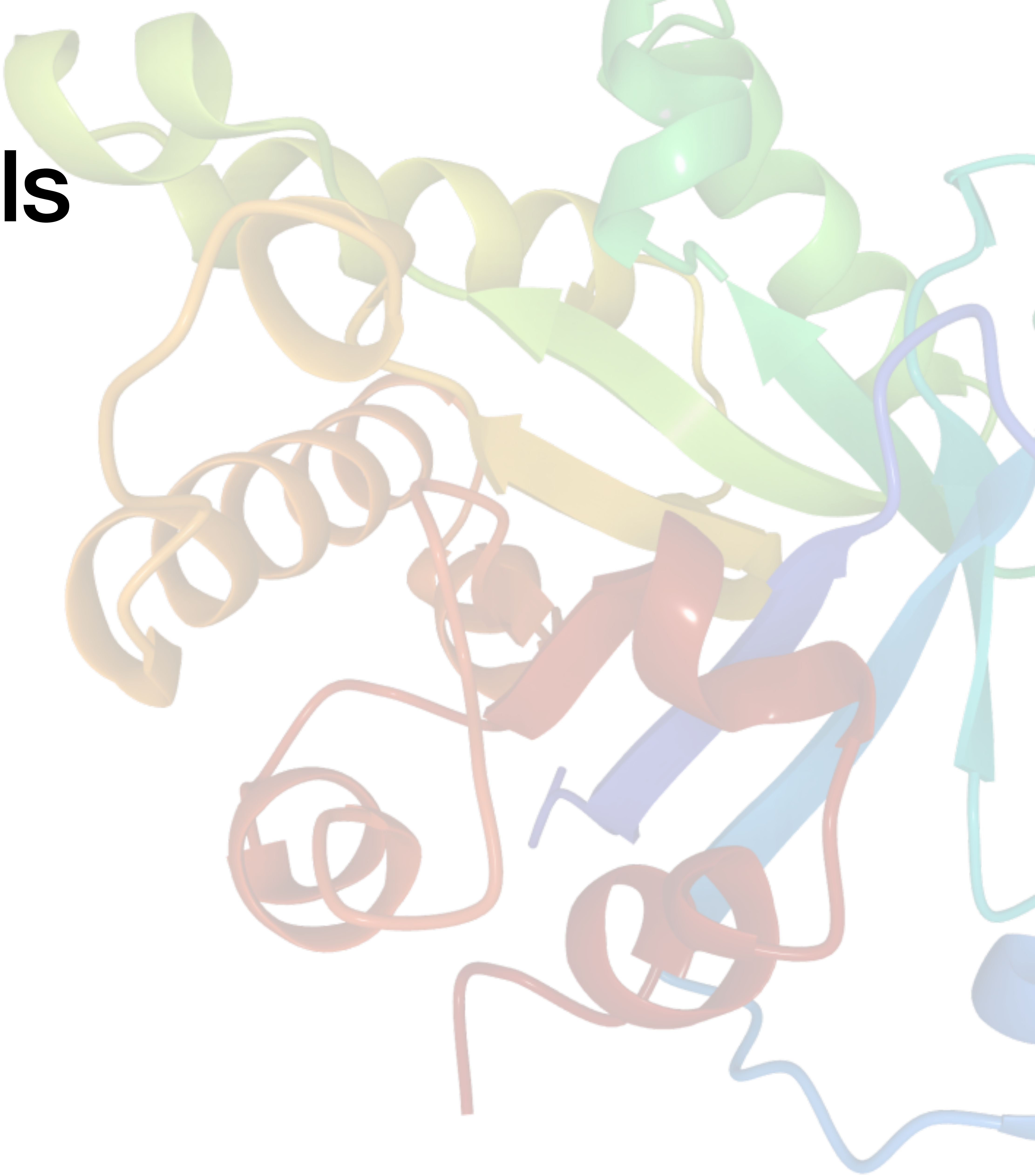
Geometric Generative Models

Joey Bose, Oscar Davis

ÖAW AI Winter School 2025

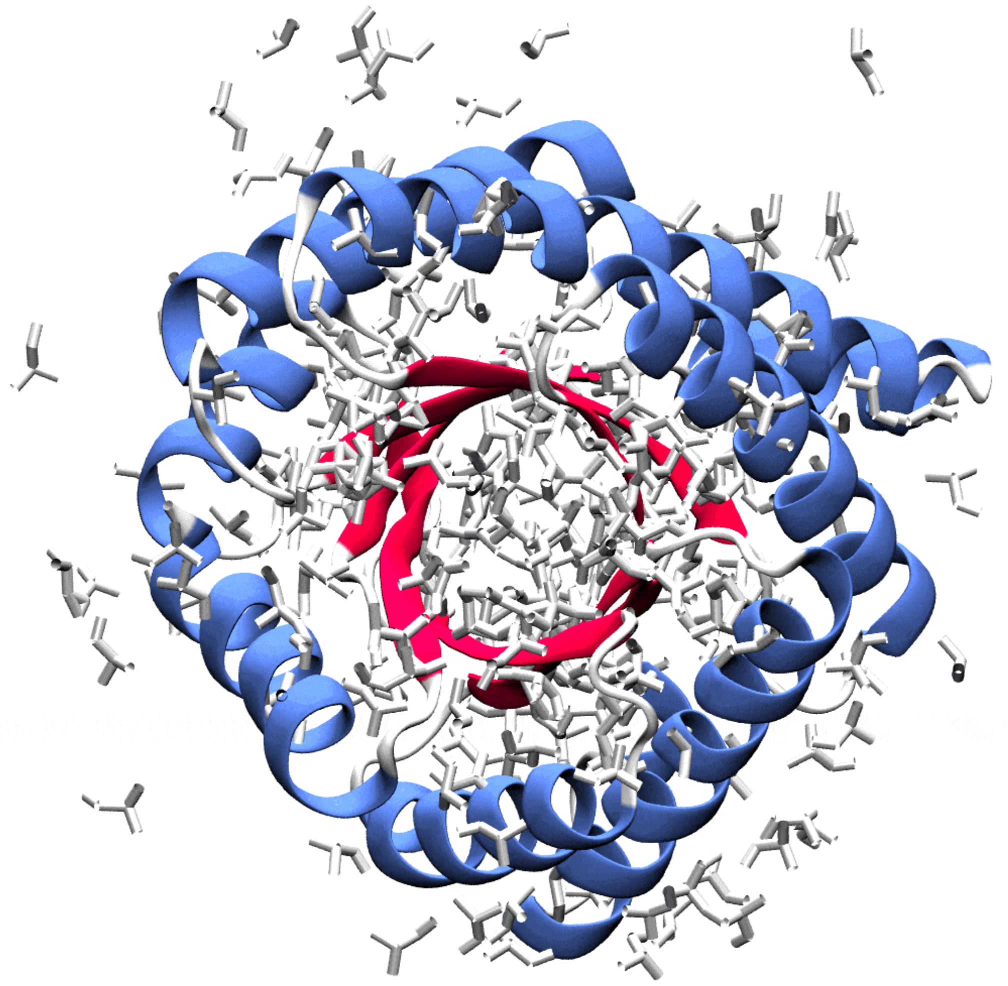


UNIVERSITY OF OXFORD

Mila

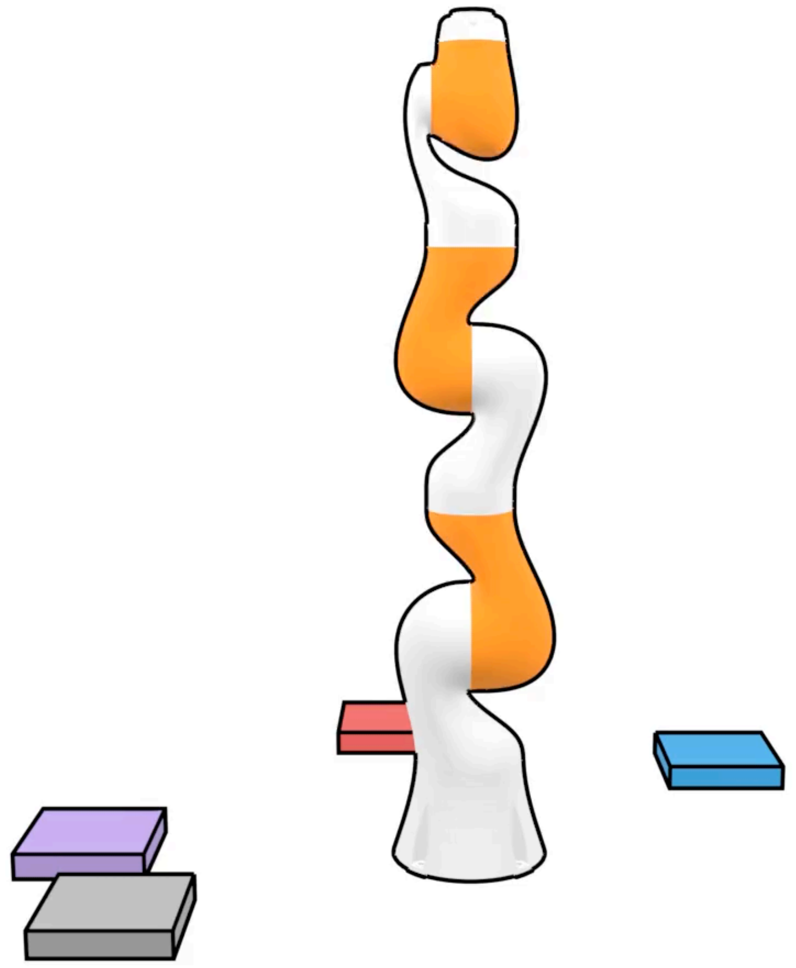
Generative Models Beyond Images and Text

Scientific Data



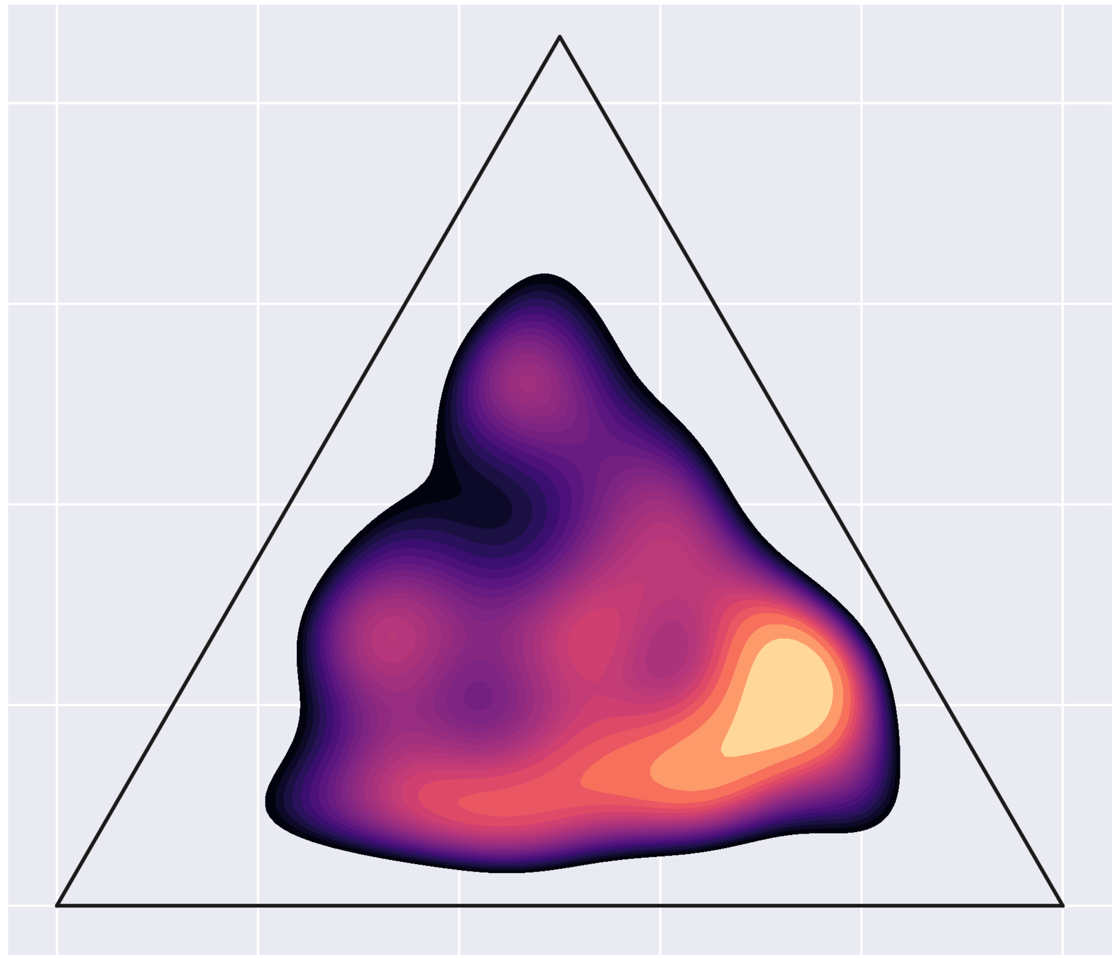
SE(3) invariant
Protein structure generation

Robotics



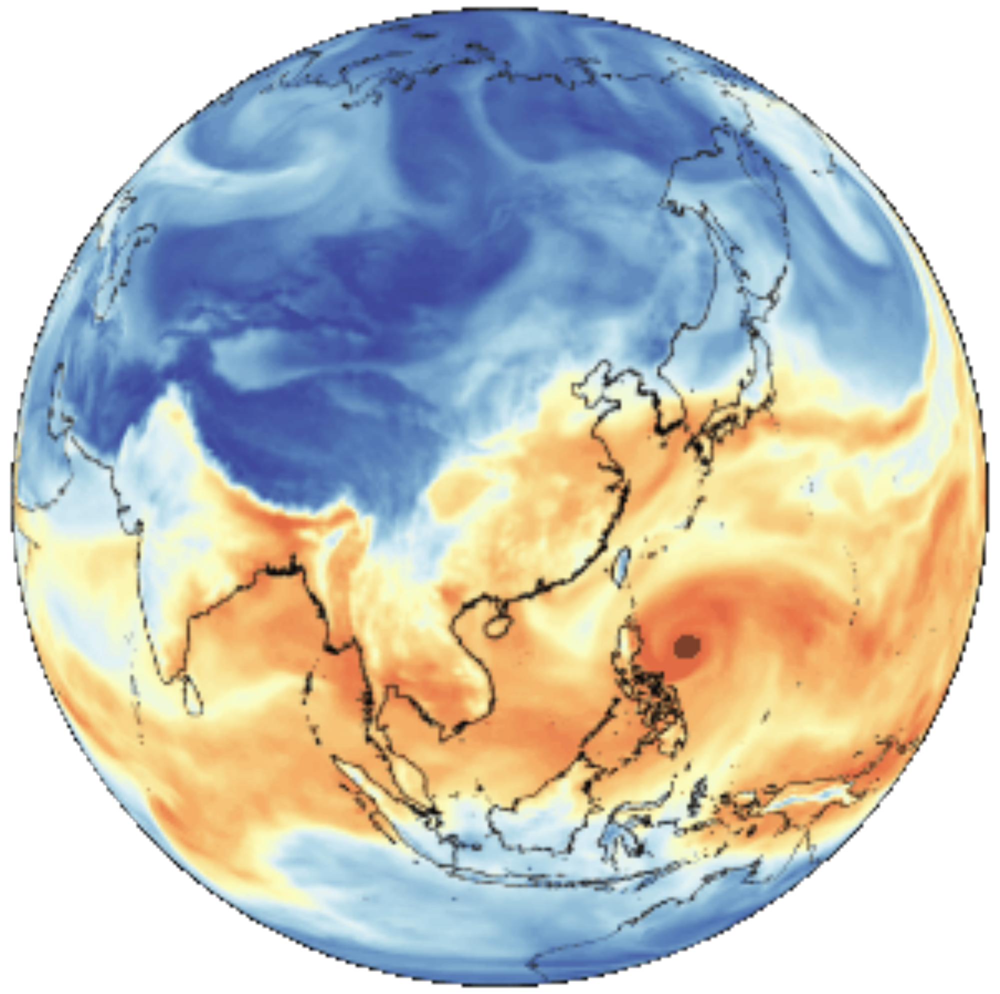
SO(2) invariant
Block stacking

Information Geometry

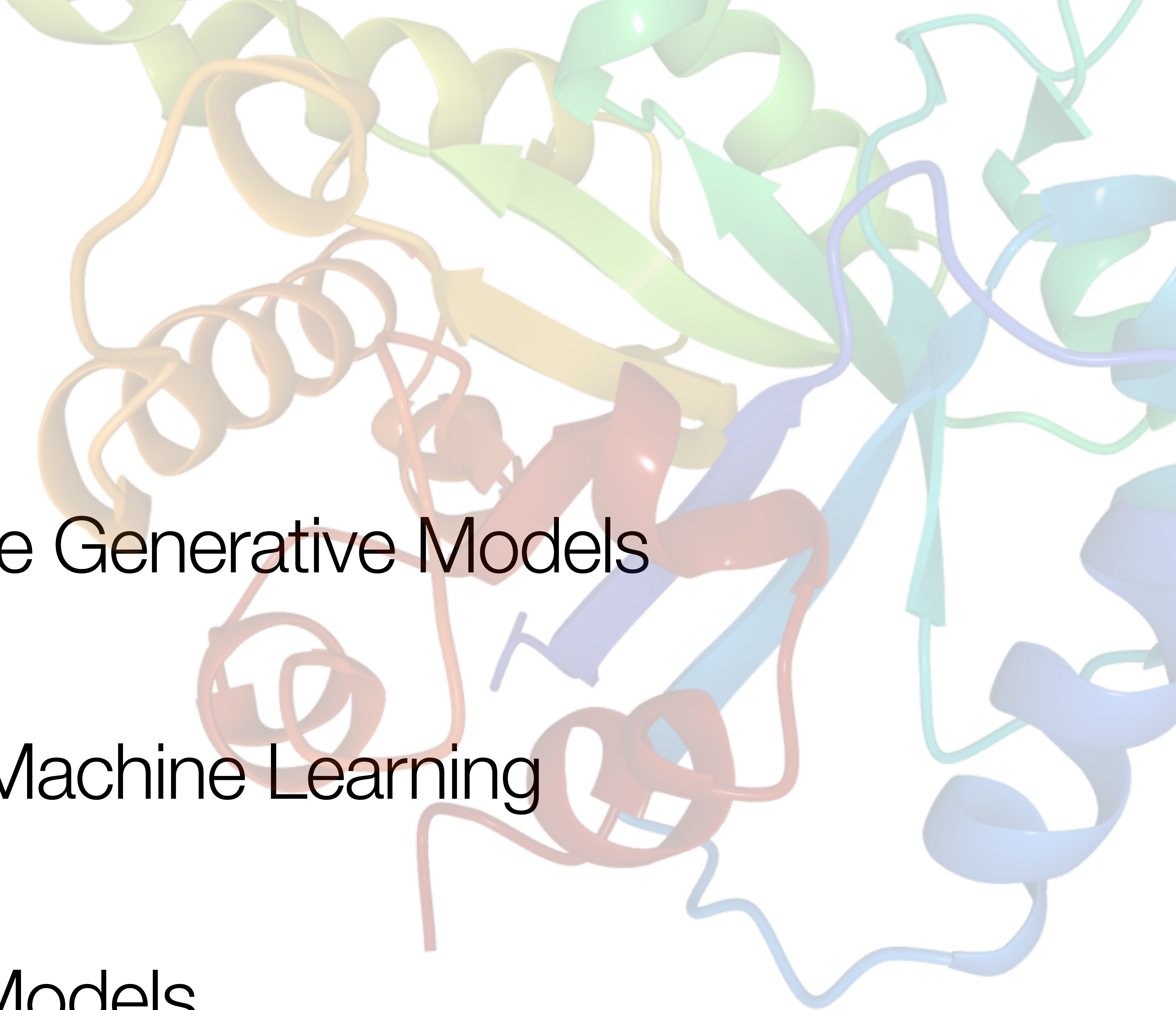


Fisher-Rao geometry
On the probability Simplex

Climate Modeling



Spherical Geometry $\mathbb{S}^2$



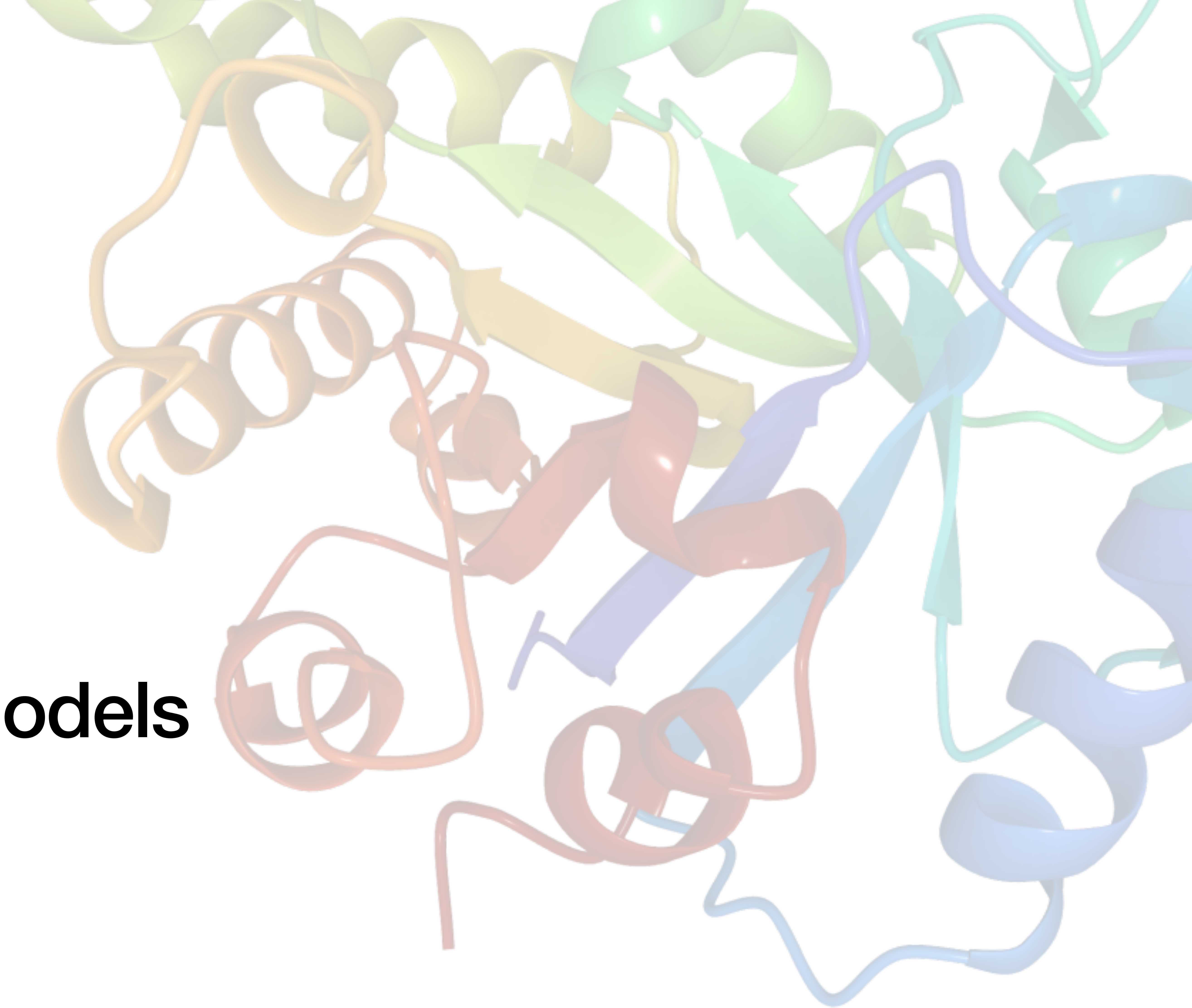
Tutorial Outline ~3hrs:

