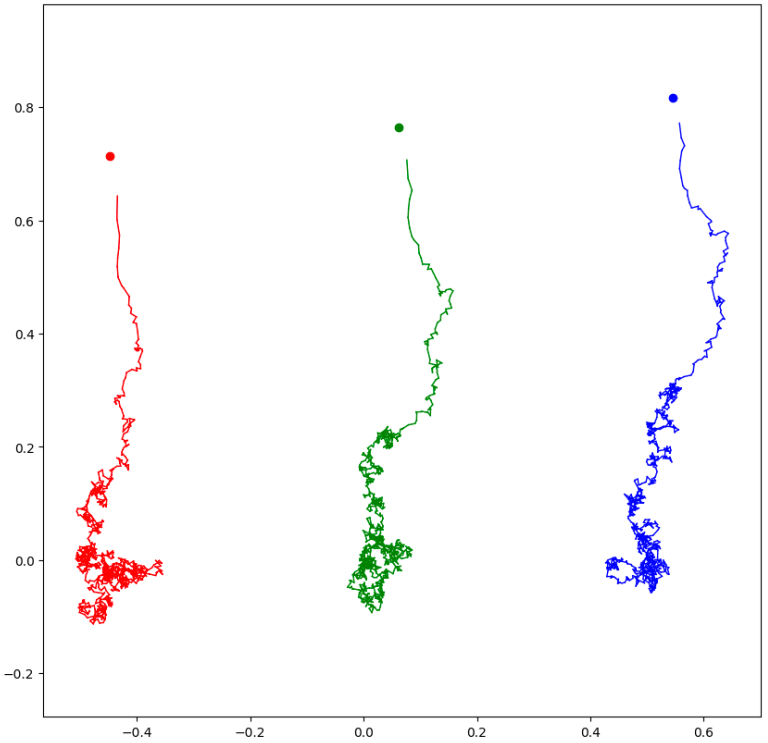
```
# jaxgeometry.dynamics.Hamiltonian

def initialize(M):
    dq = grad(M.H,argnums=1)                          # differential of q
    dp = lambda q,p: -gradx(M.H)(q,p)                 # differential of p

    def ode_Hamiltonian(c, y):                        # Hamiltonian equations
        t, x, chart = c
        dqt = dq((x[0],chart), x[1])
        dpt = dp((x[0],chart), x[1])
        return jnp.stack((dqt, dpt))

    def chart_update_Hamiltonian(xp,chart,y):         # chart update if needed
        if M.do_chart_update is None:
            return (xp, chart)

        p = xp[1]
        x = (xp[0], chart)

        update = M.do_chart_update(x)
        new_chart = M.centered_chart(x)
        new_x = M.update_coords(x,new_chart)[0]

        return (jnp.where(update,
                          jnp.stack((new_x,M.update_covector(x,new_x,new_chart,p))),
                          xp),
                jnp.where(update,
                          new_chart,
                          chart))
```