Part I: Primer on Simulation-Free Generative Models

Part II: Primer of Geometry for Machine Learning

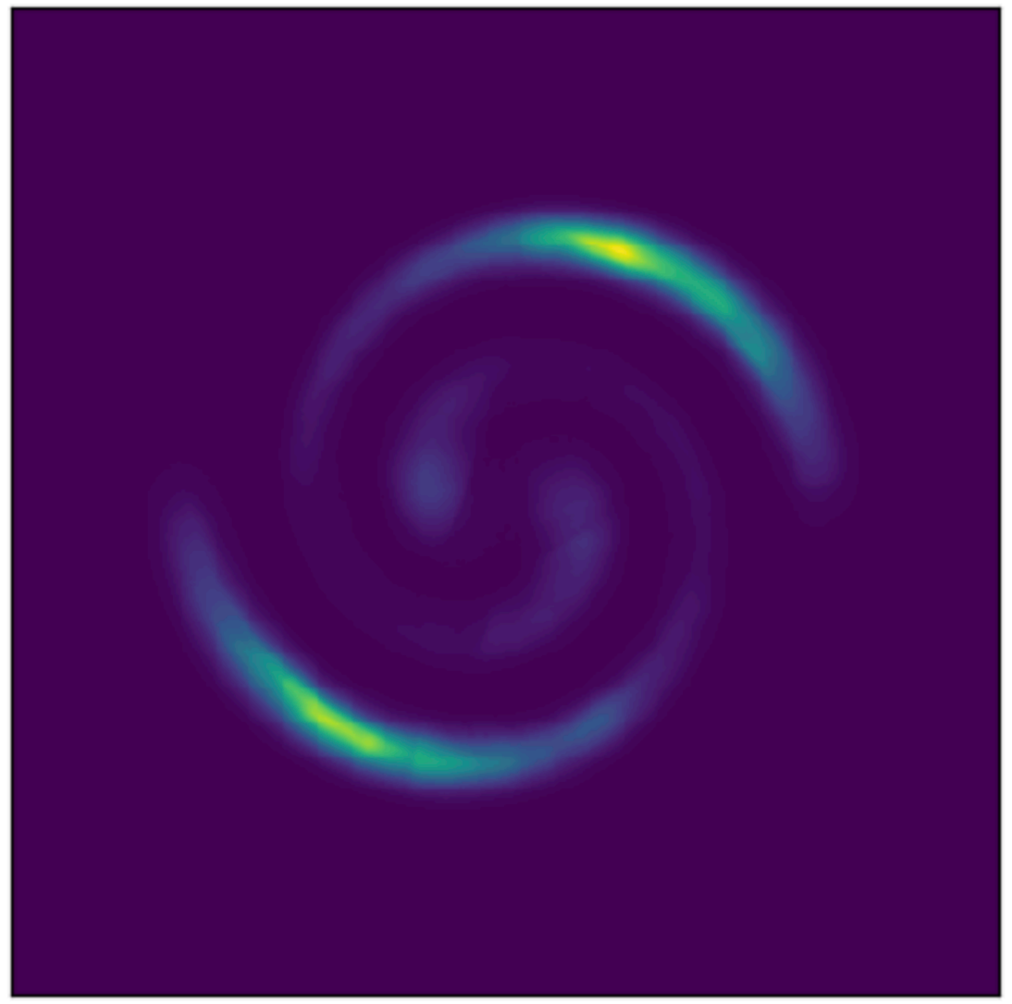
Part III: Geometric Generative Models

Part I:
Simulation Free Generative Models



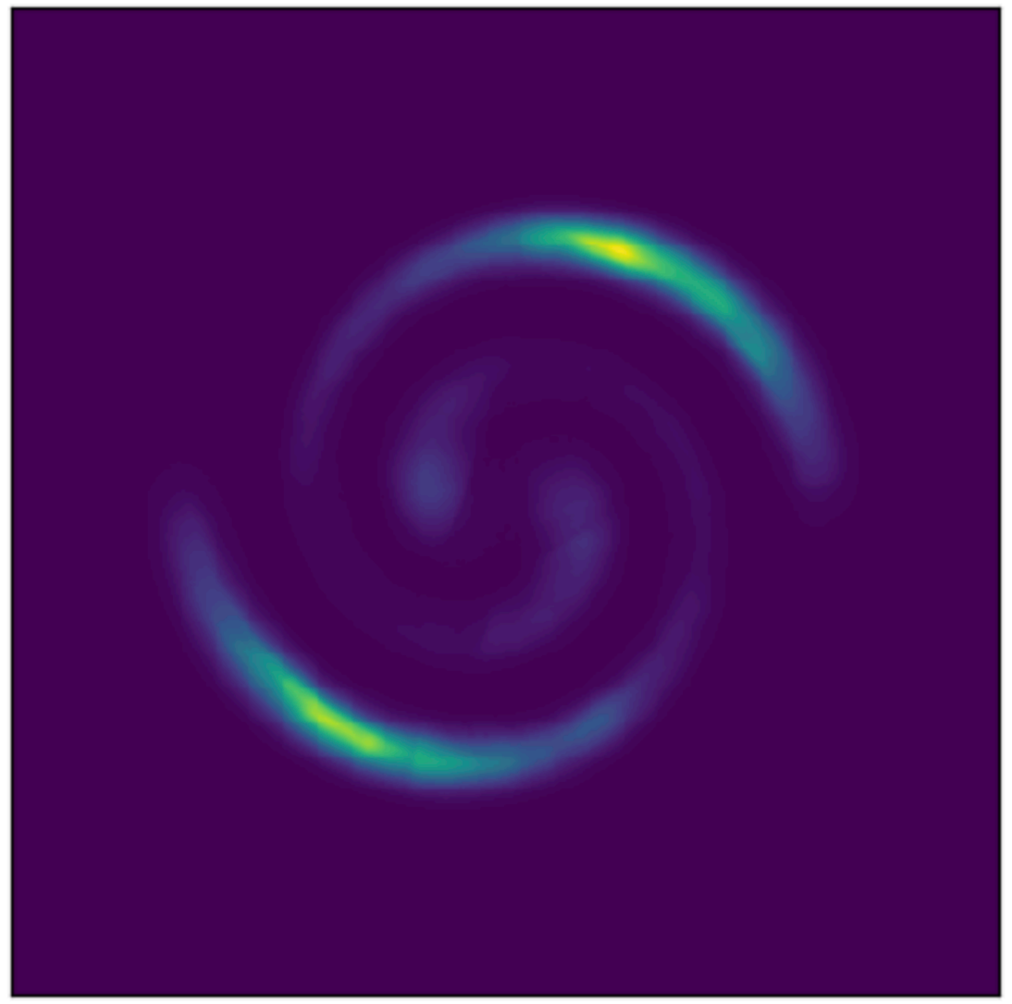
Problem Setting: Generative Modeling

- Unknown: data distribution q



Problem Setting: Generative Modeling

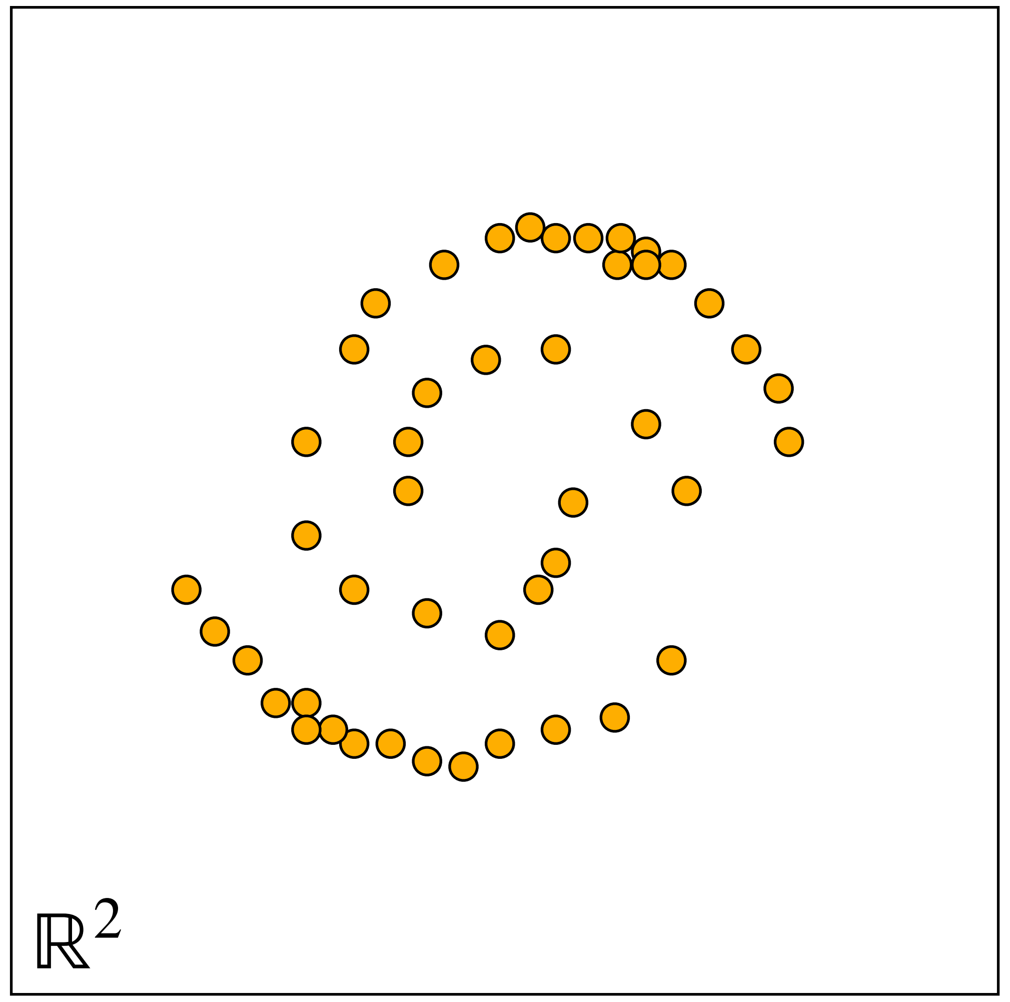
- Unknown: data distribution q
- Given: samples $x_1 \sim q$



Problem Setting: Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$

$x_1 \sim q$

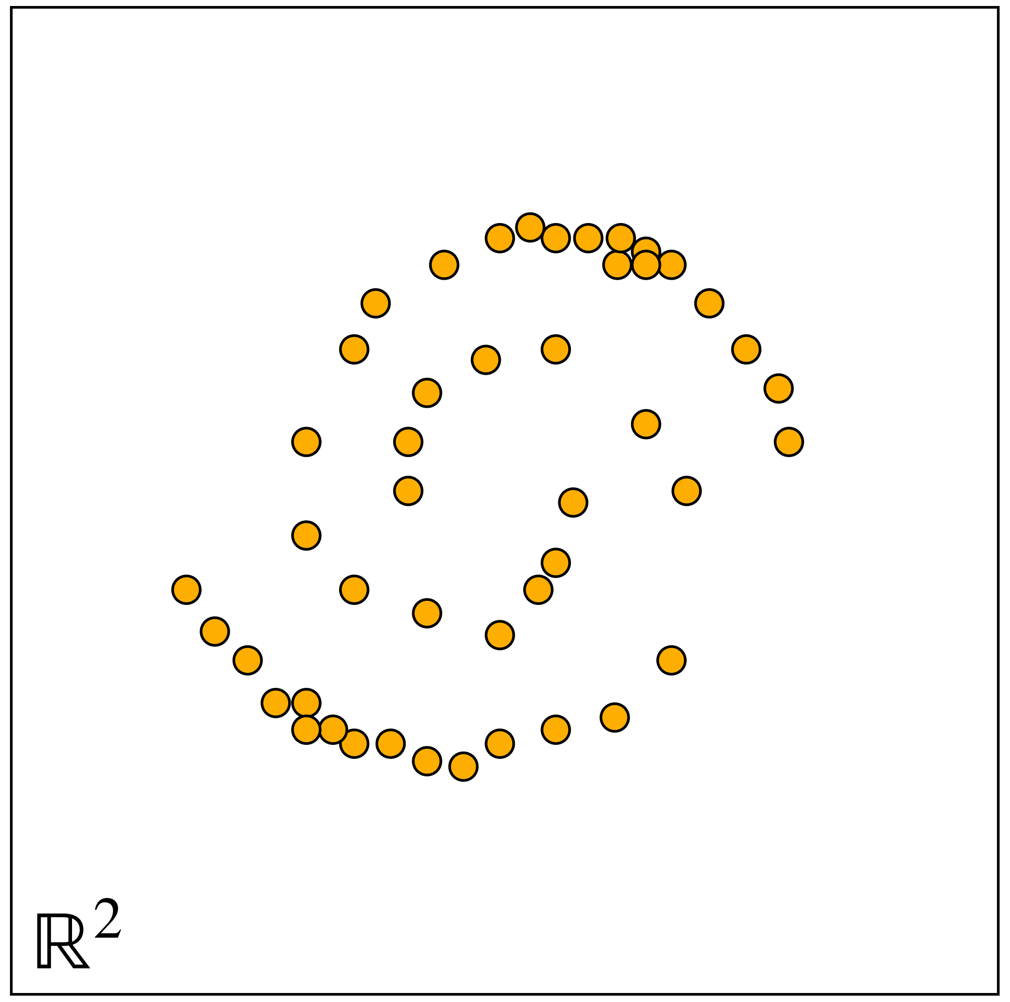


Problem Setting: Generative Modeling

- Unknown: data distribution q

- Given: samples $x_1 \sim q$

$x_1 \sim q$



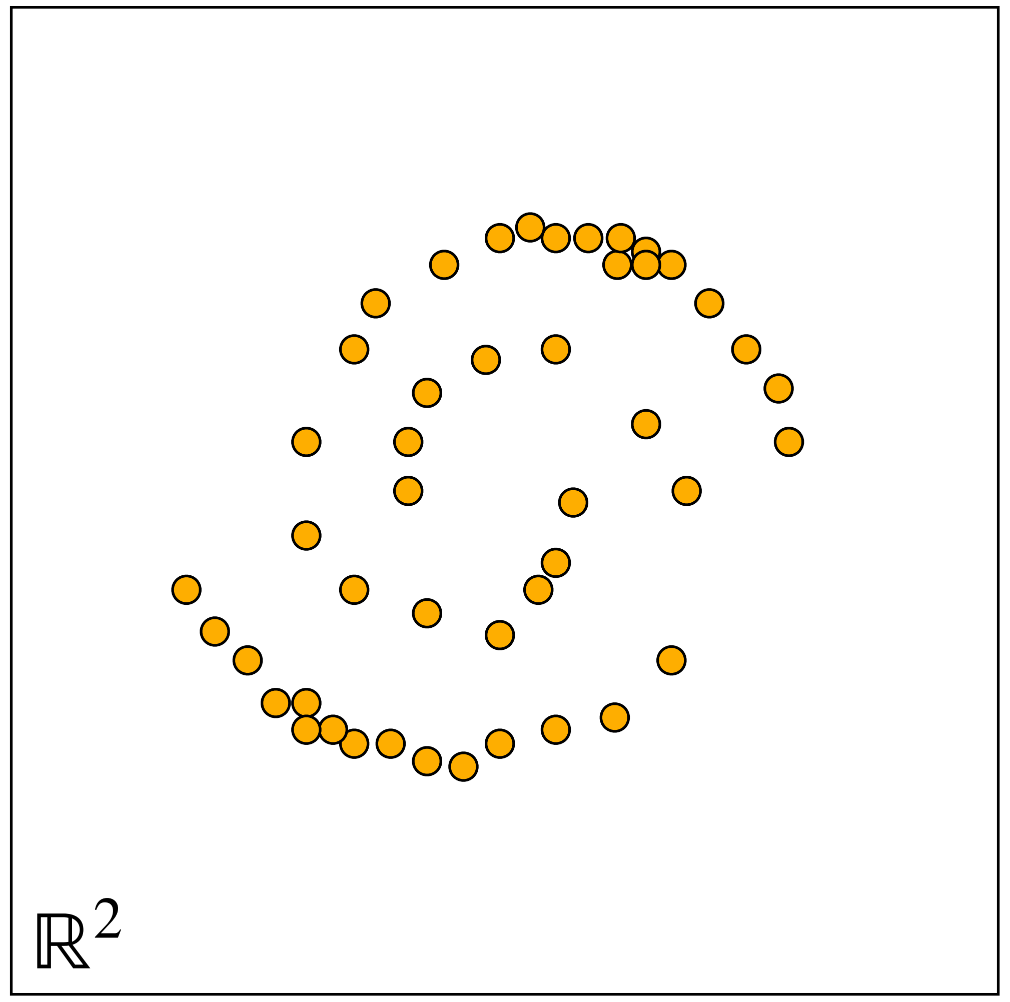
Goal: learn *a sampler* from the unknown q

Deep Generative Modeling

- Unknown: data distribution q

- Given: samples $x_1 \sim q$

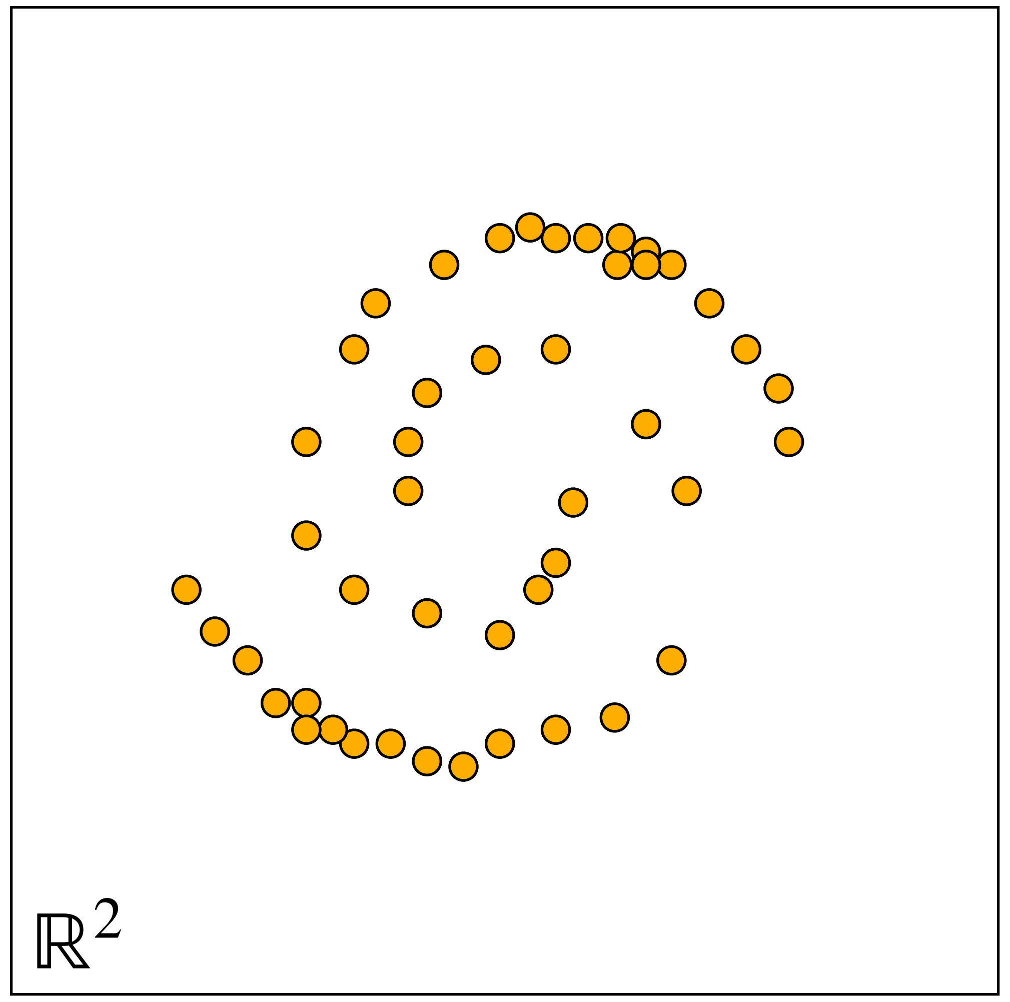
$x_1 \sim q$



Deep Generative Modeling

- Unknown: data distribution q
- Given: samples $x_1 \sim q$
- Learn: neural network with parameters θ

$x_1 \sim q$



Generative Model

$(\, \psi_\theta \,, p_\theta \,)$

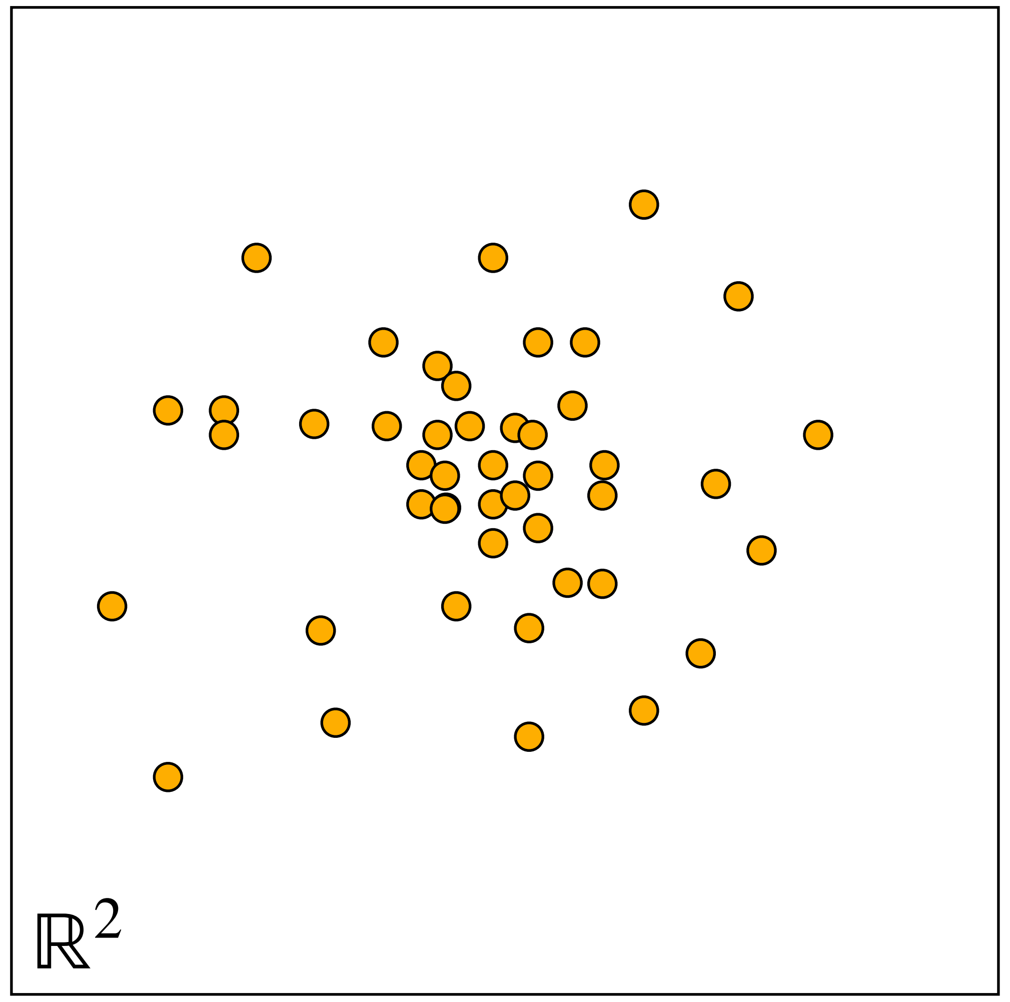
Generator   Underlying Density

Goal: find parameters θ s.t. $p_\theta \approx q$

Deep Generative Modeling

Easy to sample from

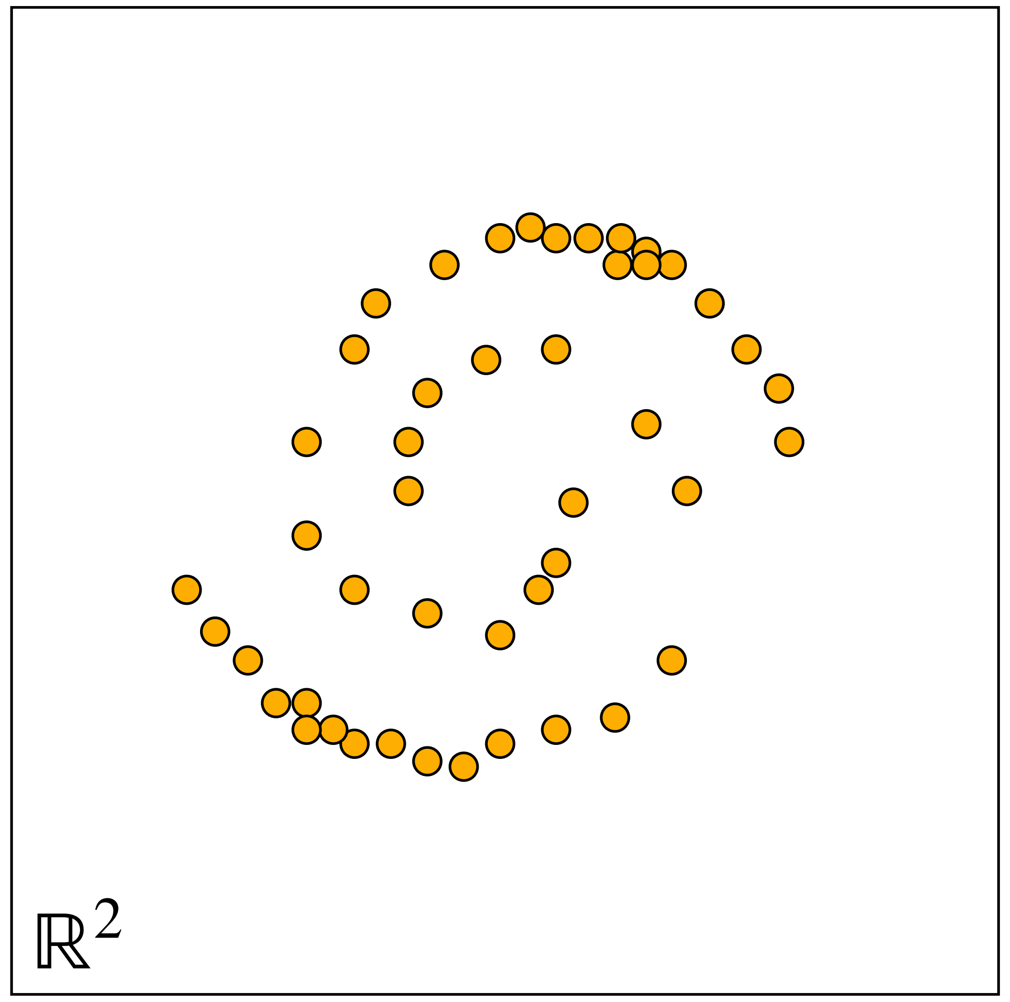
$x_0 \sim p$



Generator

ψ_θ

$x_1 \sim q$

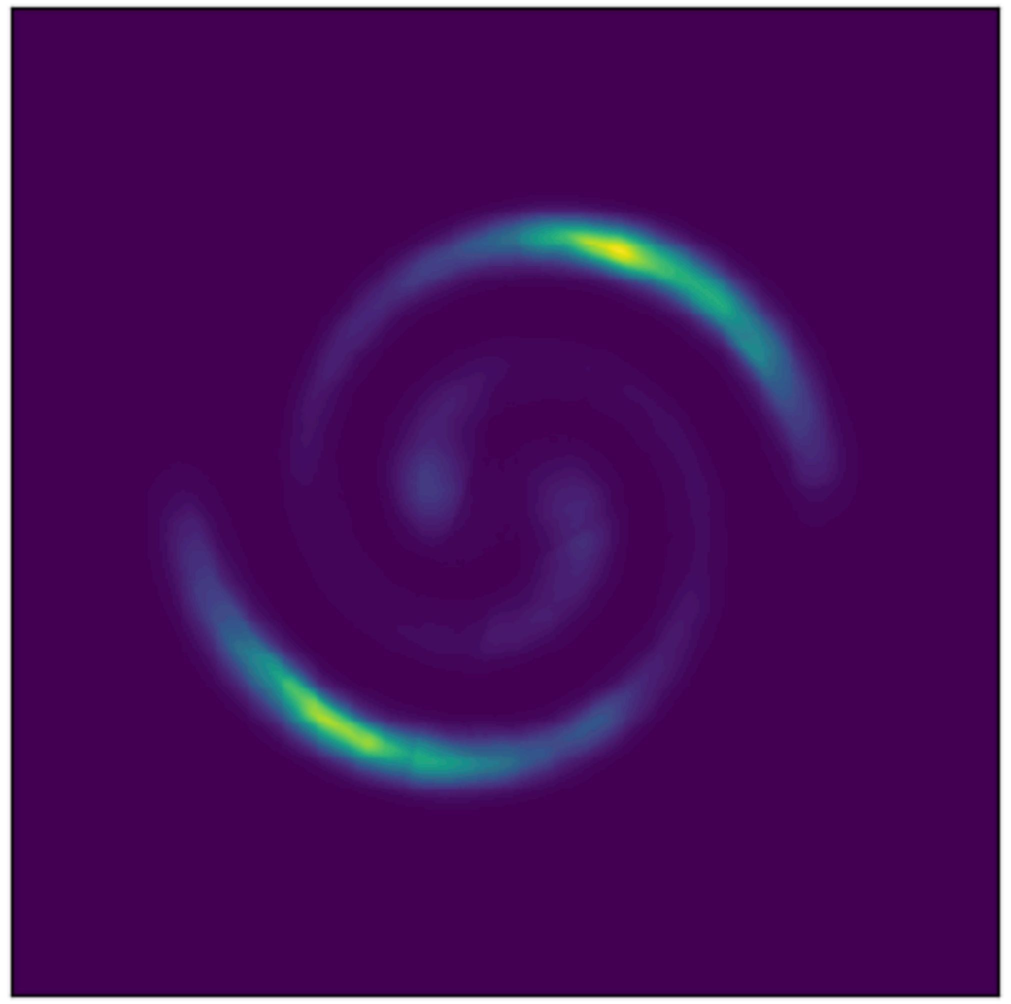


Sampling

$x_0 \sim p$

$\psi_\theta(x_0) \sim q$

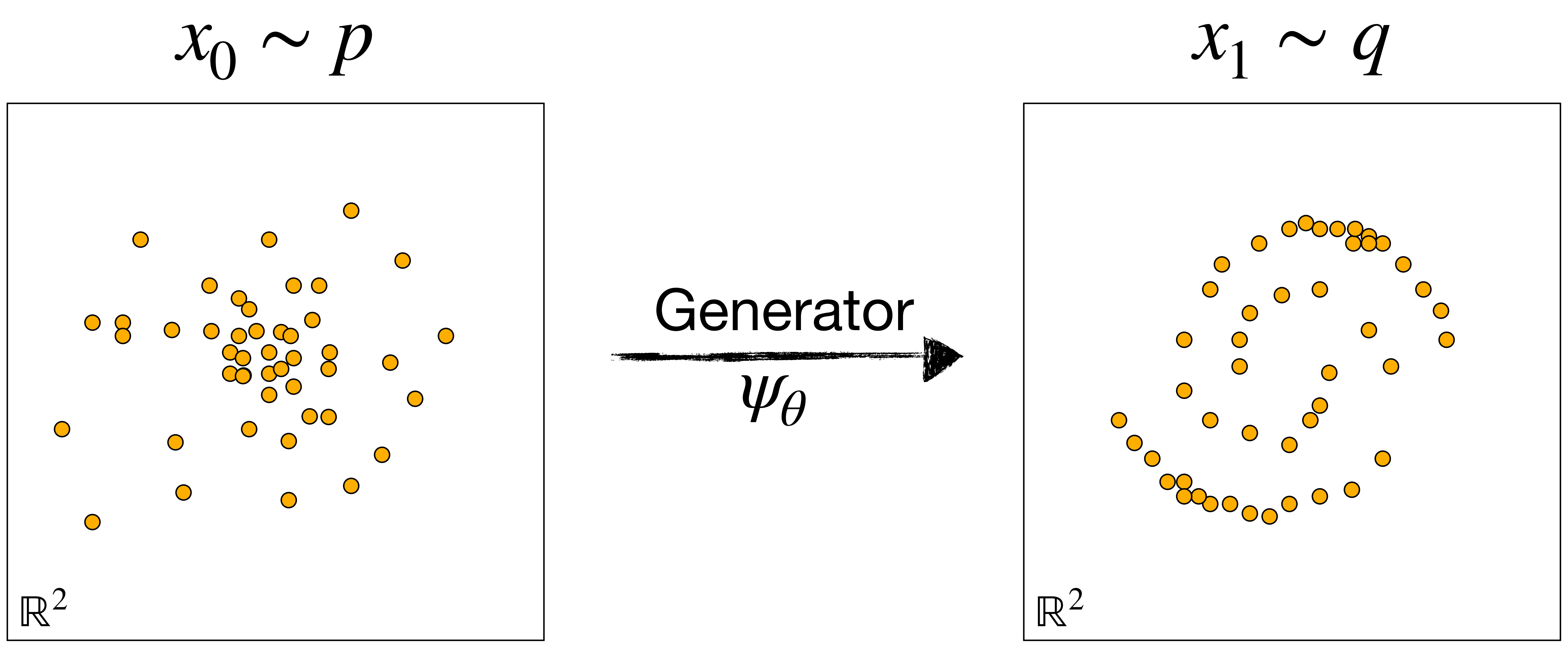
How to model $\boldsymbol{\psi_\theta}$?



Density Estimation

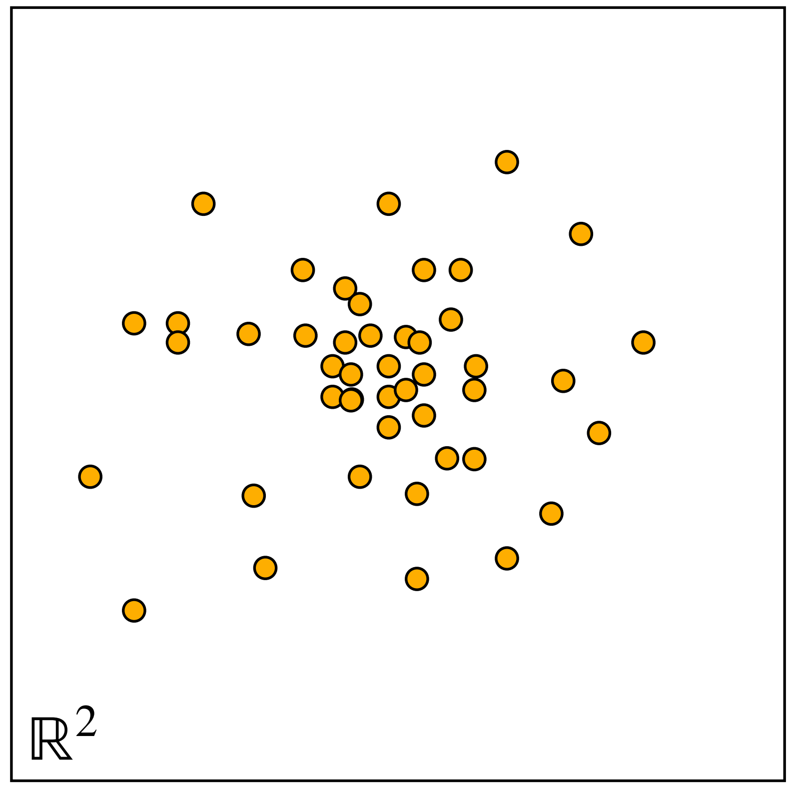
$p_\theta \approx q$

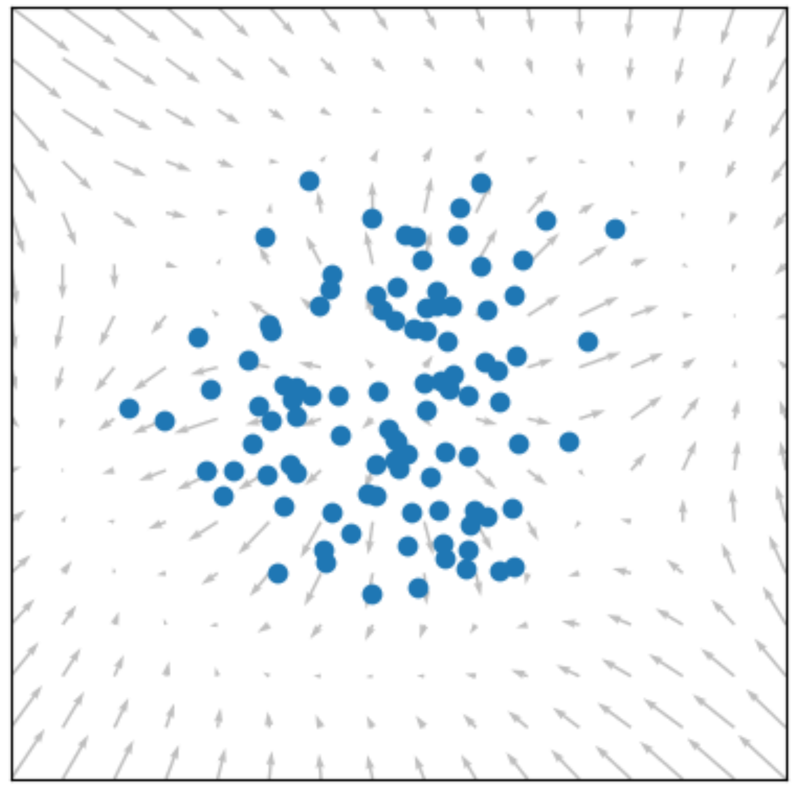
Focus: Dynamical Systems as Generative Models



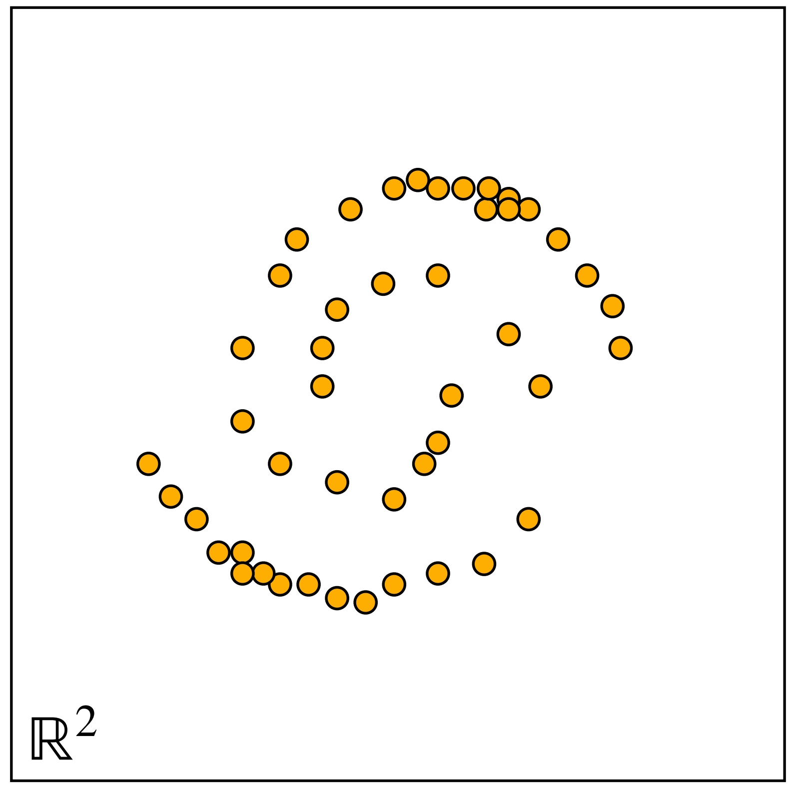
Focus: Dynamical Systems as Generative Models

$x_0 \sim p$





$x_1 \sim q$



$$\psi_t : [0,1] \times \mathbb{R}^2 \to \mathbb{R}^2$$

Time-Dependent Generator

Sampling = Simulating

$x_0 \sim p \longrightarrow \text{simulate}(x_0, t) = \psi_t(x_0)$

Deterministic and Stochastic Dynamics

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

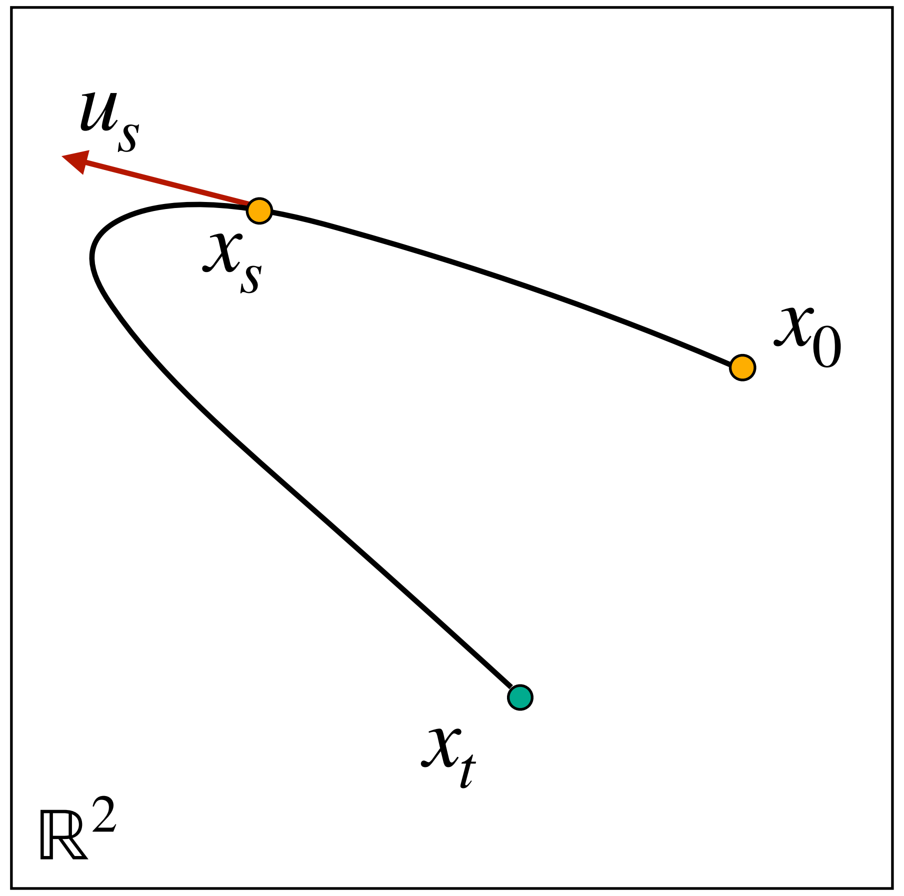
Deterministic and Stochastic Dynamics

Flows

ODE $dx_t = u_t(x_t)dt$

Velocity field

Deterministic



$$x_t = x_0 + \int_0^t u_s(x_s)ds$$

Diffusion

SDE $dx_t = f_t(x_t)dt + g_t dw_t$

Drift

Diffusion Coefficient

Brownian Motion

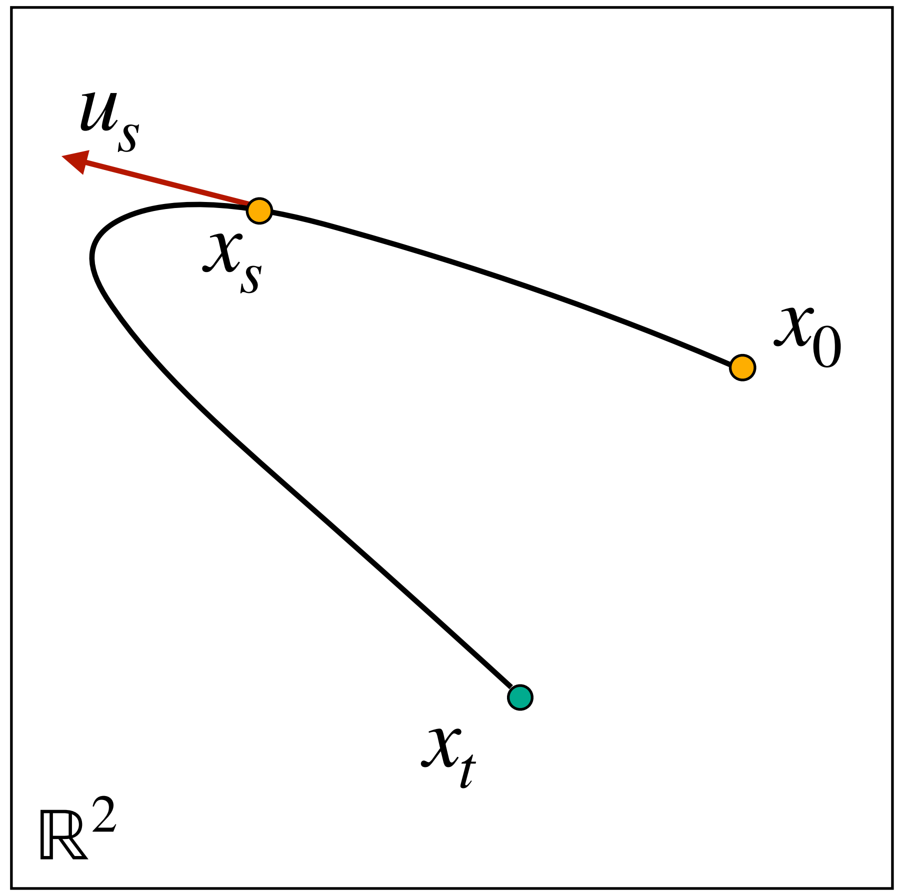
Deterministic and Stochastic Dynamics

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Deterministic



$$x_t = x_0 + \int_0^t u_s(x_s)ds$$

Diffusion

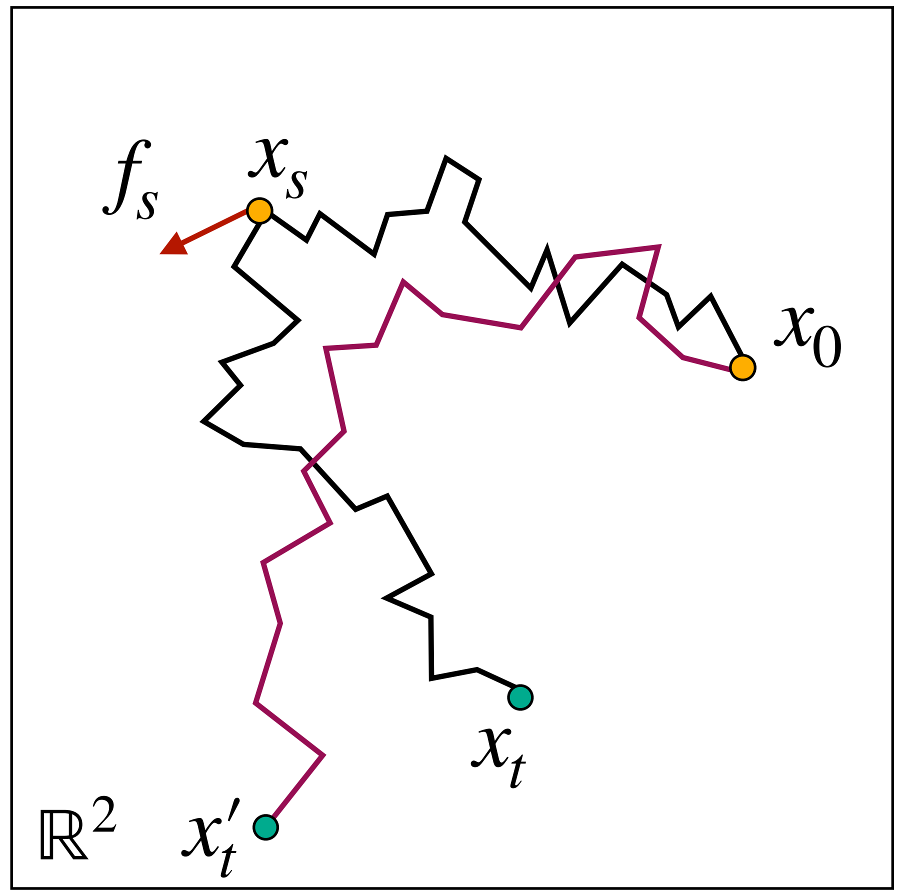
SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

Stochastic



$$x_t = x_0 + \int_0^t f_s(x_s)ds + \int_0^t g_s dw_s$$

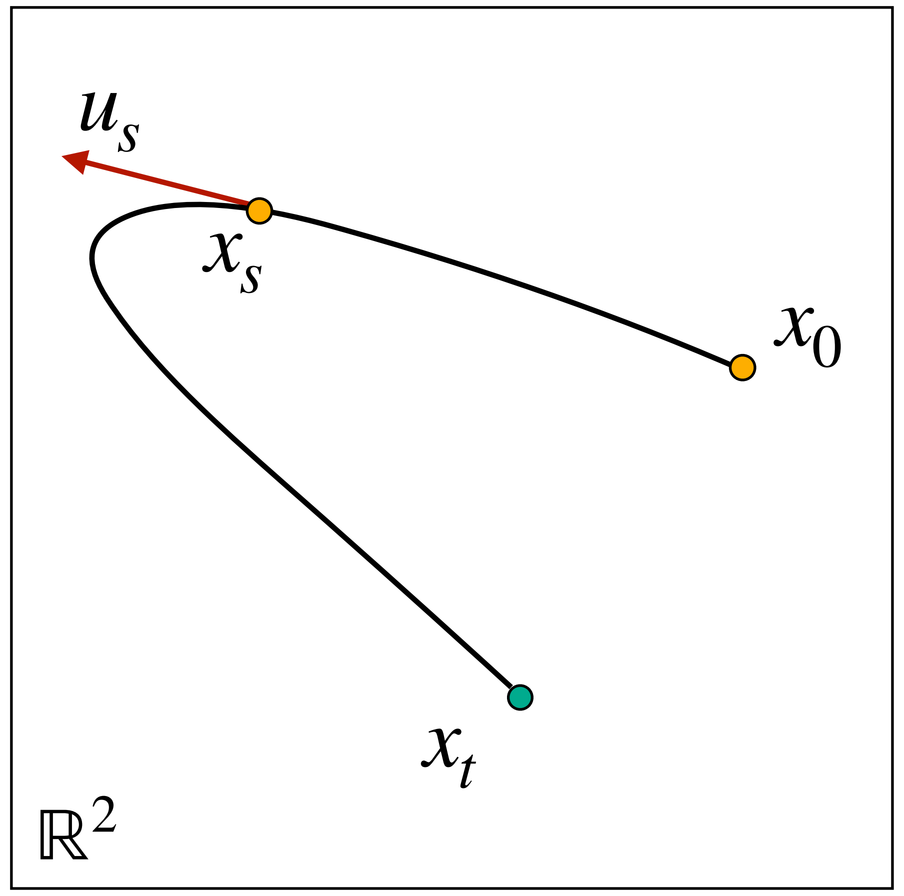
Deterministic and Stochastic Dynamics

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

Deterministic



$$x_t = x_0 + \int_0^t u_s(x_s)ds$$

Diffusion

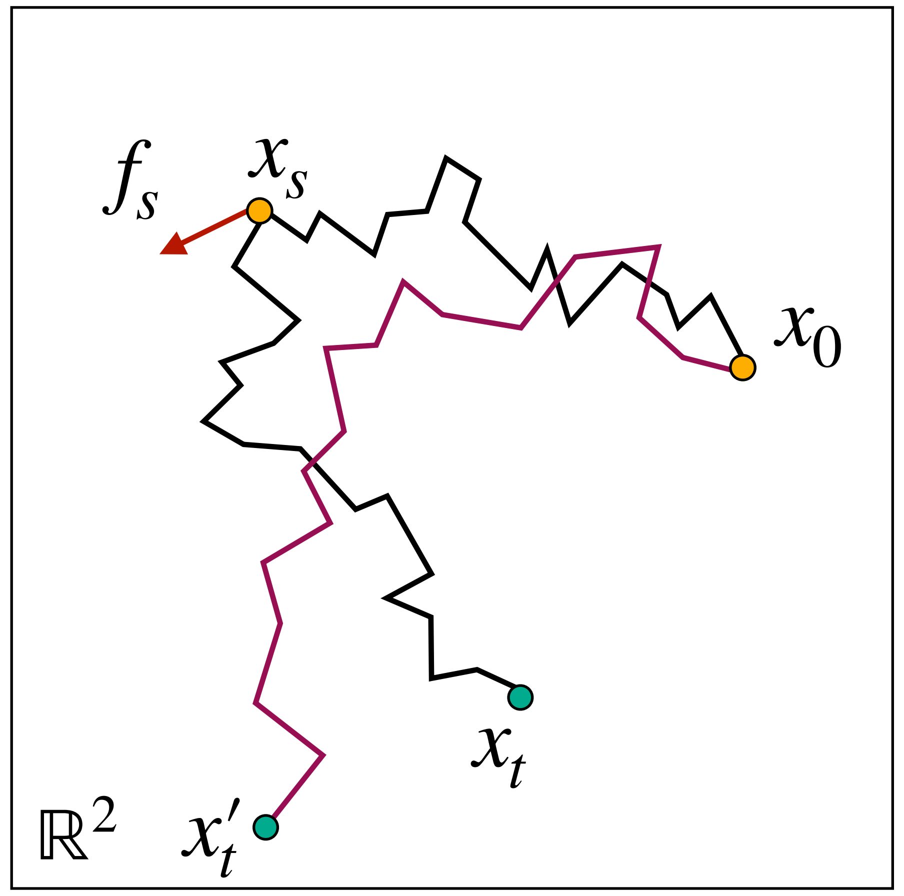
SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