JAXGeometry
--Landmarks and LDDMM

Brownian motion of 3 landmarks



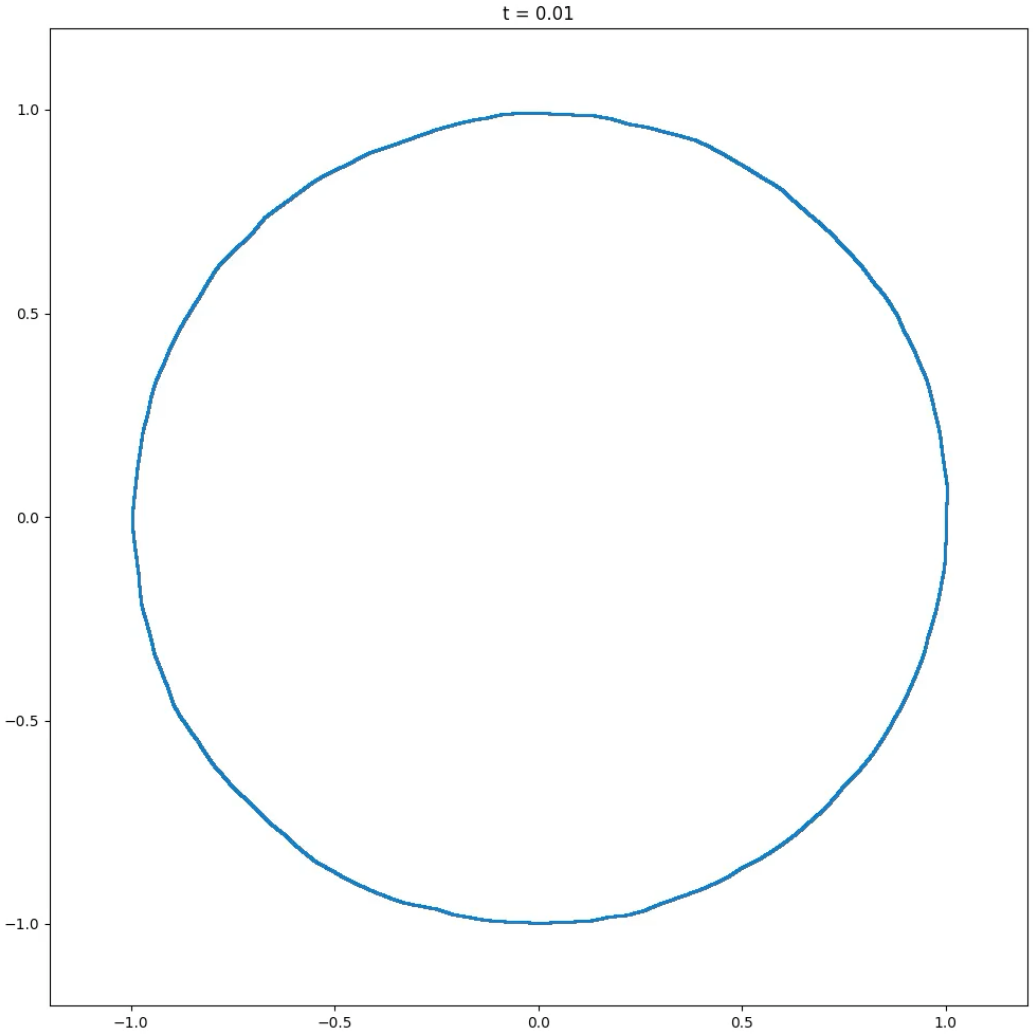
Guided bridge[1] of 3 landmarks



[1] Delyon, B., & Hu, Y. (2006). Simulation of conditioned diffusion and application to parameter estimation. *Stochastic Processes and Their Applications*, *116*(11), 1660-1675. https://doi.org/10.1016/j.spa.2006.04.004

JAXGeometry
--Landmarks and LDDMM

1,000,000 landmarks governed by the Eulerian SDE[1]



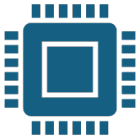
[1] Sommer, SH, Schauer, M & Meulen, FVD 2021, Stochastic flows and shape bridges. in *Statistics of Stochastic Differential Equations on Manifolds and Stratified Spaces (hybrid meeting).*

JAXGeometry
--Other modules

- Most probable path[1]
  - `jaxgeometry.dynamics.MPP_*.py`
- Diffusion mean[2]
  - `jaxgeometry.statistics.diffusion_mean.py`
- Tangent PCA
  - `jaxgeometry.statistics.tangent_PCA.py`
- …

[1] Grong, E., & Sommer, S. (2021). Most probable paths for anisotropic Brownian motions on manifolds. *ArXiv*. /abs/2110.15634
[2] Eltzner, B., Hansen, P., Huckemann, S. F., & Sommer, S. (2021). Diffusion Means in Geometric Spaces. *ArXiv*. /abs/2105.12061