Stochastic



$$x_t = x_0 + \int_0^t f_s(x_s)ds + \int_0^t g_s dw_s$$

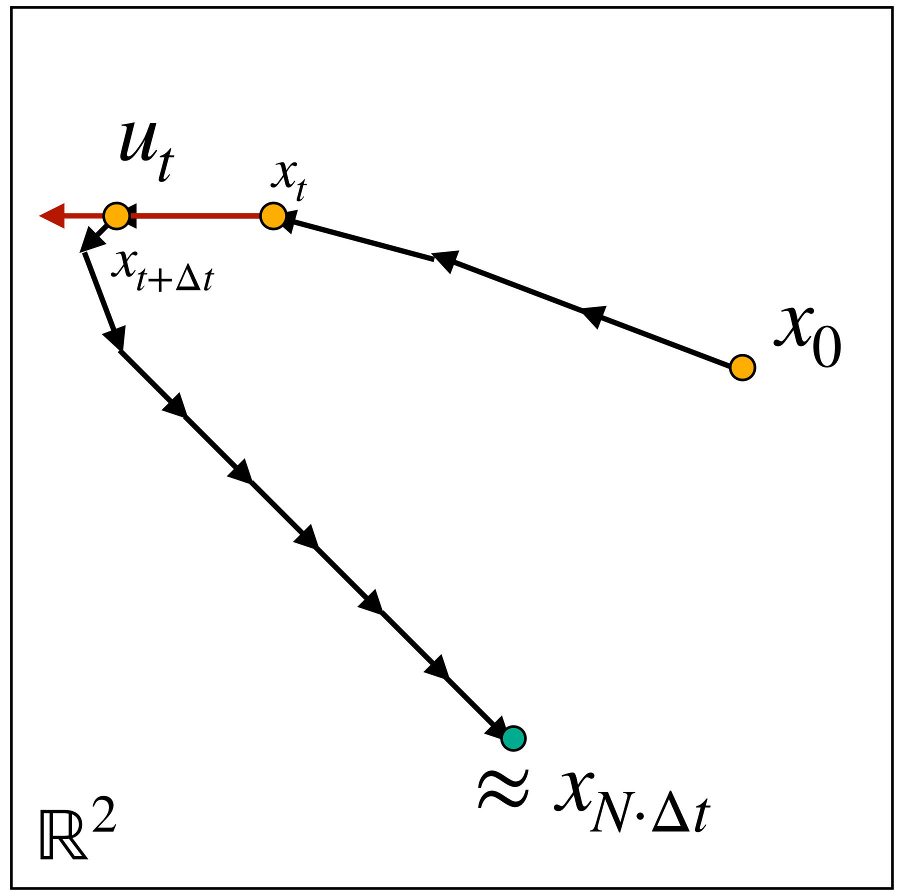
Deterministic and Stochastic Dynamics

Flows

ODE $dx_t = u_t(x_t)dt$

Velocity field

Deterministic



Euler:

$$x_{t+\Delta t} = x_t + \Delta t \cdot u_t(x_t)$$

Diffusion

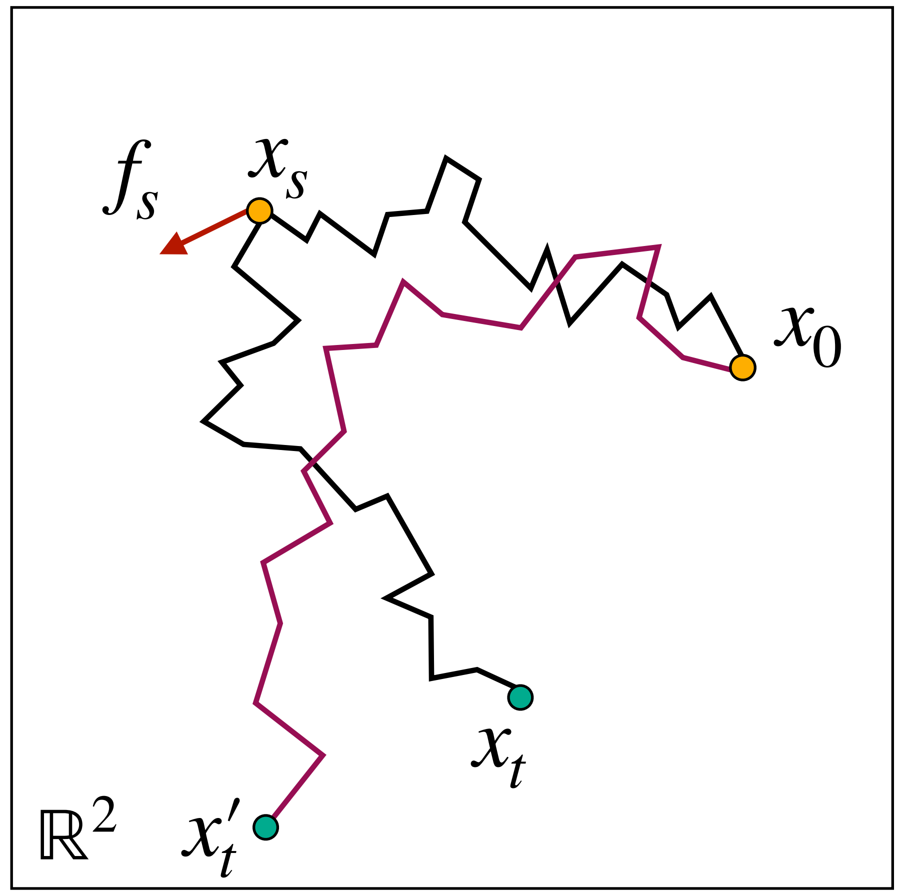
SDE $dx_t = f_t(x_t)dt + g_t dw_t$

Drift

Diffusion Coefficient

Brownian Motion

Stochastic



Euler-Maruyama:

$$x_{t+\Delta t} = x_t + \Delta t \cdot f_t(x_t) + g_t\sqrt{|\Delta t|}\, z_t \qquad z_t \sim N(0,1)$$

Where are the probabilities?

Flows

ODE $dx_t = u_t(x_t)dt$

Velocity field

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Diffusion

SDE $dx_t = f_t(x_t)dt + g_t dw_t$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Where are the probabilities?

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Yes!

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing…

Can we build a generative model with these?

Diffusion and Score Models

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing…

Sohl-Dickstein et al., *Deep unsupervised learning using nonequilibrium thermodynamics.* (ICML 2015)
Ho et at., *Denoising Diffusion Probabilistic Models.* (NeurIPS 2020)
Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations.* (ICLR 2021)

Diffusion and Score Models

Forward SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Data → Noise

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing…

Sohl-Dickstein et al., *Deep unsupervised learning using nonequilibrium thermodynamics.* (ICML 2015)
Ho et at., *Denoising Diffusion Probabilistic Models.* (NeurIPS 2020)
Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations.* (ICLR 2021)

Diffusion and Score Models

Forward SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$ Data → Noise

Reverse SDE $$d\bar{x}_t = (f_t(x_t) - g_t^2 \boxed{\nabla \log p_t})dt + g_t d\bar{w}_t$$ Noise → Data

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Need one more thing… The Score!

Sohl-Dickstein et al., *Deep unsupervised learning using nonequilibrium thermodynamics.* (ICML 2015)
Ho et at., *Denoising Diffusion Probabilistic Models.* (NeurIPS 2020)
Song et al., *Score-Based Generative Modeling through Stochastic Differential Equations.* (ICLR 2021)

Diffusion and Score Models

Forward SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$ Data → Noise

Reverse SDE $$d\bar{x}_t = (f_t(x_t) - g_t^2 \boxed{\nabla \log p_t})dt + g_t d\bar{w}_t$$ Noise → Data

Learn the score by regressing to conditional scores:

$$\min_\theta \mathbb{E}_{p_{data}, p_t(x|x_{data})} \left[\| s_t^\theta(x) - \nabla \log p_t(x \mid x_{data}) \|^2 \right]$$

Simulation-free

Known SDEs: Variance Exploding
Variance Preserving

Where are the probabilities?

Flows

ODE $$dx_t = u_t(x_t)dt$$

Velocity field

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Yes!

Diffusion

SDE $$dx_t = f_t(x_t)dt + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Fokker-Planck Equation

$$\partial_t p_t = -\operatorname{div}(p_t f_t) + \frac{1}{2} g_t^2 \nabla^2 p_t$$

Learn: score $\nabla \log p_t$

- Only Gaussian source
- Solution asymptotically reaches source

Where are the probabilities?

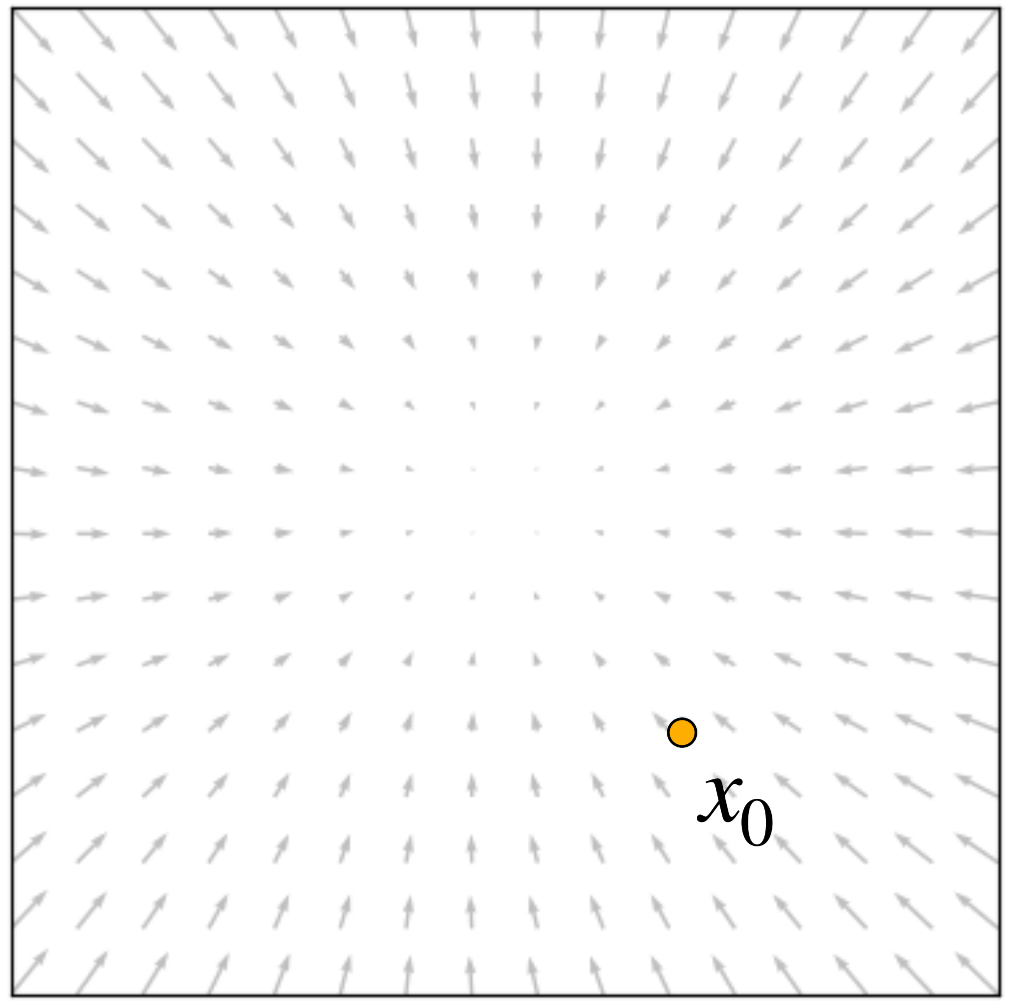
Flows

ODE $dx_t = u_t(x_t)dt$

Velocity field

Flow ODE

$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$



$\psi_t(x)$ is smooth with smooth **inverse** defined by $-u_t(x)$

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Yes!

Where are the probabilities?

Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Learn: velocity field u_t

- Universal transformation between densities
- Defined on finite time interval

Diffusion

SDE $$dx_t = \underbrace{f_t}_{\text{Drift}}(x_t)dt + \underbrace{g_t}_{\text{Diffusion Coefficient}}\underbrace{dw_t}_{\text{Brownian Motion}}$$

The Liouville Equation

$$\partial_t p_t = -\operatorname{div}\left(p_t\left(f_t - \frac{1}{2}g_t^2 \nabla \log p_t\right)\right)$$

Learn: score $\nabla \log p_t$

- Only Gaussian source
- Solution asymptotically reaches source

Where are the probabilities?

Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

The Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Learn: velocity field u_t

- Universal transformation between densities
- Defined on finite time interval

Diffusion

SDE $$\underbrace{dx_t = f_t(x_t)dt}_{} + g_t dw_t$$

Drift

Diffusion Coefficient

Brownian Motion

The Liouville Equation

$$\partial_t p_t = -\operatorname{div}\left(p_t\left(f_t - \frac{1}{2}g_t^2 \nabla \log p_t\right)\right)$$

Learn: score $\nabla \log p_t$

- Only Gaussian source
- Solution asymptotically reaches source

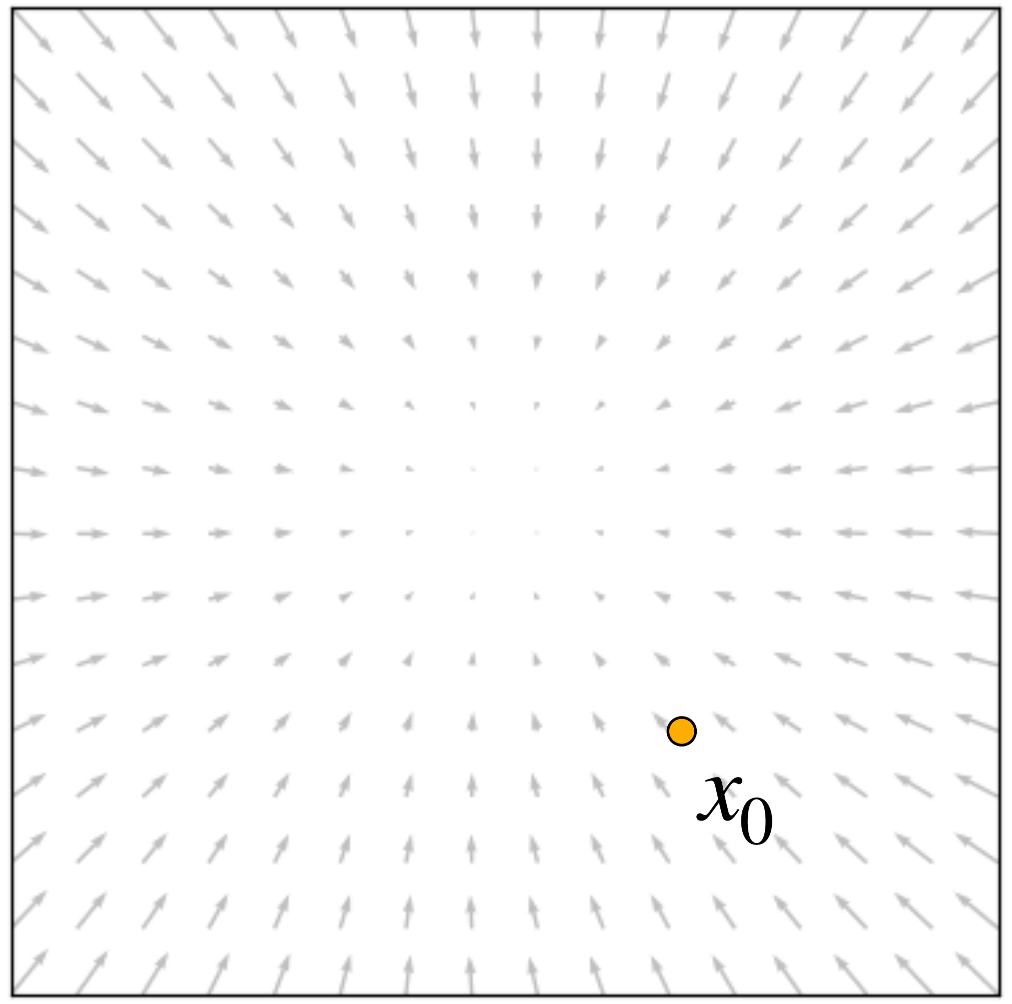
Flows as Generative Models

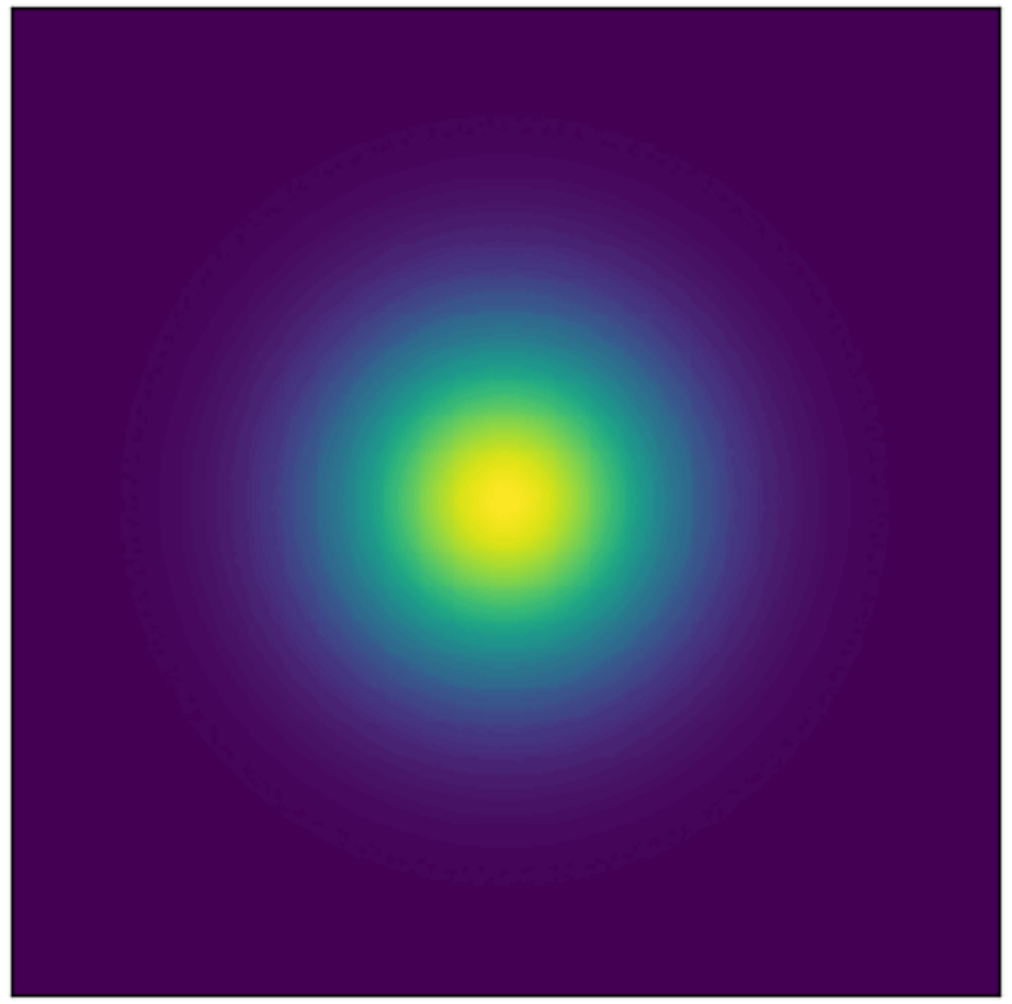
Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

Continuity Equation

$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$





Goal: find velocity field u_t s.t. $p_1 \approx q$

Learn: velocity field u_t

Chen et al., *Neural Ordinary Differential Equations.* (NeurIPS 2018)

Training with Simulation

Flow ODE

$$\dot{\psi}_t(x_0) = u_t(\psi_t(x_0))$$

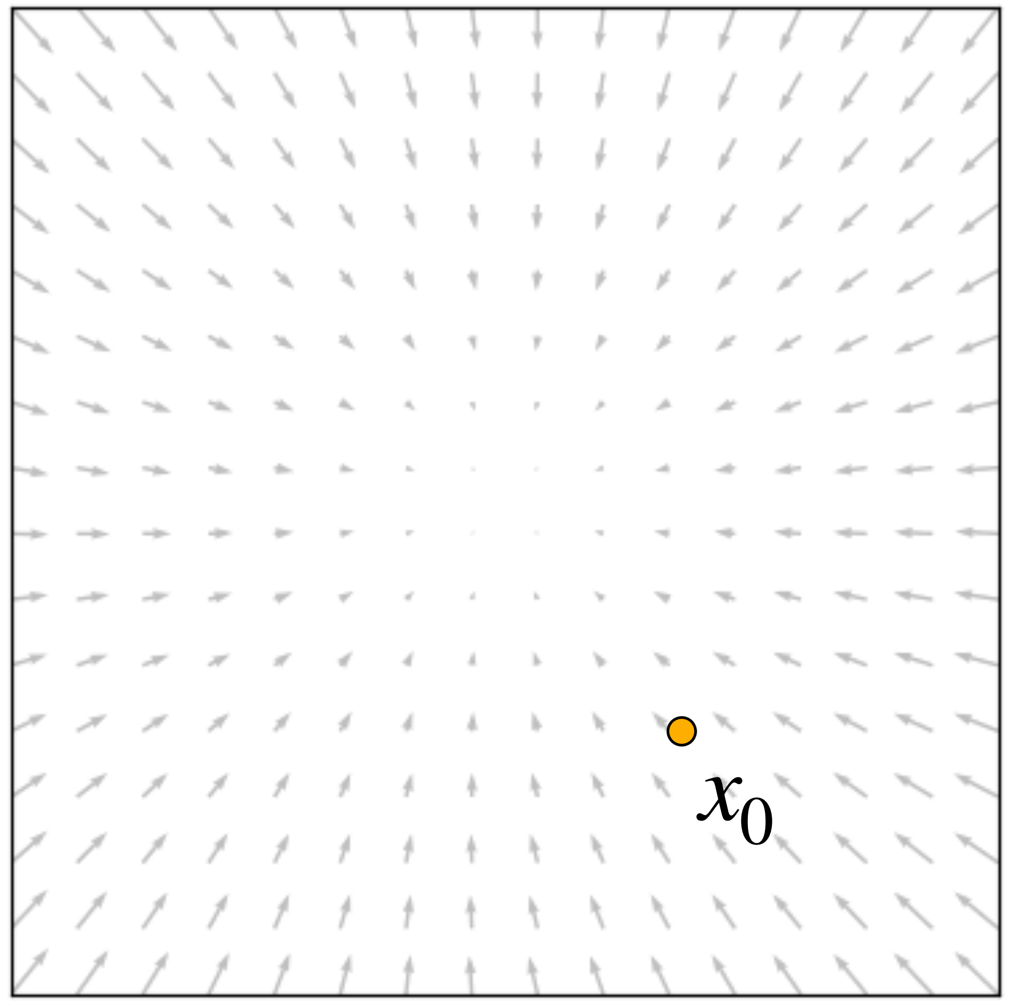
Continuity Equation

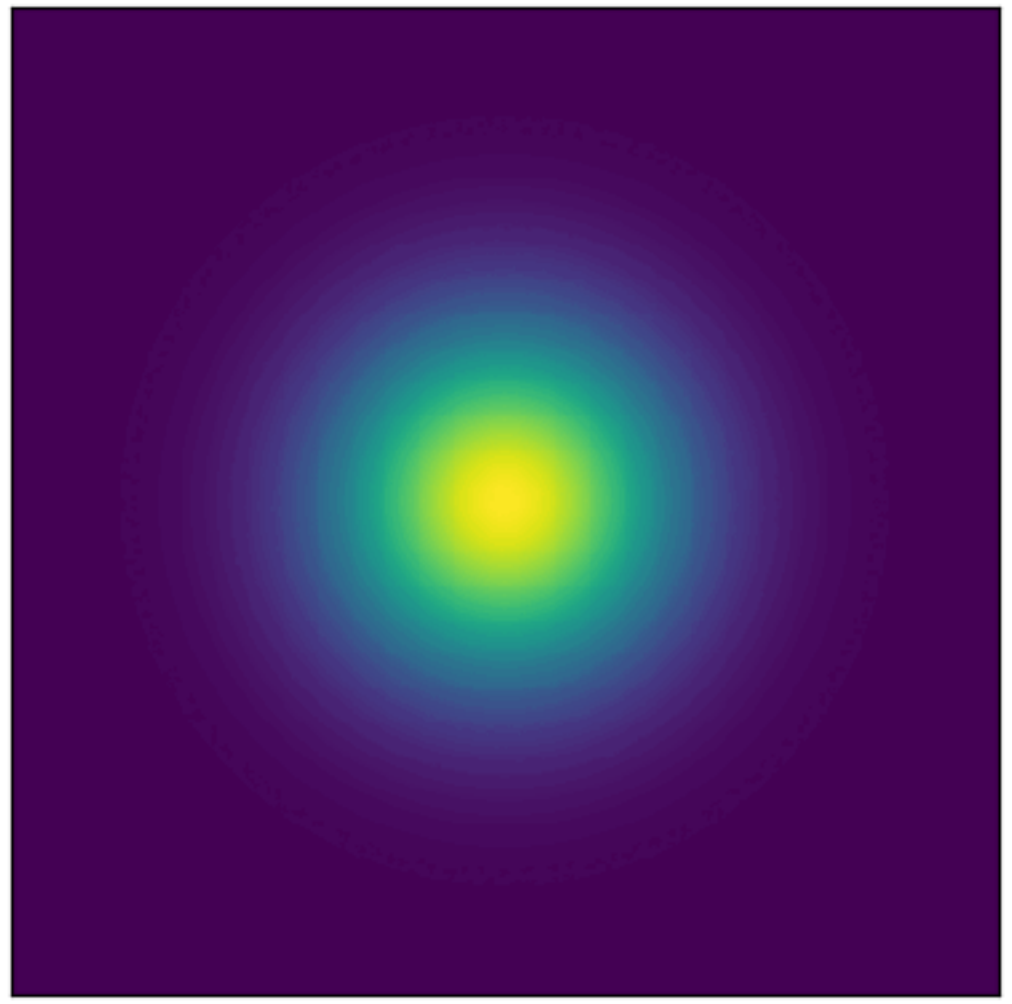
$$\partial_t p_t = -\operatorname{div}(p_t u_t)$$

Log-likelihood computation

$$\log p_1(x_1) = \log p(x_0) + \int_1^0 \operatorname{div}(u_t(x_t))dt$$

$$x_t = x_1 + \int_1^t u_s(x_s)ds$$





Requires:

- Simulating x_t
- Backprop through simulation
- (Unbiased) estimator of $\operatorname{div}(u_t)$
- Can compute $\log p(x)$

Maximum Likelihood Objective

$$D_{\mathrm{KL}}(q \,\|\, p_1) = -\mathbb{E}_{x\sim q} \log p_1(x) + c$$

Chen et al., *Neural Ordinary Differential Equations.* (NeurIPS 2018)

Flow Matching

$$L_{\mathrm{FM}}(\theta) = \min \mathbb{E}_{t, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

Core Principle:

u_t generates p_t

iff they satisfy the continuity equation

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

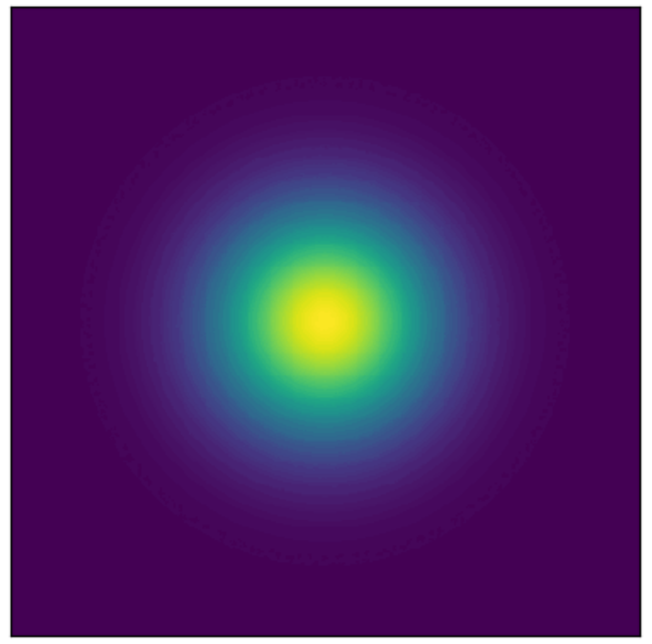
Find a tractable optimization objective

Conditional Probability Paths

Marginal path

Law of total probability

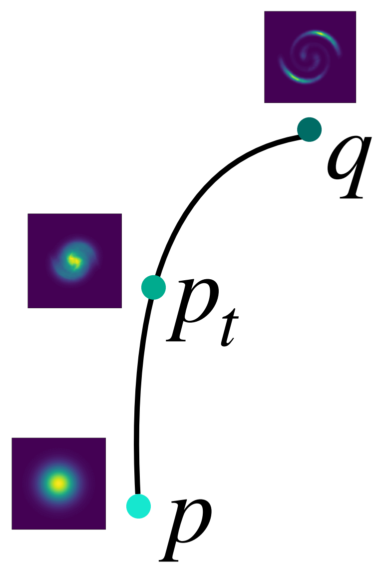
$$p_t(x) = \int p_t(x \mid x_1) q(x_1) dx_1$$



Boundary conditions:

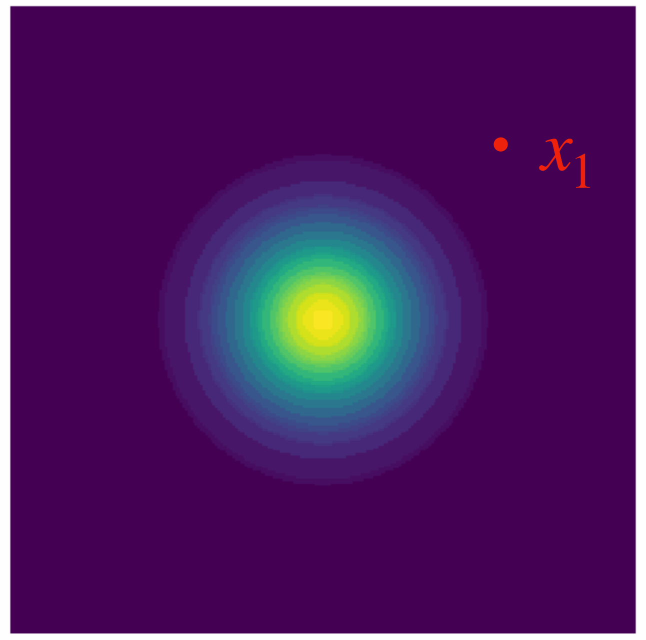
$$p_0 = p$$

$$p_1 = q$$



Conditional path

$$p_t(x \mid x_1)$$



$$p_0(\cdot \mid x_1) = p$$

$$p_1(\cdot \mid x_1) = \delta_{x_1}$$

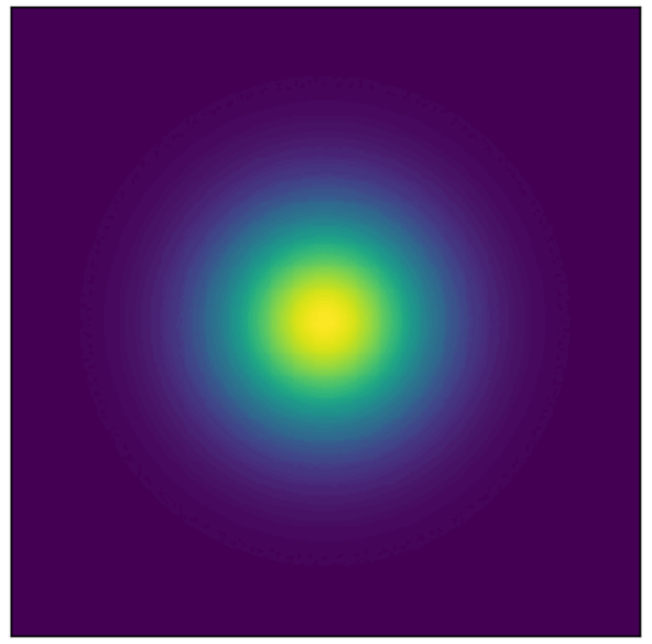
Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

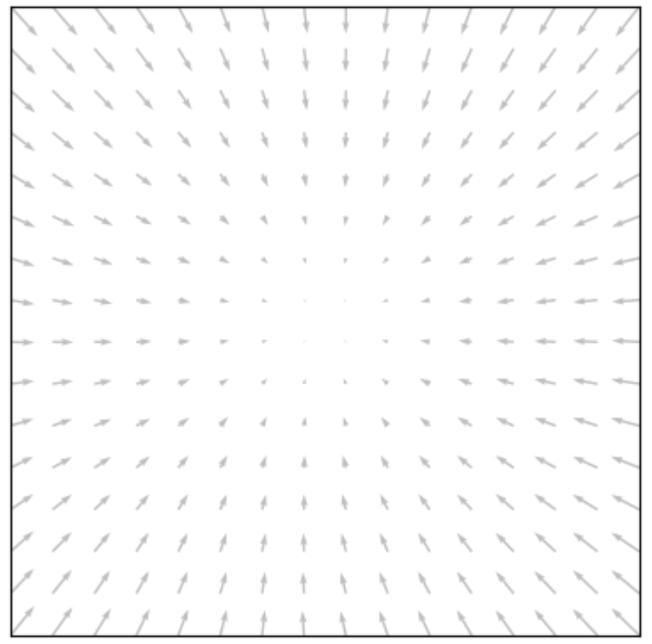
The marginalization “trick”

$$p_t(x) = \int p_t(x \mid x_1) q(x_1) dx_1$$

$$u_t(x) = \int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q(x_1)}{p_t(x)} dx_1$$

Marginal path

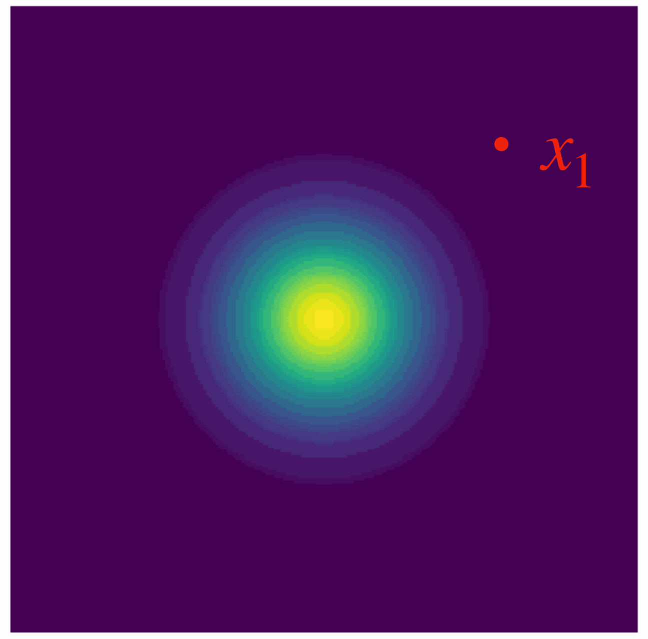


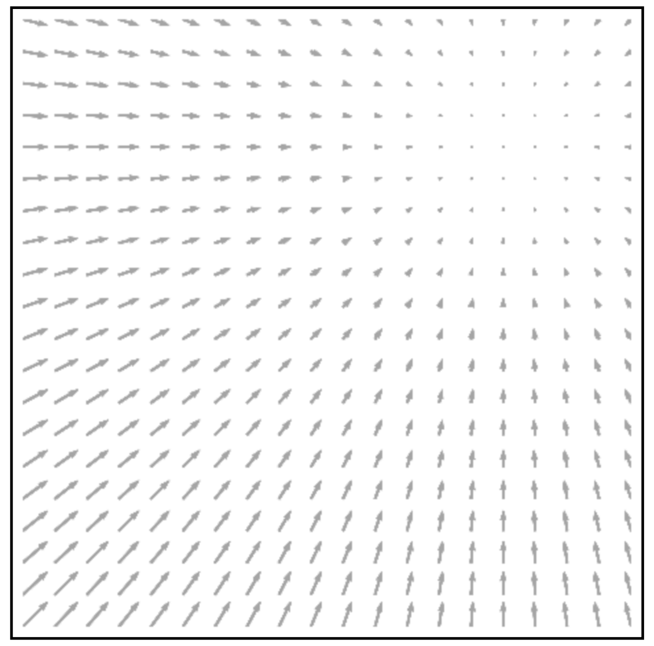


$$p_t(x \mid x_1)$$

$$u_t(x \mid x_1)$$

Conditional path





Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Flow Matching

$$L_{\mathrm{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^{\theta}(x) - u_t(x) \|^2$$

$$u_t(x) = \int u_t(x \mid x_1) \frac{p_t(x \mid x_1) q(x_1)}{p_t(x)} dx_1$$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

✔

Find a tractable optimization objective

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Flow Matching

$$L_{\mathrm{FM}}(\theta) = \min \mathbb{E}_{t,\, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$u_t(x) = \int u_t(x \mid z) \frac{p_t(x \mid z) q(z)}{p_t(x)} dz$$

Useful examples:

$z = (x_0, x_1) \to q(x_0, x_1)$

$z = x_0 \to p(x_0)$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

✔

Find a tractable optimization objective

Pooladian*, Ben-Hamu*, Enrich* et al., *Multisample Flow Matching: Straightening Flows with Minibatch Couplings.* (ICML 2023)
Tong et al., *Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport.* (TMLR)

$$p_t(z|x) = \frac{p_t(x|z)q(z)}{p_t(x)}$$

Flow Matching

$$L_{\text{FM}}(\theta) = \min \mathbb{E}_{t, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$u_t(x) = \int u_t(x|z)p_t(z|x)dz$$

Useful examples:

$z = (x_0, x_1) \rightarrow q(x_0, x_1)$

$z = x_0 \rightarrow p(x_0)$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t ✔

Find a tractable optimization objective

Pooladian*, Ben-Hamu*, Enrich* et al., *Multisample Flow Matching: Straightening Flows with Minibatch Couplings.* (ICML 2023)
Tong et al., *Improving and Generalizing Flow-Based Generative Models with Minibatch Optimal Transport.* (TMLR)

Conditional Flow Matching Loss

$$L_{\mathrm{FM}}(\theta) = \min \mathbb{E}_{t, p_t(x)} \| u_t^{\theta}(x) - u_t(x) \|^2$$

$\Updownarrow$

$$L_{\mathrm{CFM}}(\theta) = \min \mathbb{E}_{t, q(z), p_t(x|z)} \| u_t^{\theta}(x) - u_t(x \mid z) \|^2$$

The gradients of losses coincide:

$$\nabla_\theta L_{\mathrm{FM}} = \nabla_\theta L_{\mathrm{CFM}}$$

Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Flow Matching

$$L_{\mathrm{FM}}(\theta) = \min \mathbb{E}_{t, p_t(x)} \| u_t^\theta(x) - u_t(x) \|^2$$

$$L_{\mathrm{CFM}}(\theta) = \min \mathbb{E}_{t, q(z), p_t(x|z)} \| u_t^\theta(x) - u_t(x \mid z) \|^2$$

Construct:

- Target probability path p_t s.t. $p_0 = p$, $p_1 \approx q$
- Generating velocity field u_t

✔

Find a tractable optimization objective

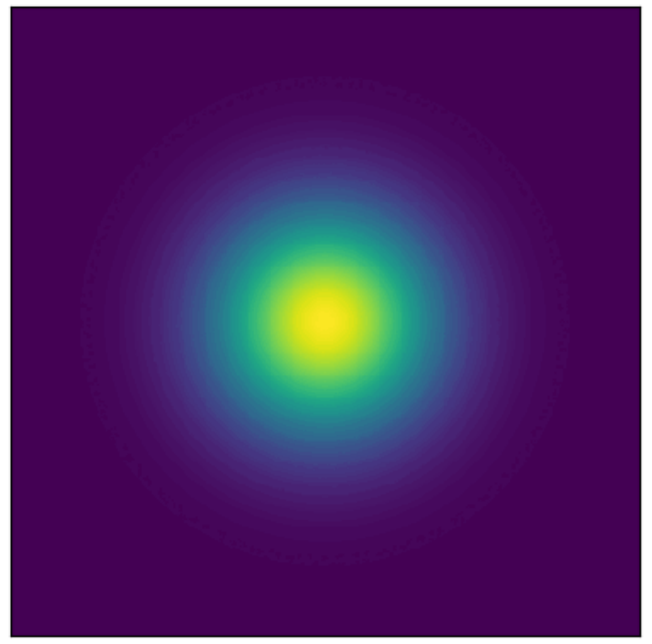
✔

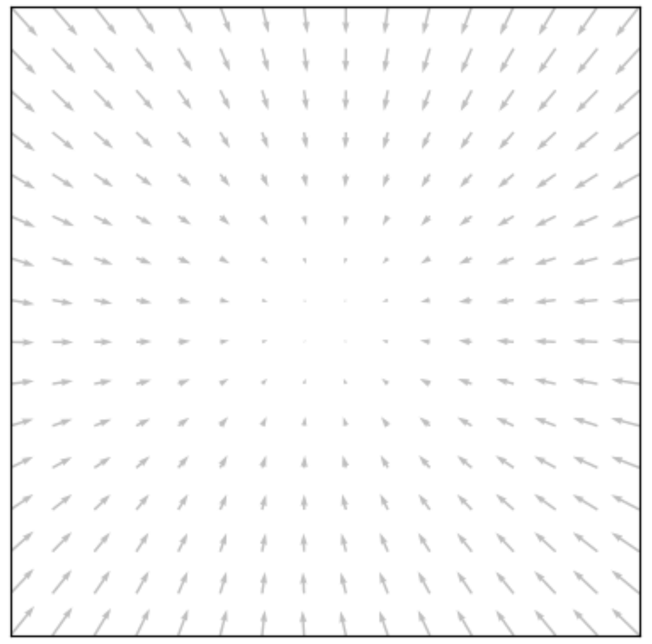
Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Conditional Flows

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1 \qquad u_t(x) = \int u_t(x|x_1)p_t(x_1|x)dx_1$$

Marginal path



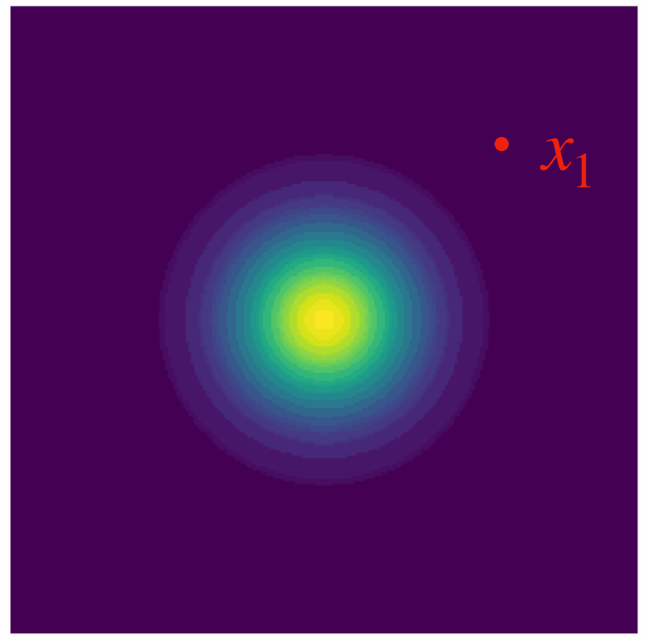


Construct a conditional flow s.t.

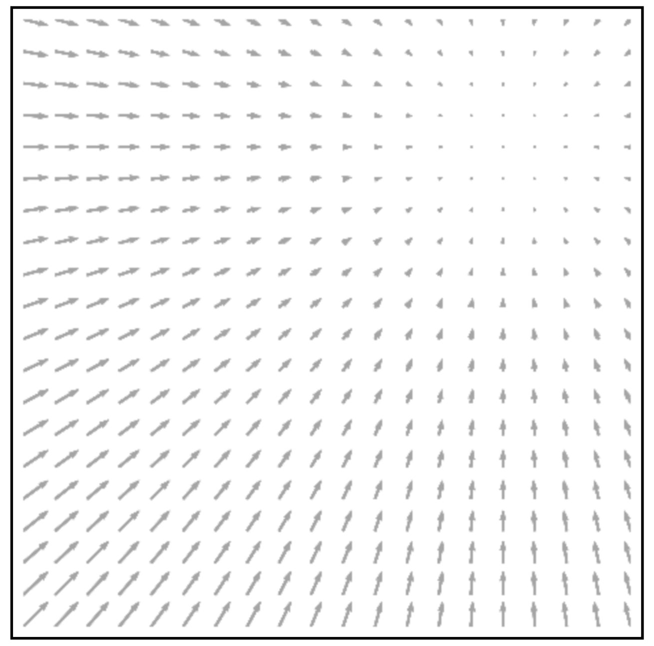
$$\psi_0(x|x_1) = x \ , \ \psi_1(x|x_1) = x_1$$

Conditional path

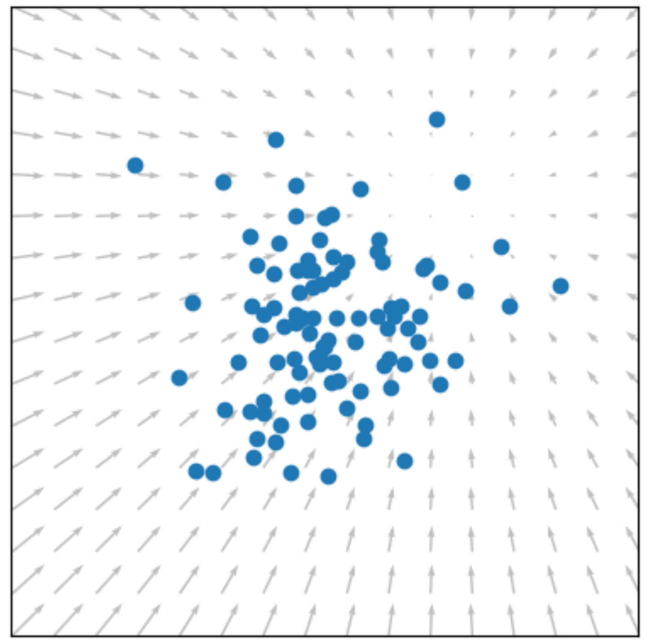
$p_t(x|x_1)$



$u_t(x|x_1)$



$\psi_t(x|x_1)$



Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

Conditional Optimal Transport Flows

Construct a conditional flow s.t.
$\psi_0(x|x_1) = x \quad , \quad \psi_1(x|x_1) = x_1$

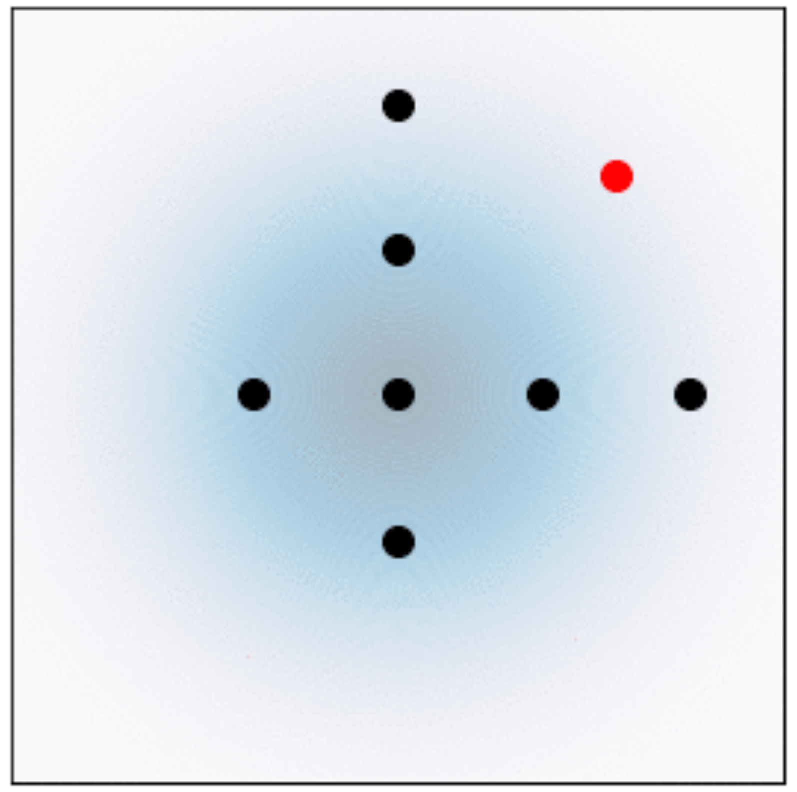
Cond-OT flow coefficients: $\alpha_t = t \quad , \quad \sigma_t = 1 - t$

Cond-OT flow: $\psi_t(x_0|x_1) = tx_1 + (1-t)x_0$

$$u_t(\psi_t(x_0|x_1)|x_1) = x_1 - x_0$$

$$u_t(x|x_1) = \frac{x_1 - x}{1 - t}$$

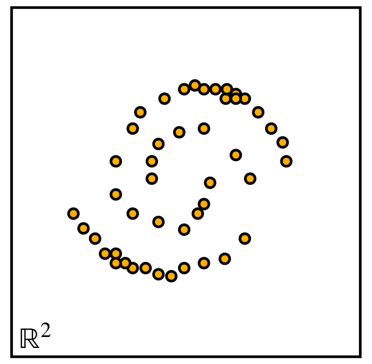
$\psi_t(x_0|x_1)$



Lipman et al., *Flow Matching for Generative Modeling.* (ICLR 2023)