Hyperiax



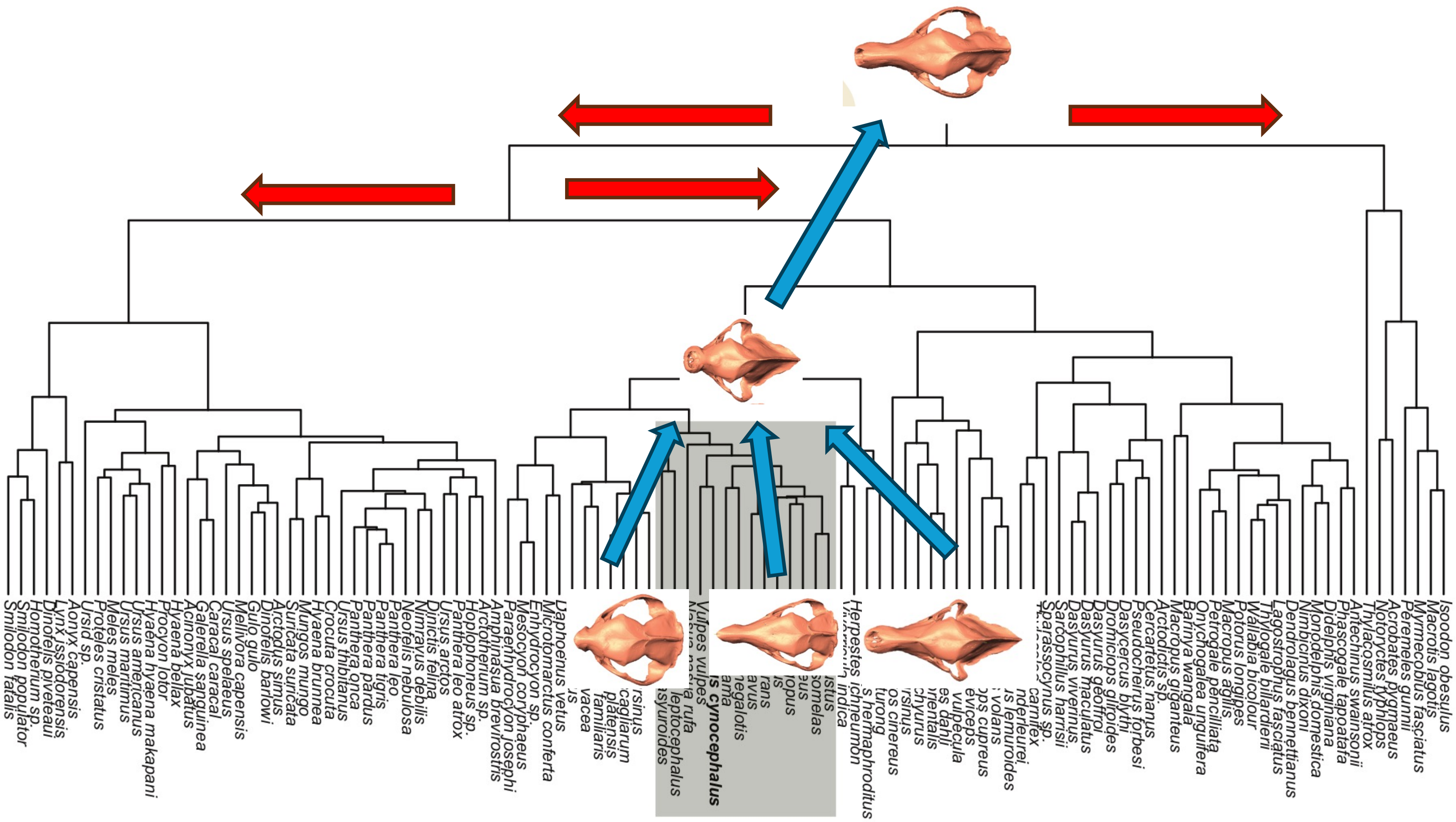
JAXGeometry
+
Tree traversal
+
JAX auto-vectorization/JIT compiling
=
Fast stochastic shape simulation and inference along the whole tree

high-performance stochastic shape evolution simulation
+
parameter inference on a phylogenetic tree
+
An easy-to-use tree simulation interface

Support:
Optimized tree execution
+
Customized execution functions
+
MCMC parameter inference

Hyperiax
--Tree level execution (BFFG)[1]