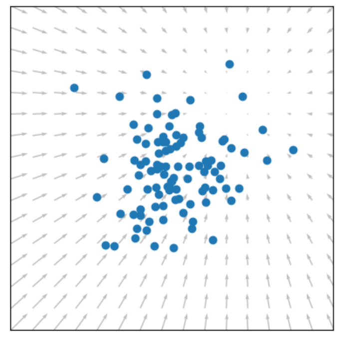
Recipe: Flow Matching

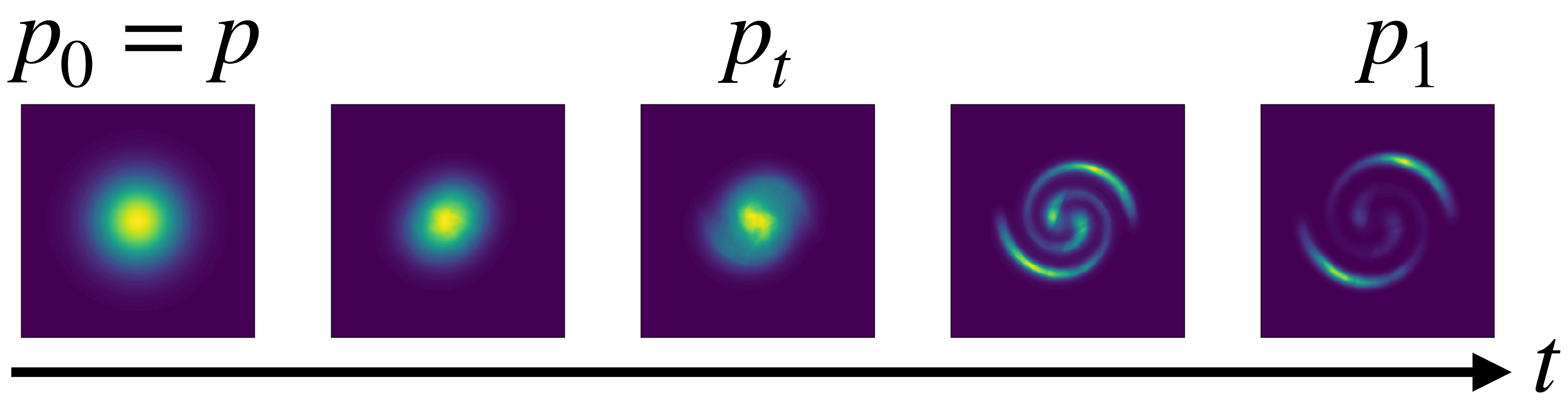
- Given: samples $x_1 \sim q$



- Construct: p_t s.t. $p_0 = p \ , p_1 \approx q$

  via conditional flows $\psi_t(x \mid z)$

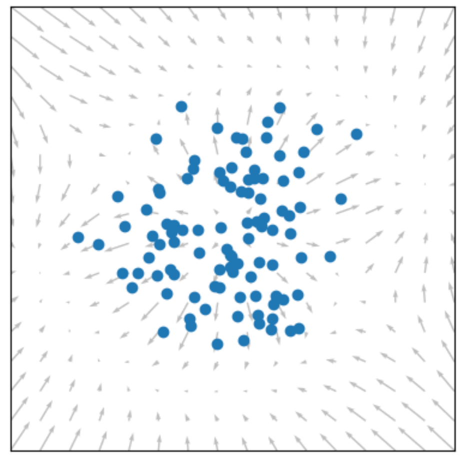




- Learn: velocity field u_t with CFM loss

  s.t. $\psi_t(x_0) \sim p_t$ where $x_0 \sim p$

$$L_{\text{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x) - u_t(x \mid z) \|^2$$



$\psi_t : [0,1] \times \mathbb{R}^2 \to \mathbb{R}^2$

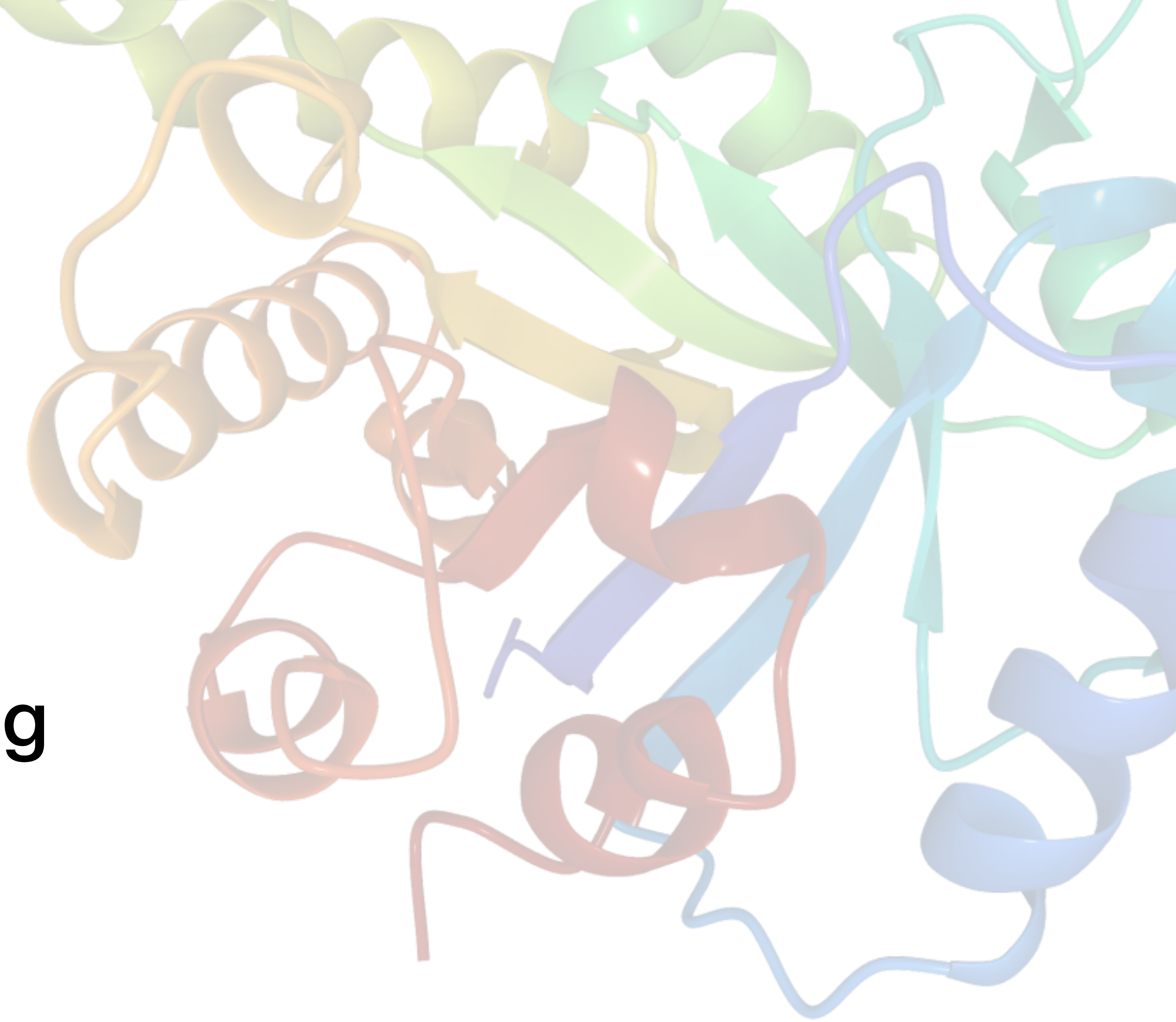
Summary

- Flows are powerful generative models when supervised adequately
  - Flow Matching is a **flexible** framework for training generative flows
- Improved sampling speed and stability compared to diffusion models

- <u>Open challenges:</u>
  - Learn a one-step model (without distillation).
  - Scale to other data domains - such as language.

Image from Esser et al. 2024

Part II:
Geometry for Machine Learning



So Generative Models on Manifolds?

- Given: samples $x_1 \sim q$ → How do you represent x_1 on a manifold?

- Construct: p_t s.t. $p_0 = p$, $p_1 \approx q$ → There is no "Gaussian dist." on manifold

  via conditional flows $\psi_t(x \mid z)$

$$\alpha_t x_1 + \sigma x_0$$

Can't do addition! No Vector space structure!

- Learn: velocity field u_t with CFM loss

  s.t. $\psi_t(x_0) \sim p_t$ where $x_0 \sim p$ → The notion of velocity/vector fields needs to be generalized

Smooth Manifolds

- Informally: A (smooth) topological space that locally looks like patches of a Vector Space (think Euclidean space) when glued together look globally different.

Points live on this topological space
e.g. $x_1 \in \mathscr{M}$

Does curvature impact numerical stability?

How do we parametrize $\mathscr{M}$

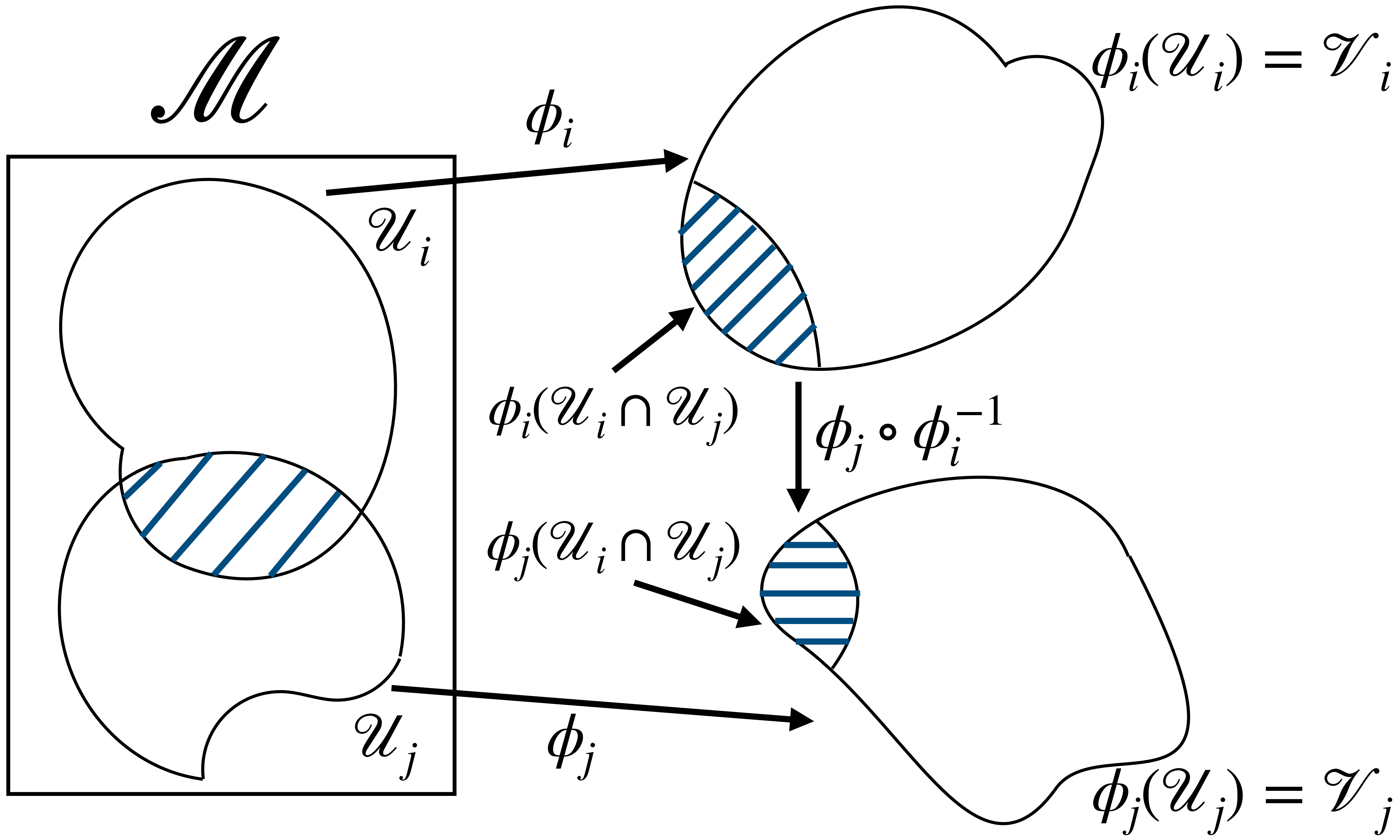
What additional structure do we need on $\mathscr{M}$ for FM/Diffusion?

Smooth Manifolds

- Informally: A (smooth) topological space that locally looks like patches of a Vector Space (think Euclidean space) when glued together look globally different.

- A chart $\{U_i, \phi_i \,|\, i \in \mathscr{A}\}$ maps each patch to a vector space $\phi_i : U_i \to \mathbb{R}^n$.

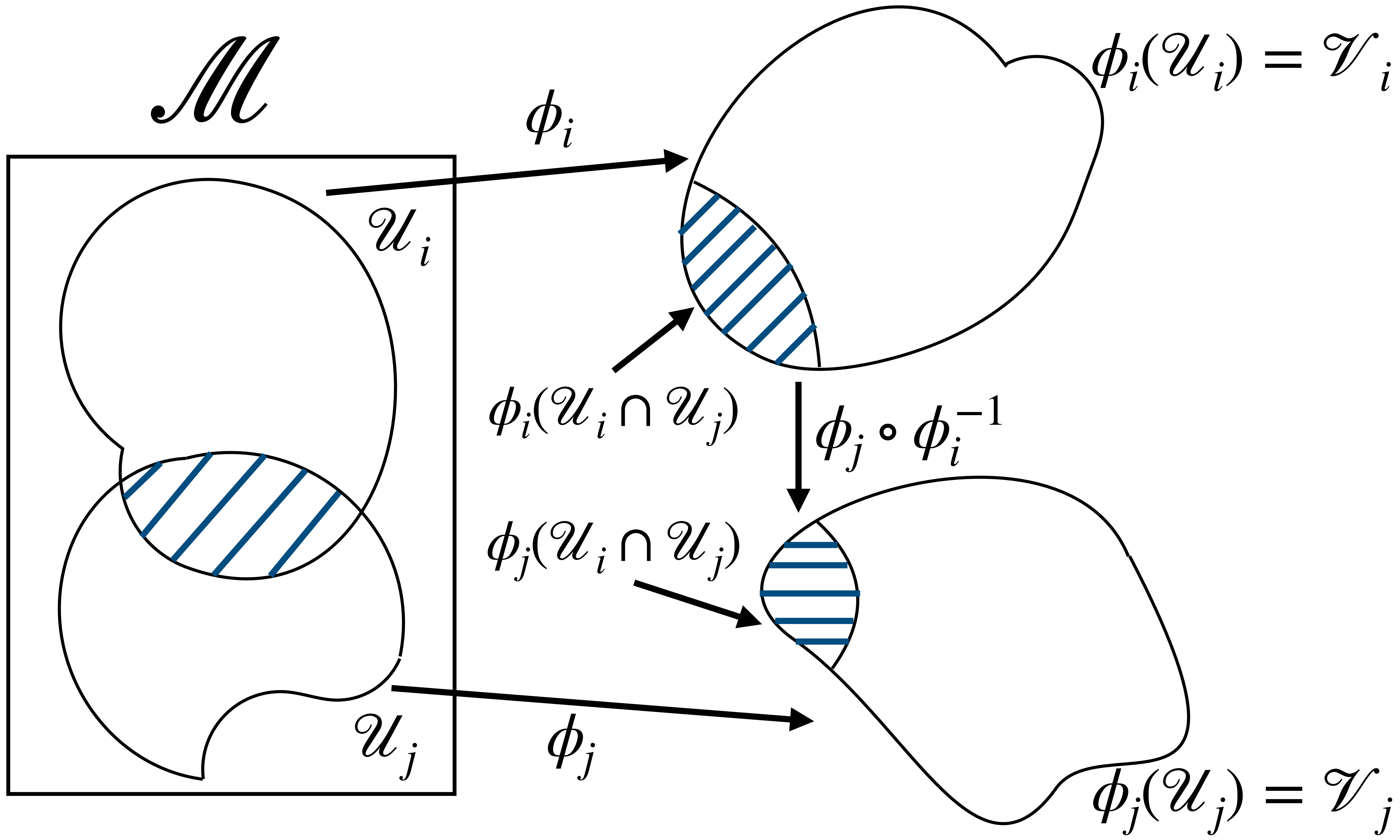
We added “smoothness” i.e. C^∞ differentiability and continuity to $\mathscr{M}$



Smooth Manifolds

- Informally: A (smooth) topological space that locally looks like patches of a Vector Space (think Euclidean space) when glued together look globally different.
- A chart $\{U_i, \phi_i \,|\, i \in \mathscr{A}\}$ maps each patch to a vector space $\phi_i : U_i \to \mathbb{R}^n$.
- Stitching charts together requires satisfying a compatibility condition if $U_i \cap U_j \neq \emptyset$

$$\phi_j \circ \phi_i^{-1}\Big|_{\phi_i(U_i \cap U_j)} : \phi_i(U_i \cap U_j) \to \phi_j(U_i \cap U_j)$$



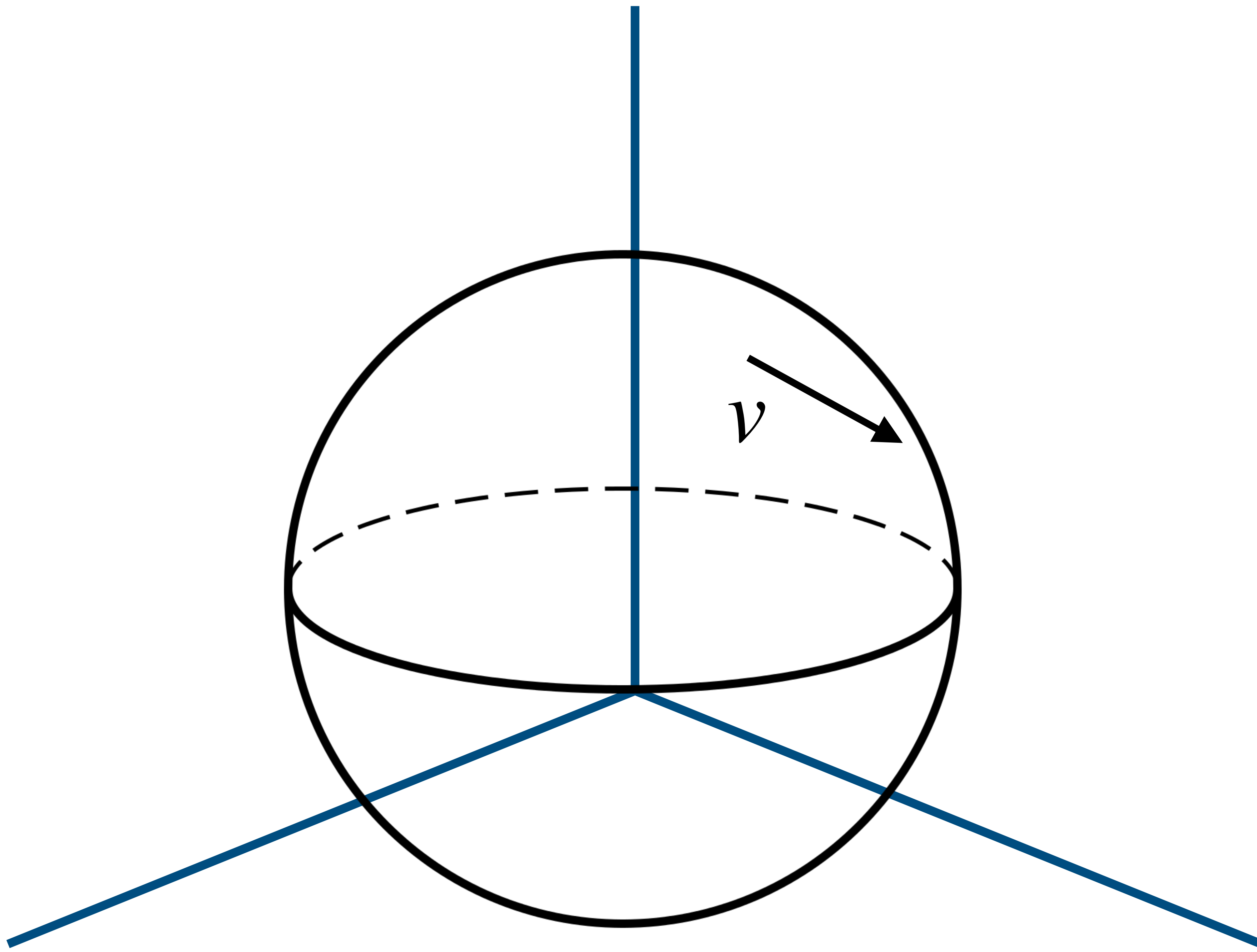
Extrinsic vs. Intrinsic Views

- Multiple ways of representing the *same geometry*. Two main ways are Extrinsic vs. Intrinsic perspectives of Riemannian geometry.

- Extrinsic: A manifold is embedded in $\mathbb{R}^n$, $n > d$, if there is an inclusion map $\imath(x) = x \in \mathbb{R}^n, \forall x \in \mathcal{M}$.

Which parametrization should you use?

General principle: Think like a deep learner



$\mathbb{S}^2$ sphere embedded in $\mathbb{R}^3$

Extrinsic vs. Intrinsic Views

- Intrinsic: A local coordinate system is “a choice” of charts that cover the manifold.
- Computation in “local coordinates” means using coordinate charts to put it in in subsets of $\mathbb{R}^d$ instead of a manifold.

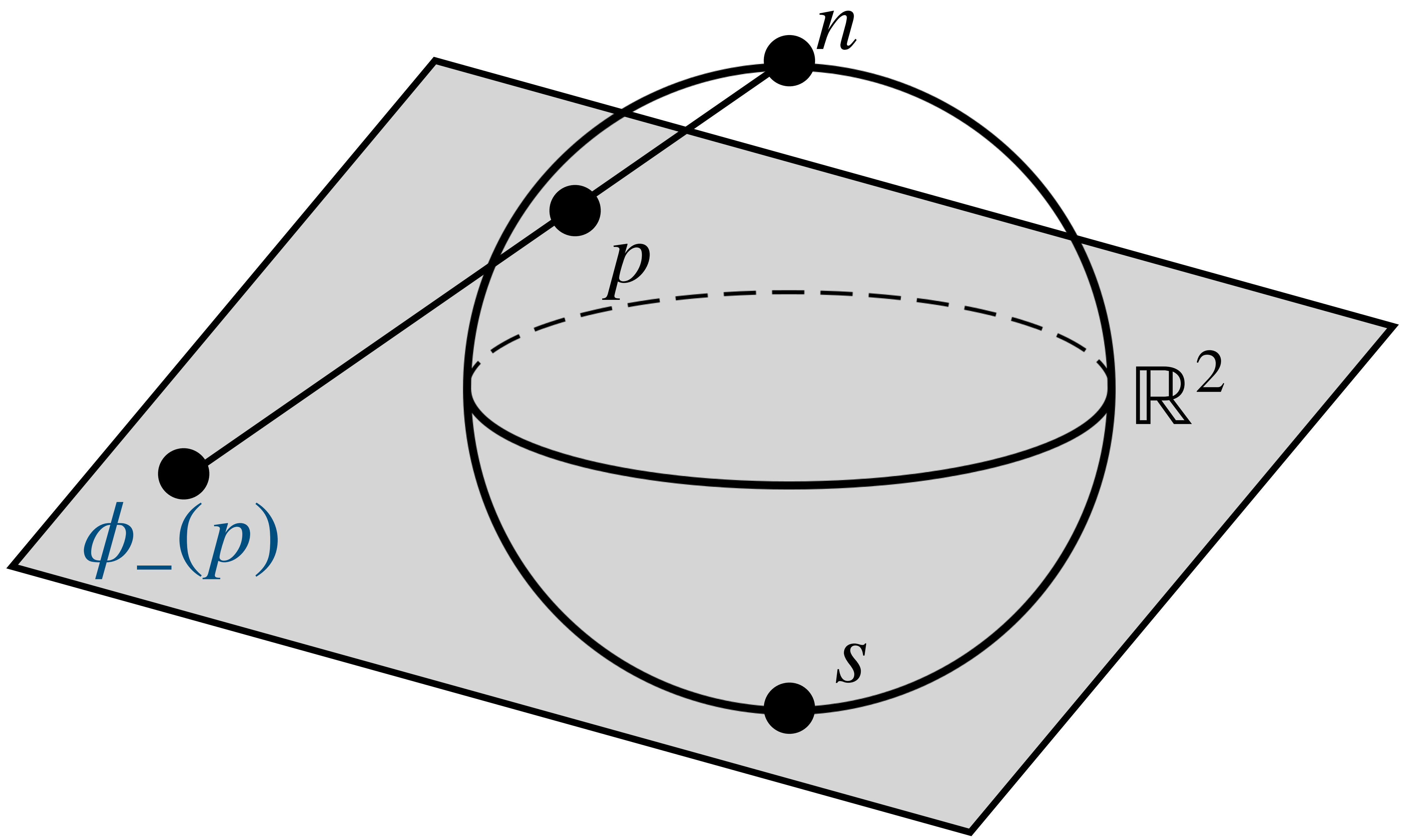
Example: Stereographic projection of $\mathbb{S}^2$

$$U_+ = \mathbb{S}^2 \backslash \{s\} \quad \phi_+ : U_+ \to \mathbb{R}^2$$

$$U_- = \mathbb{S}^2 \backslash \{n\} \quad \phi_- : U_- \to \mathbb{R}^2$$

Numerical instability near the poles!

Stereographic projection



Global Coordinate Systems

- Global coordinates: A coordinate chart that covers the entire manifold

(r, θ, φ)

(Almost) Global coordinate system

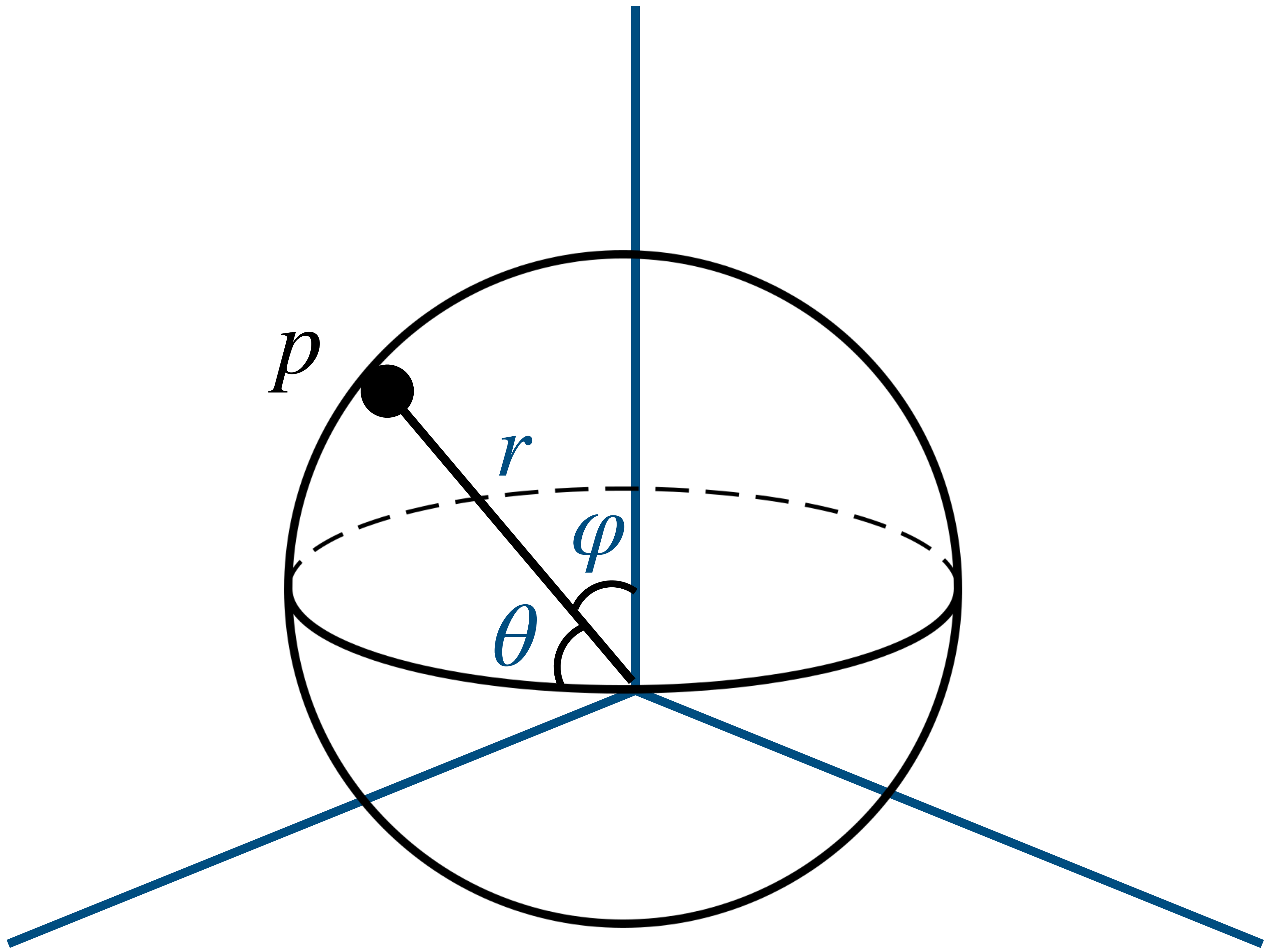
r - radius

θ - azimutal angle

φ - polar angle

Are trigonometric functions numerically stable (always?). What about their inverses?

Spherical Coordinate System



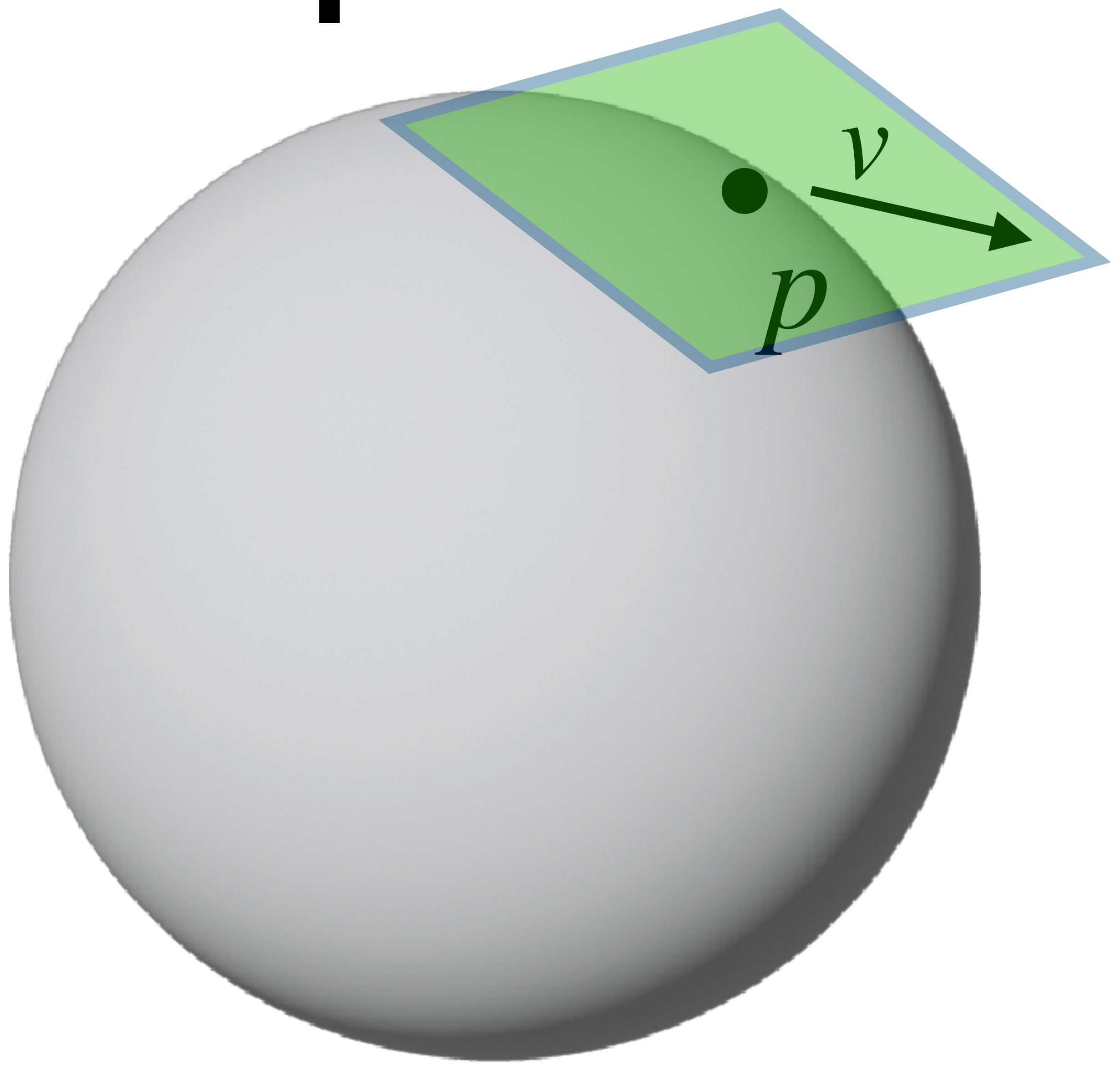
Defining Vectors

- Tangent space: For each $p \in \mathscr{M}$, a tangent vector is a smooth map $v : \mathscr{F} \to \mathbb{R}^n$.

How do we model $\boldsymbol{u}_t$?

Need to define a “vector” on $\mathscr{M}$

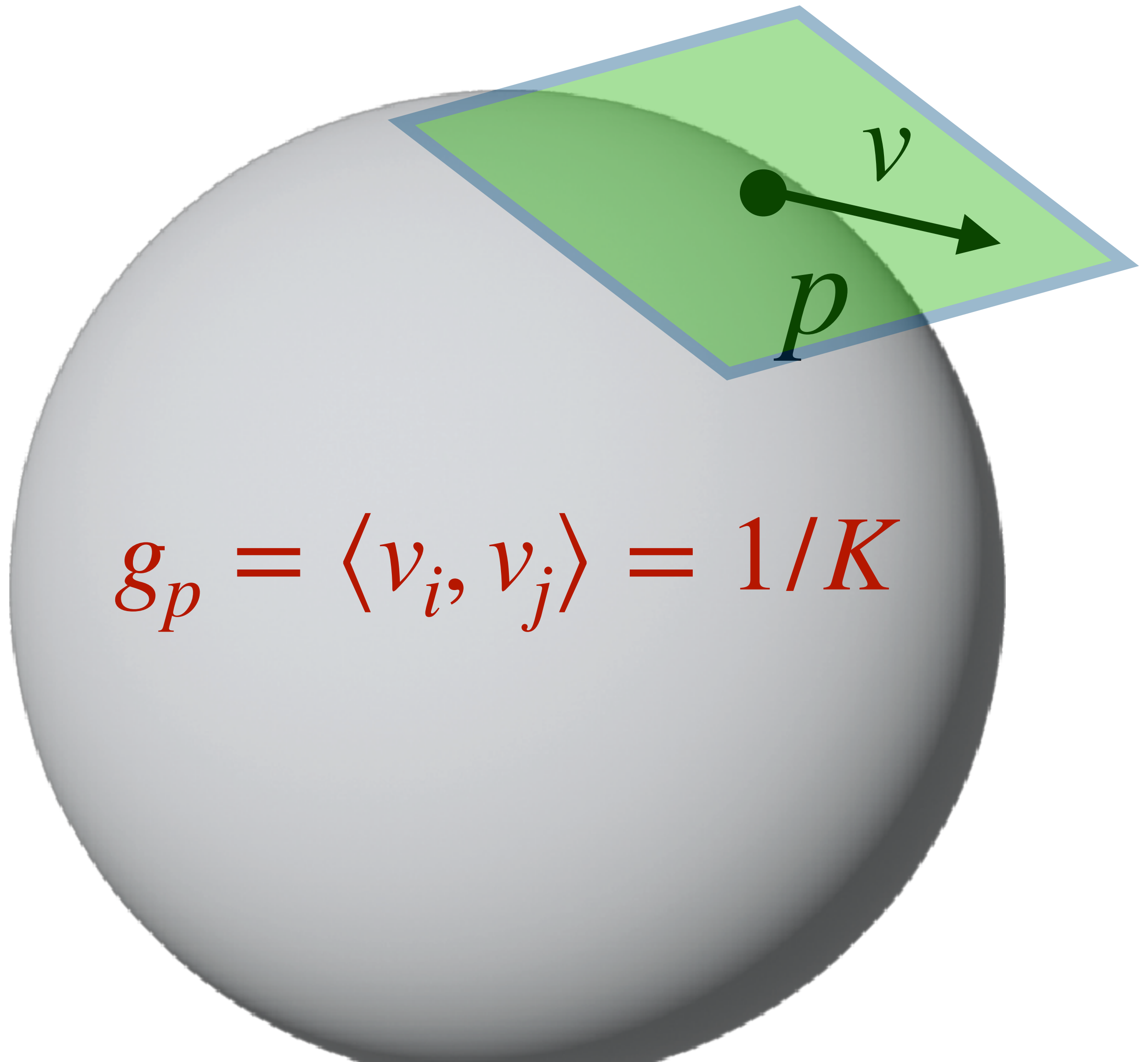
Sphere $\mathbb{S}^2$

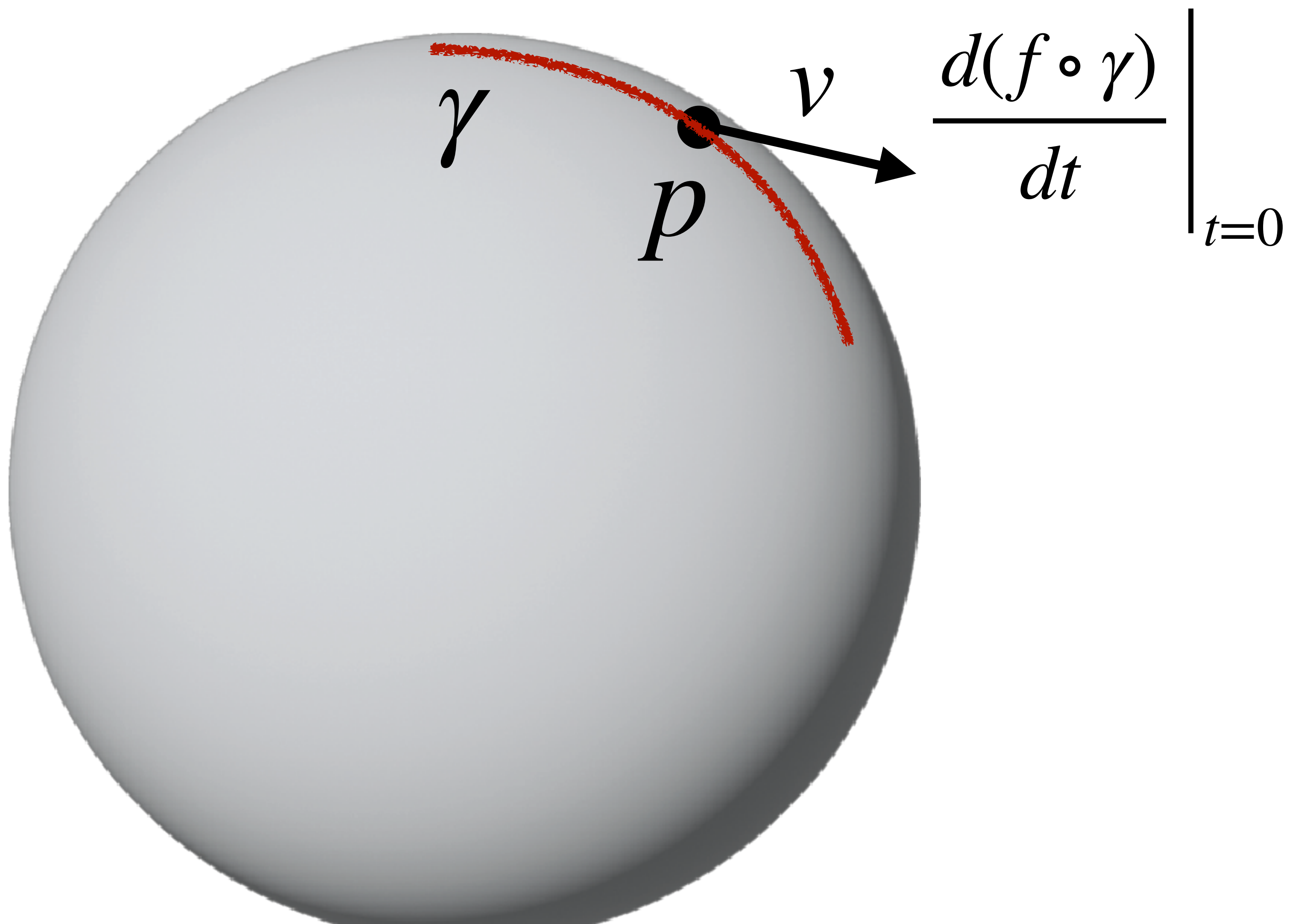


In a chart we can use the local basis $(e_1, \ldots, e_d)$

A bit more on Tangent Spaces

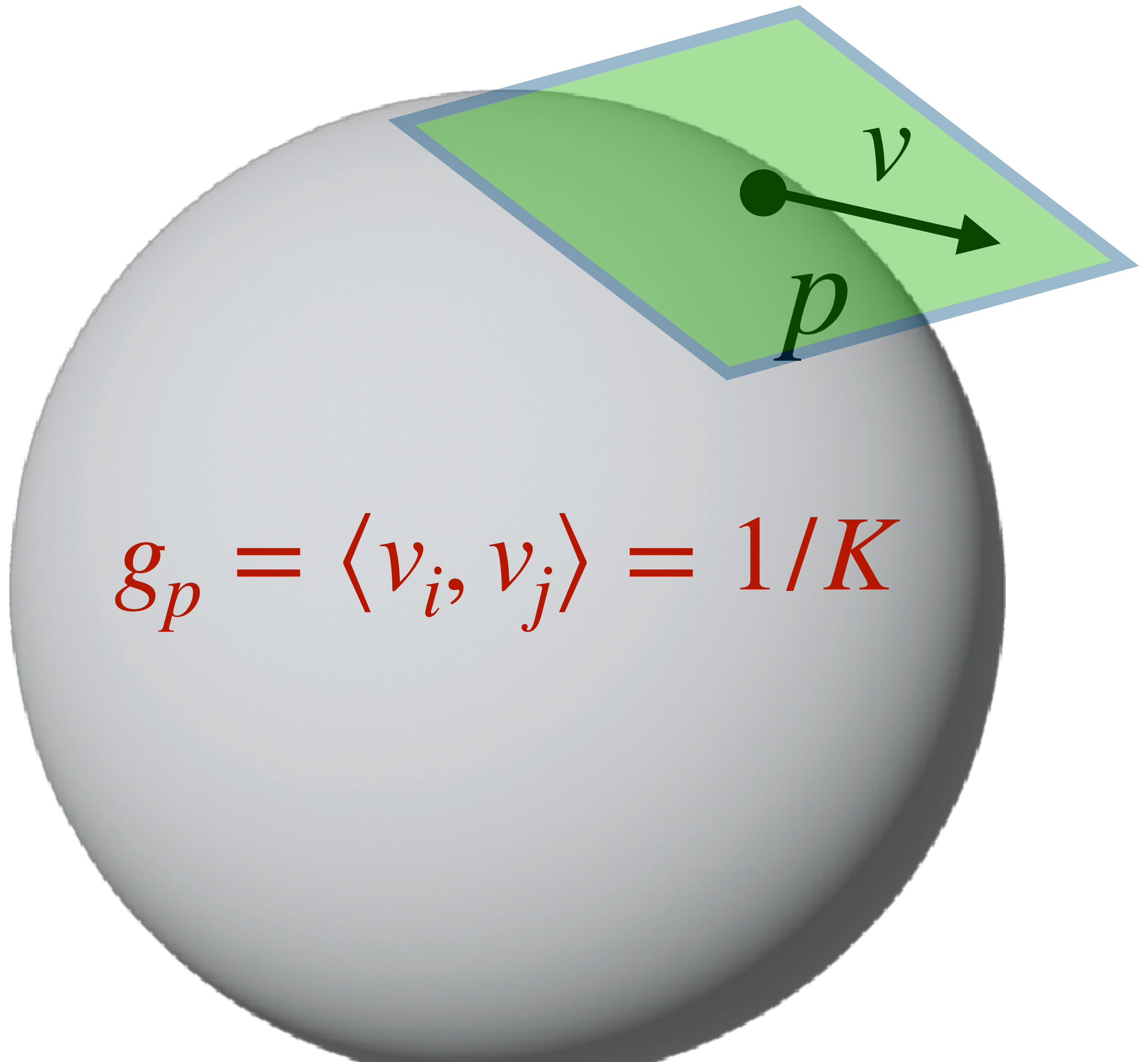
- A curve: A smooth map $\gamma : [-1,1] \to \mathscr{M}, \gamma(0) = p$.
- Tangent Basis: Any $v \in T_p\mathscr{M}$ can be expressed as a linear combination of basis vectors which are taken from the chart (U_i, ϕ_i) (by pulling them back via ϕ_1^{-1}).

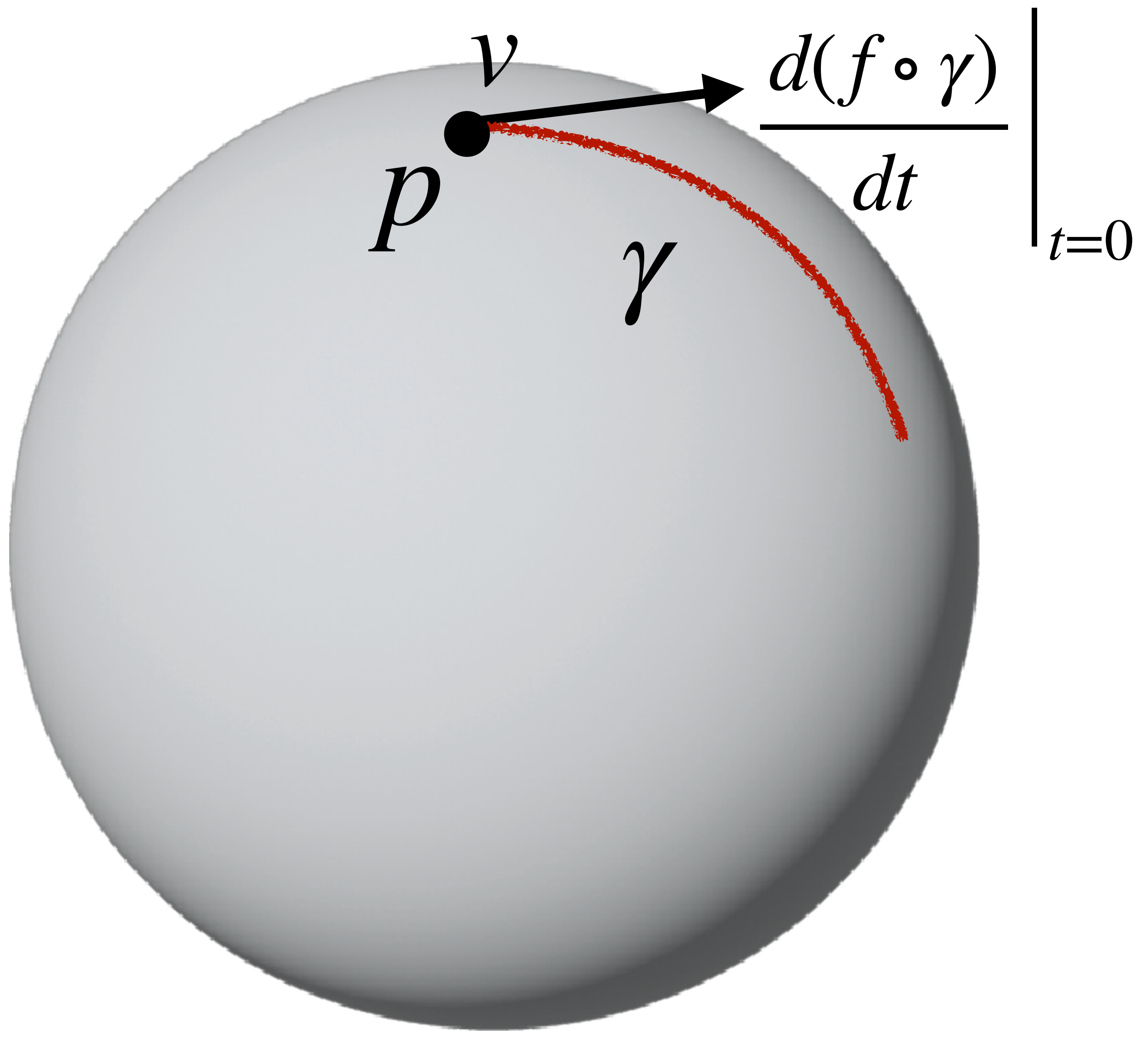


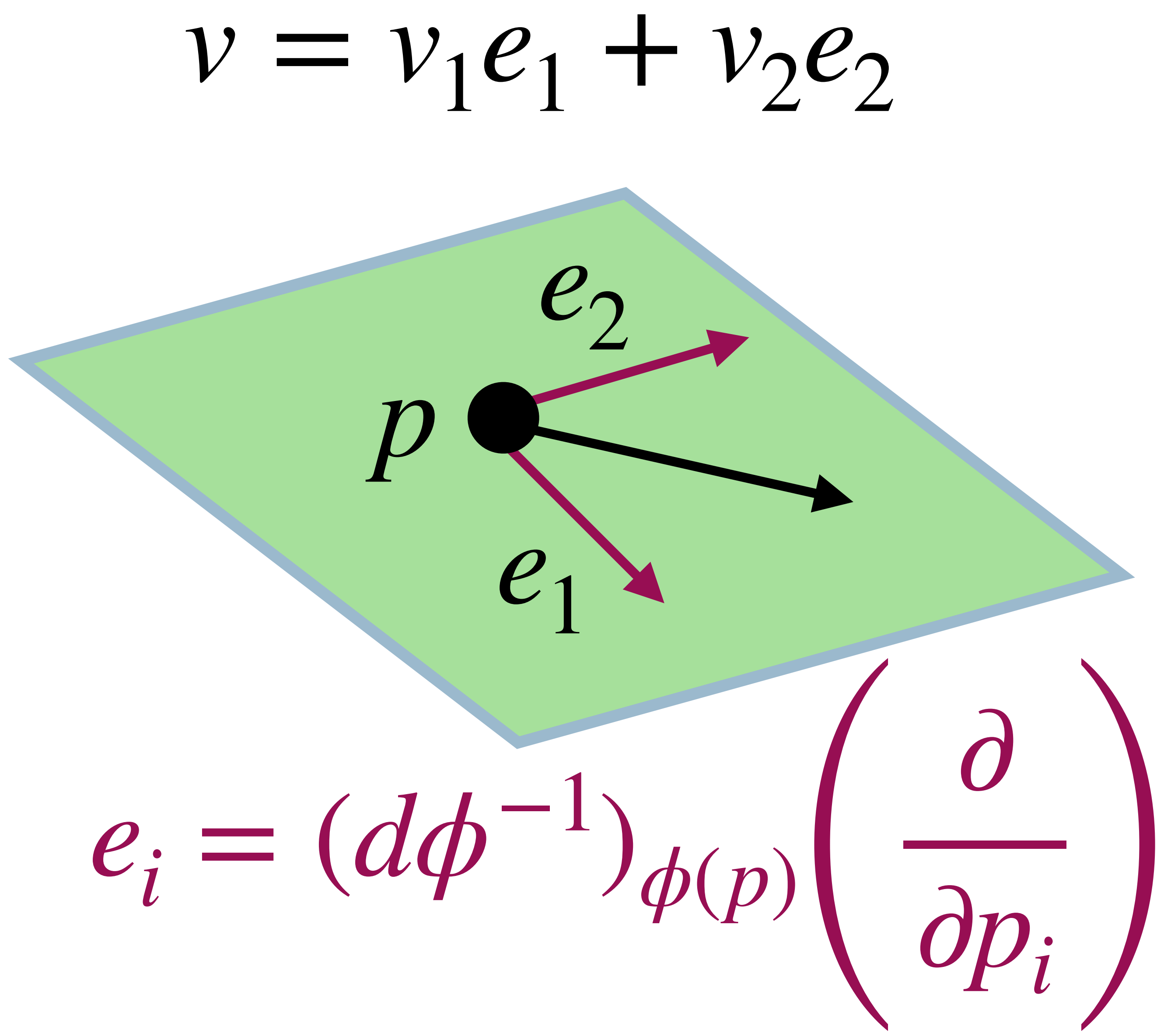


A bit more on Tangent Spaces

- A curve: A smooth map $\gamma : [0,1] \to \mathcal{M}, \gamma(0) = p$.
- Tangent Basis: Any $v \in T_p\mathcal{M}$ can be expressed as a linear combination of basis vectors which are taken from the chart (U_i, ϕ_i) (by pulling them back via ϕ_1^{-1}).
- Let $p = (p_1, \ldots, p_d) = \phi(p)$ be local coordinates and $d\phi_p : T_p\mathcal{M} \to T_{\phi(p)}\mathbb{R}^d$