Incorporate leaf observations x_{V_T} into probablistic model $p(X_t|x_{V_T})$



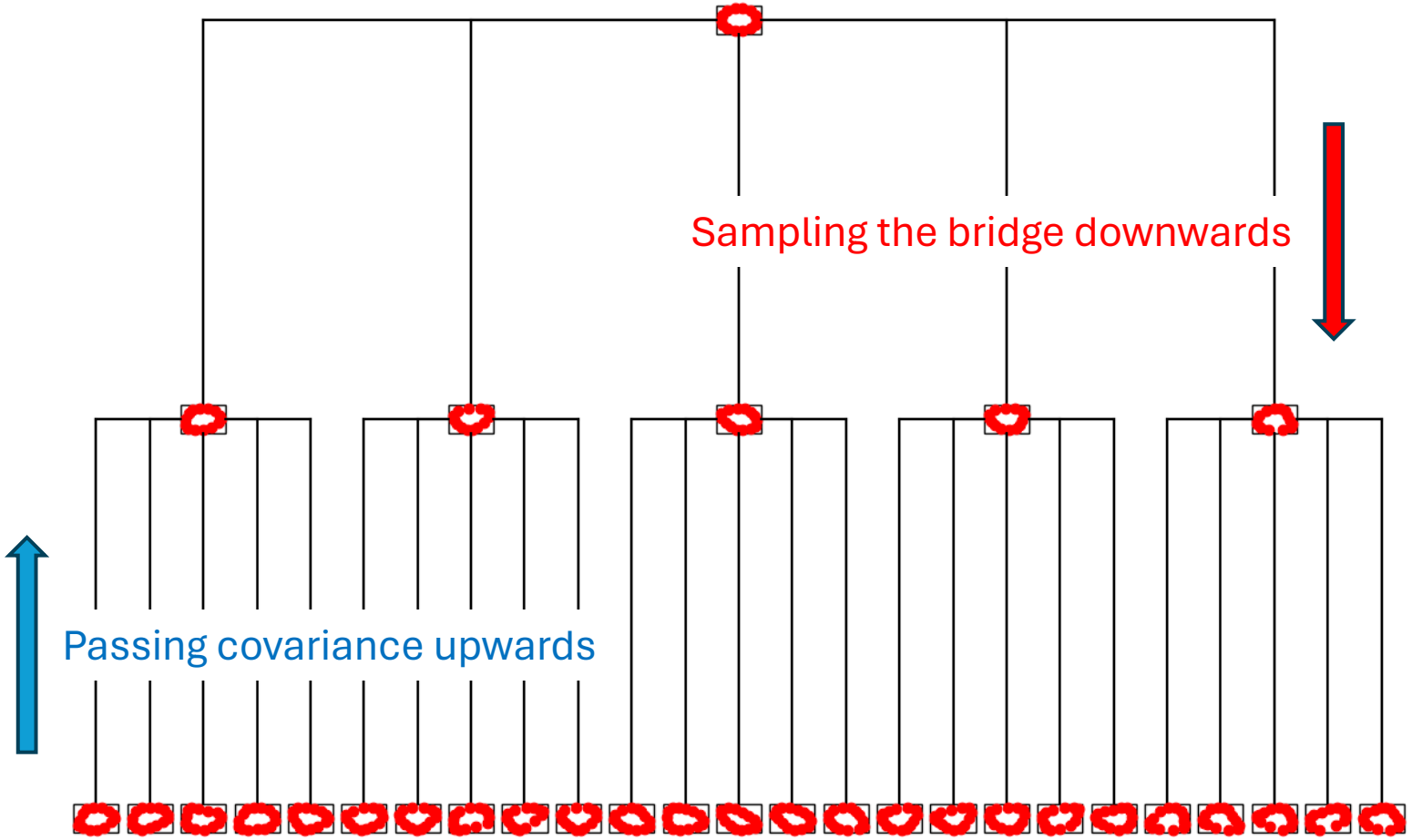
ⓒ Sommer

Upwards message passing

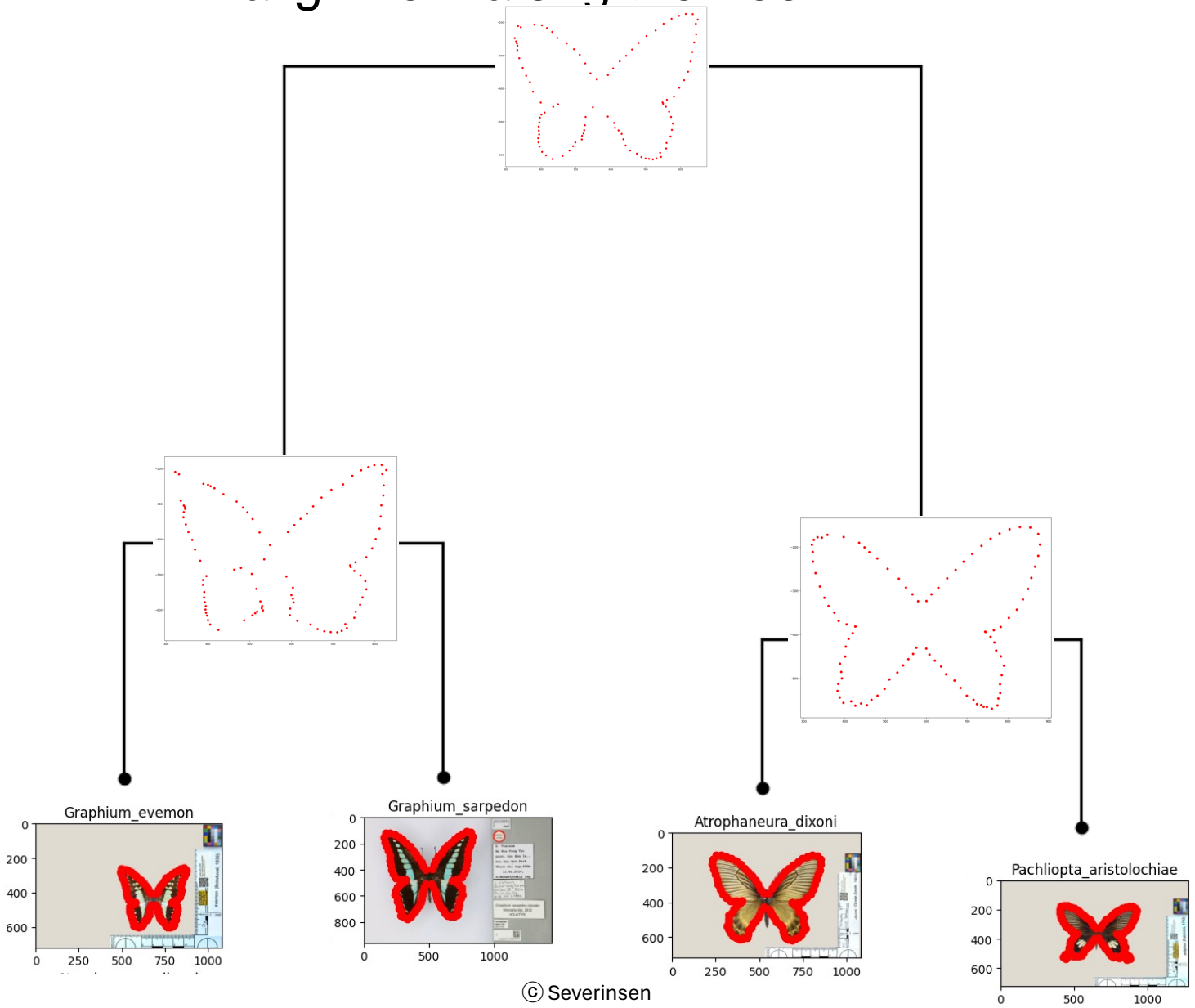
Downwards conditional sampling

[1] van der Meulen, F. (2022). Introduction to Automatic Backward Filtering Forward Guiding. *ArXiv*. /abs/2203.04155

Hyperiax
--Simulating shape processes on the tree



Hyperiax
--LDDMM alignment along the tree



Hyperiax
--What's next

- Support larger trees (millions of nodes)
- Differentiate the tree using autodiff for backpropagations
- On the Phylogenetic tree with real biological shape data

Public links

- JAXGeometry: https://github.com/ComputationalEvolutionaryMorphometry/jaxgeometry
- Hyperiax: https://github.com/ComputationalEvolutionaryMorphometry/hyperiax

Thank you for your listening!