Riemannian Manifolds

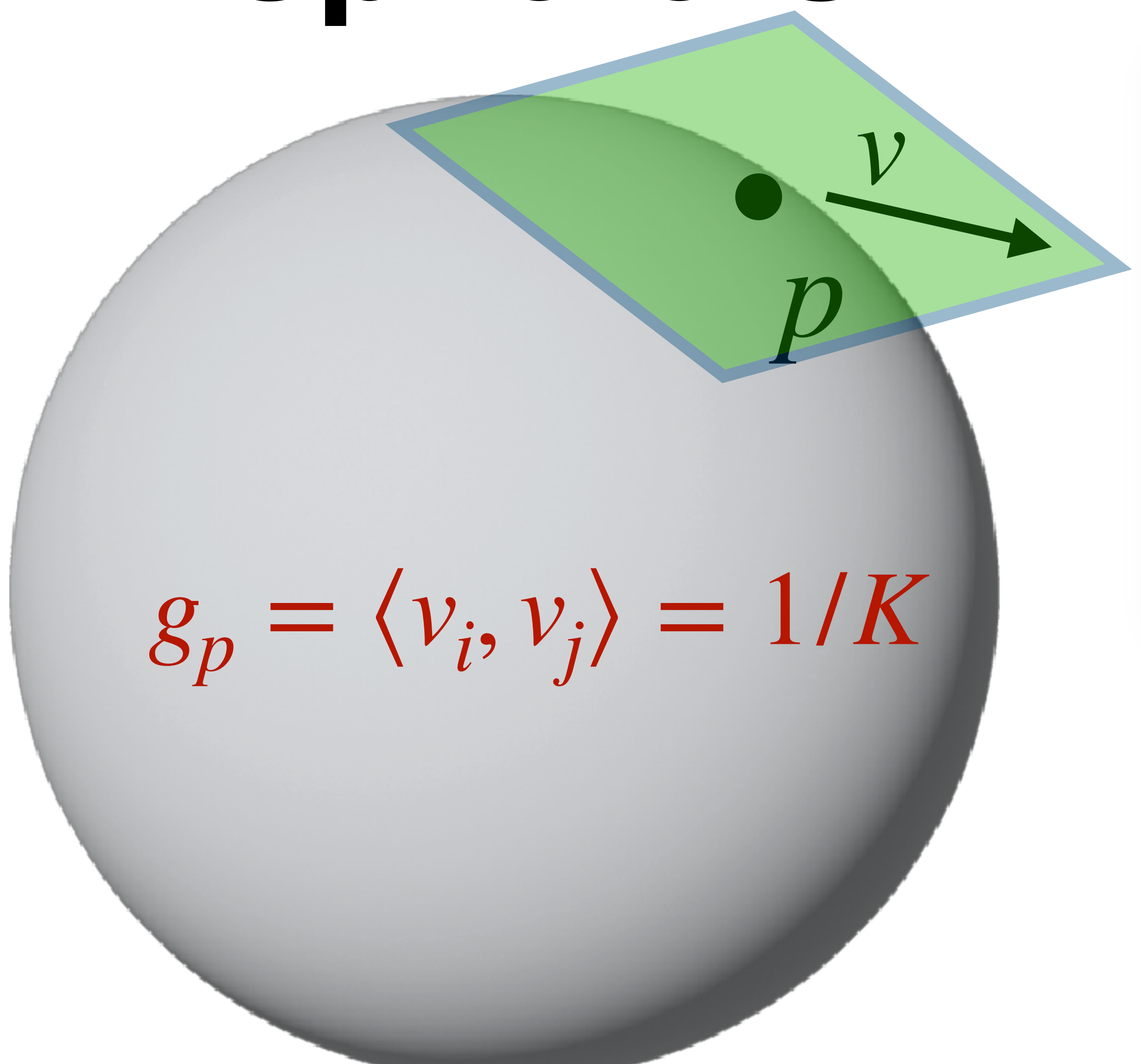
- Tangent space: For each $p \in \mathscr{M}$, a tangent vector is a smooth map $v : \mathscr{F} \to \mathbb{R}^n$.
- Riemannian metric: Inner product $g_p = \langle \cdot , \cdot \rangle_p$ on each tangent space that varies smoothly.

$$\|u_t^\theta(x) - u_t(x \mid z)\|^2$$

Q1. How do we compute norms?

Q1. How do we get $x_t = \alpha_t x_1 + \sigma x_0$?

Sphere $\mathbb{S}^2$

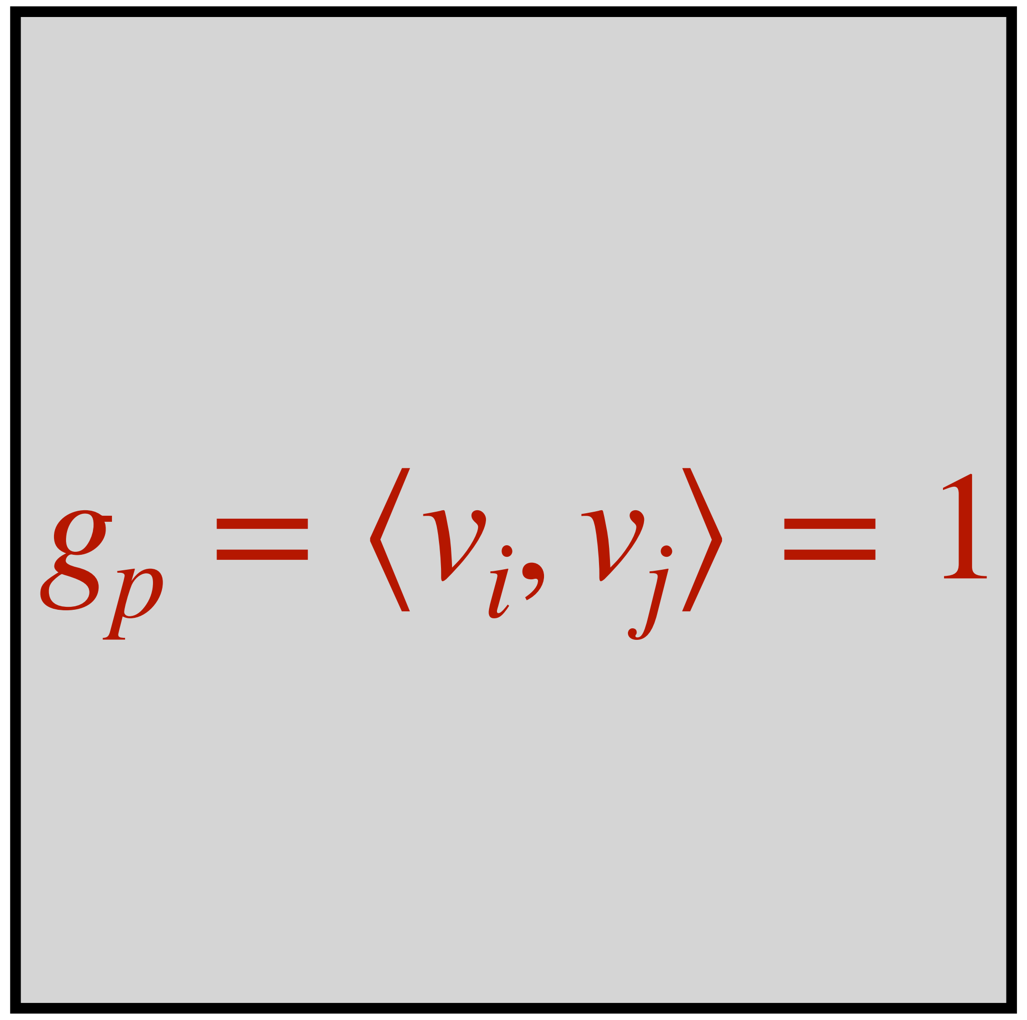


We add more structure to $\mathscr{M}$ by **choosing** a “metric”.

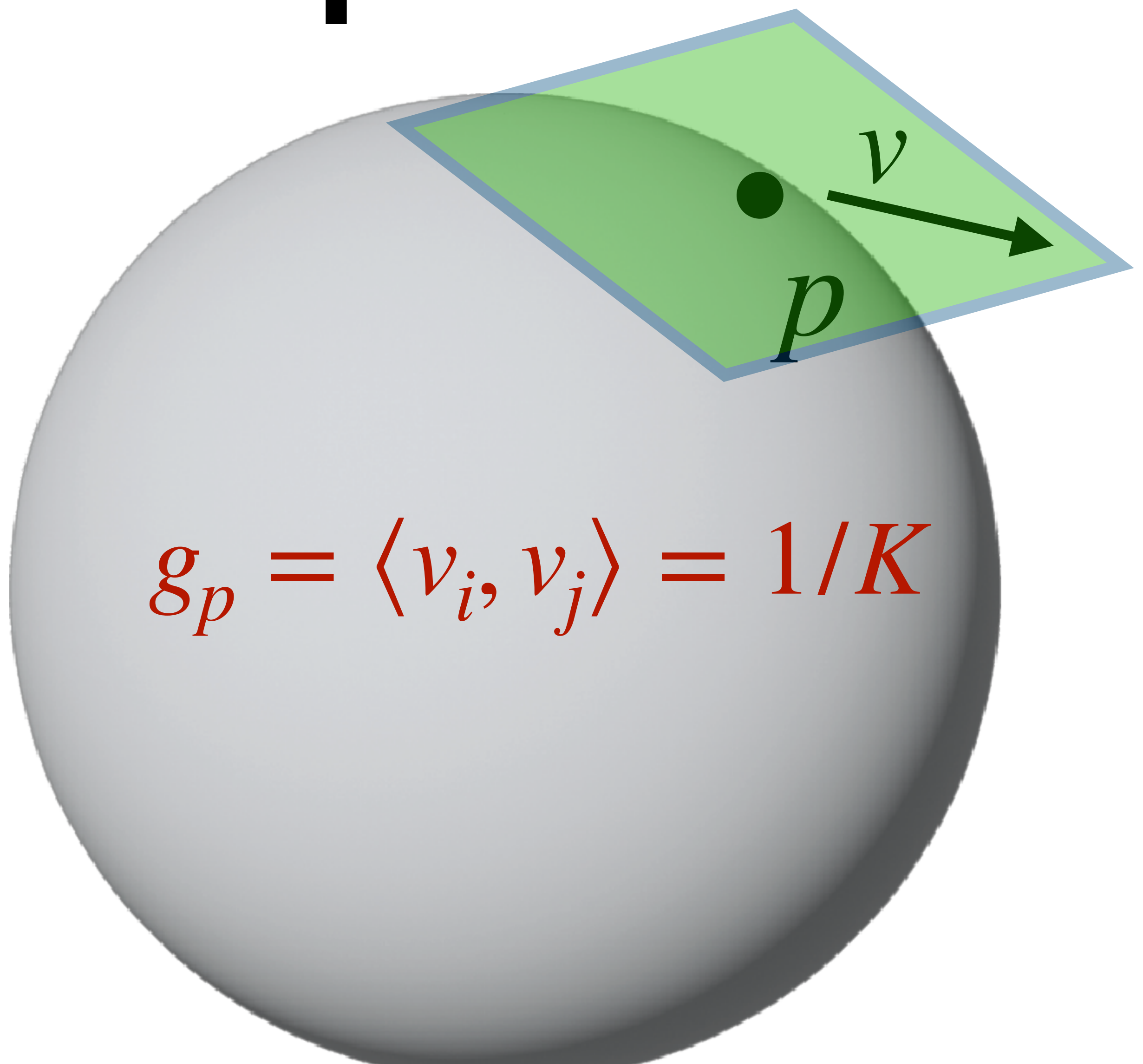
Riemannian Manifolds

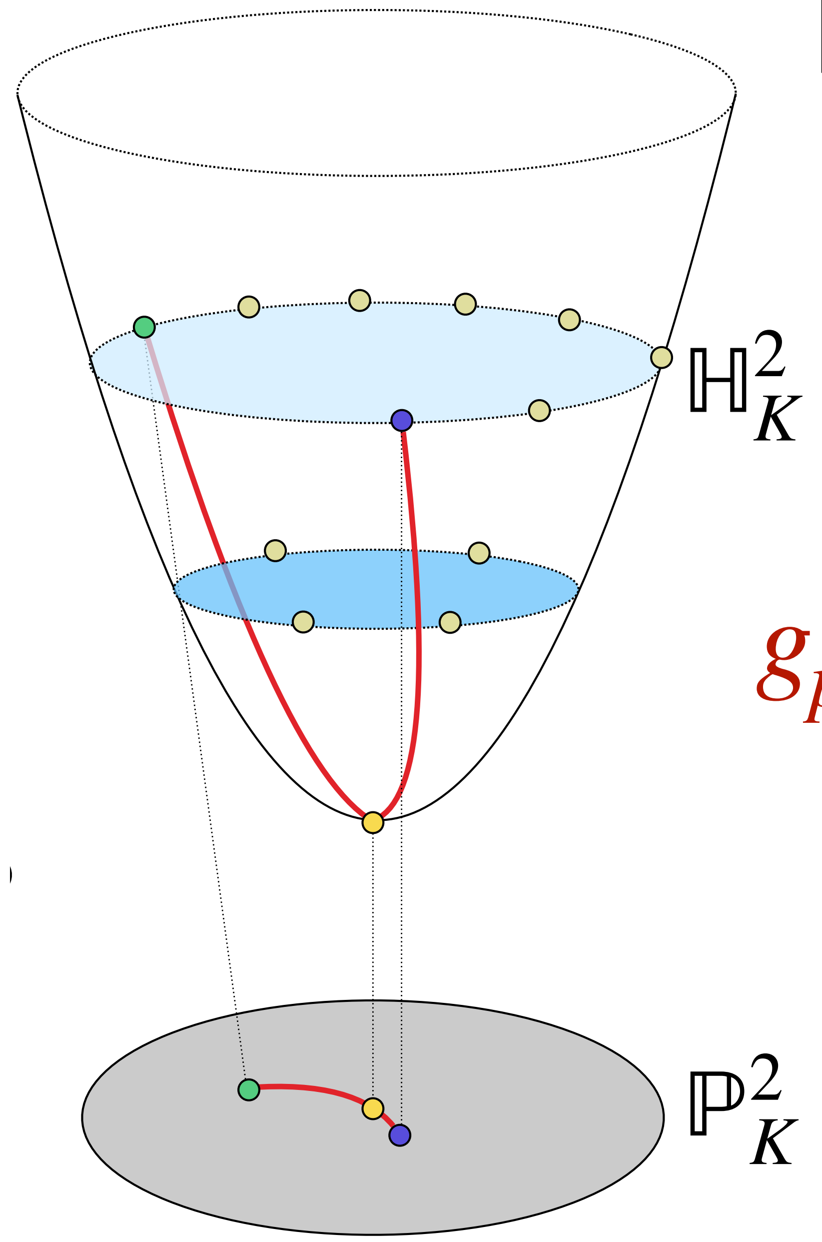
- Tangent space: For each $p \in \mathscr{M}$, a tangent vector is a smooth map $v : \mathscr{F} \to \mathbb{R}^n$.
- Riemannian metric: Inner product $g_p = \langle \cdot , \cdot \rangle_p$ on each tangent space that varies smoothly.
- Riemannian manifold: A smooth manifold equipped with an inner product $(\mathscr{M}, g)$

Euclidean $\mathbb{R}^2$



Sphere $\mathbb{S}^2$





Hyperboloid $\mathbb{H}^2_K$

$g_p = \langle v_i, v_j \rangle = -1/K$

Why are metrics important?

- Riemannian metric is not the same as saying "metric space"
- $g_p := \langle , \rangle_g$ can be used to
    - Lengths of vectors
    - Distances
    - Angles.

$$\langle u, v \rangle_g = u^T G v$$

Tangent vector

(Positive definite) matrix representation of the metric

Why are metrics important?

- Riemannian metric is not the same as saying “metric space”.
- A Riemannian metric allows us to measure many things: distances, lengths of vectors, angles. It is the main gadget that allows actual computation.

Norm of a vector $u \in T_p\mathcal{M}$

$$\|u\|_g = \sqrt{\langle u, u \rangle_g} = \sqrt{u^T G u}$$

Measures length of u using G

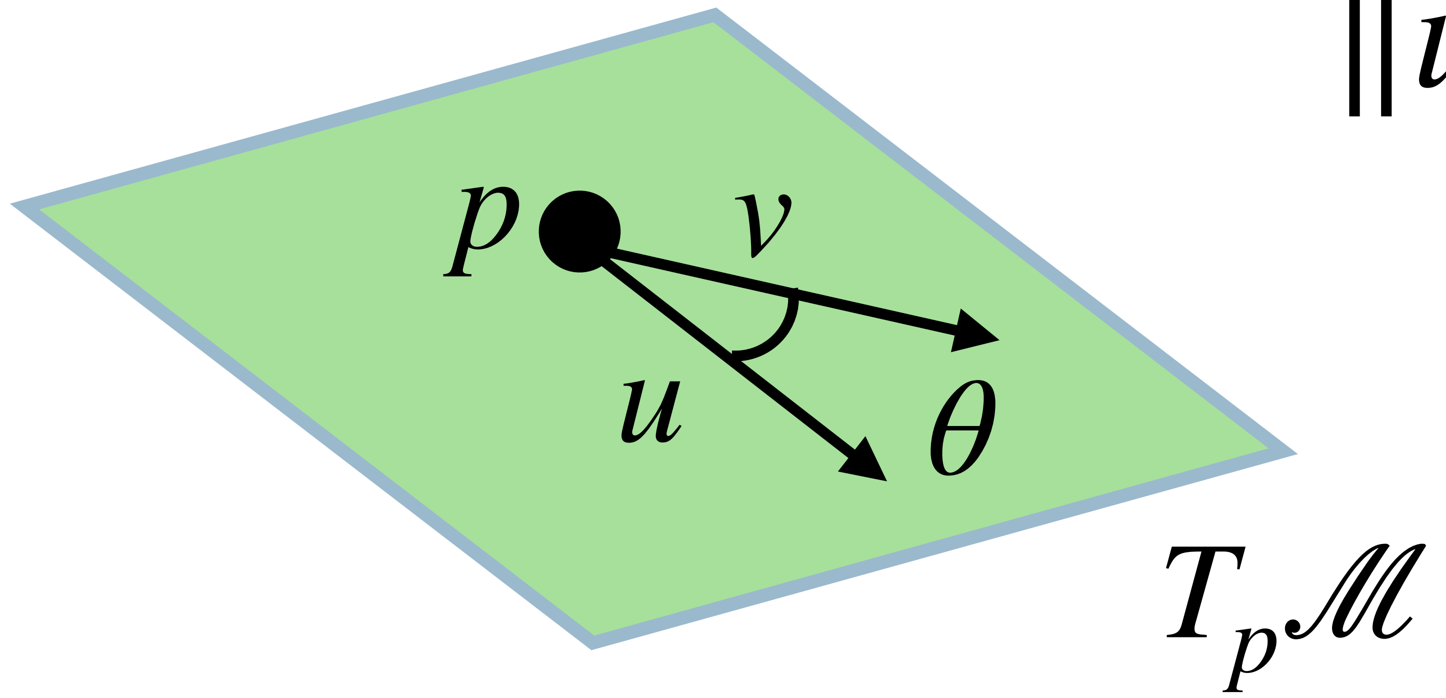
$$\|u_t^\theta(x) - u_t(x \mid z)\|^2$$

$$\|u_t^\theta(x) - u_t(x \mid z)\|_g^2$$

Norm changes!

Angle between $u, v \in T_p\mathcal{M}$

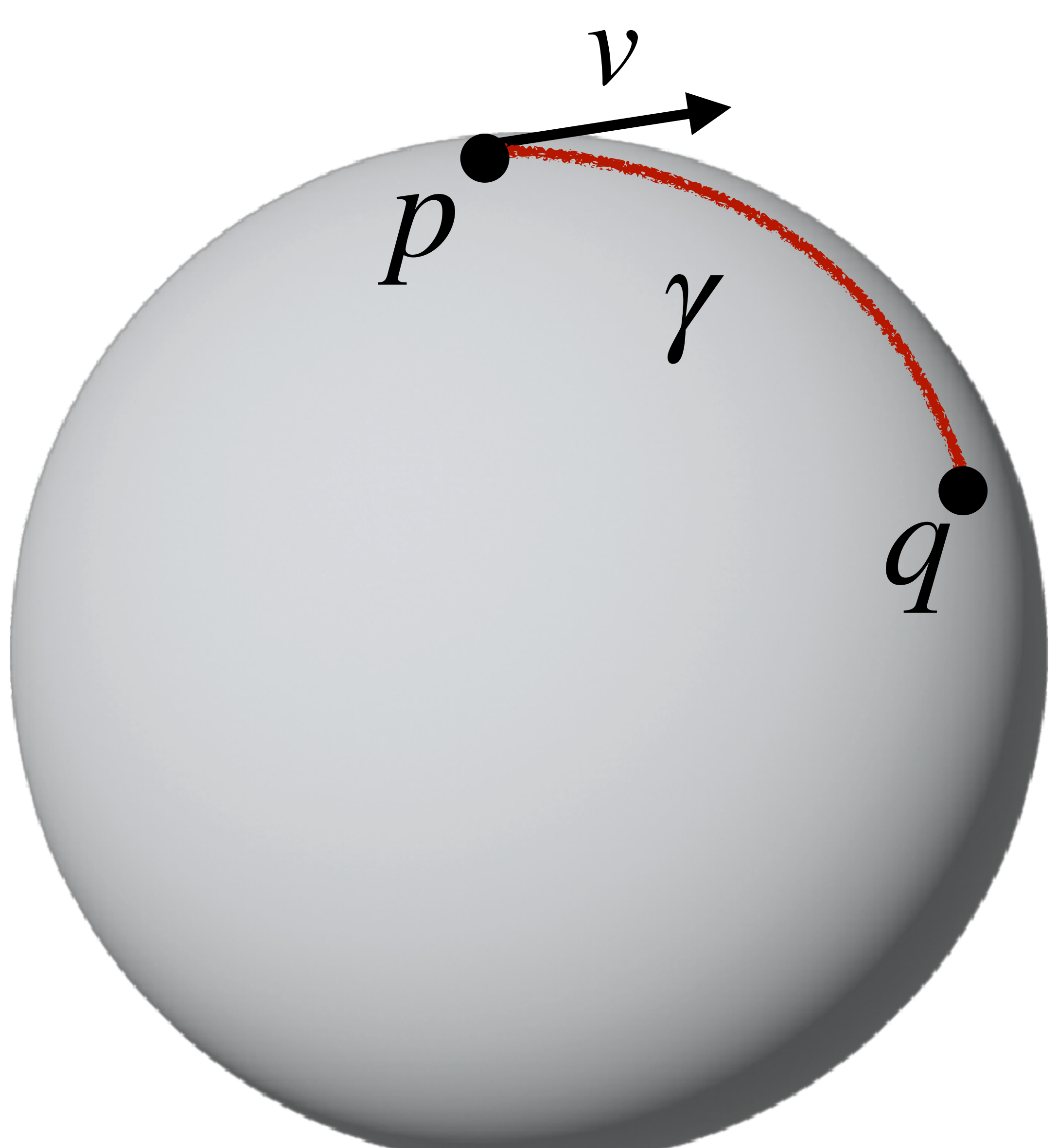
$$\cos\theta = \frac{\langle u, v \rangle_g}{\|u\|_g \|v\|_g}$$



Measuring distances and geodesics

- A curve: A smooth map $\gamma : [0,1] \to \mathscr{M},\ \gamma(-1) = p, \gamma(1) = q$.

How do we measure the distance between two points on $p, q \in \mathscr{M}$ linked by γ?



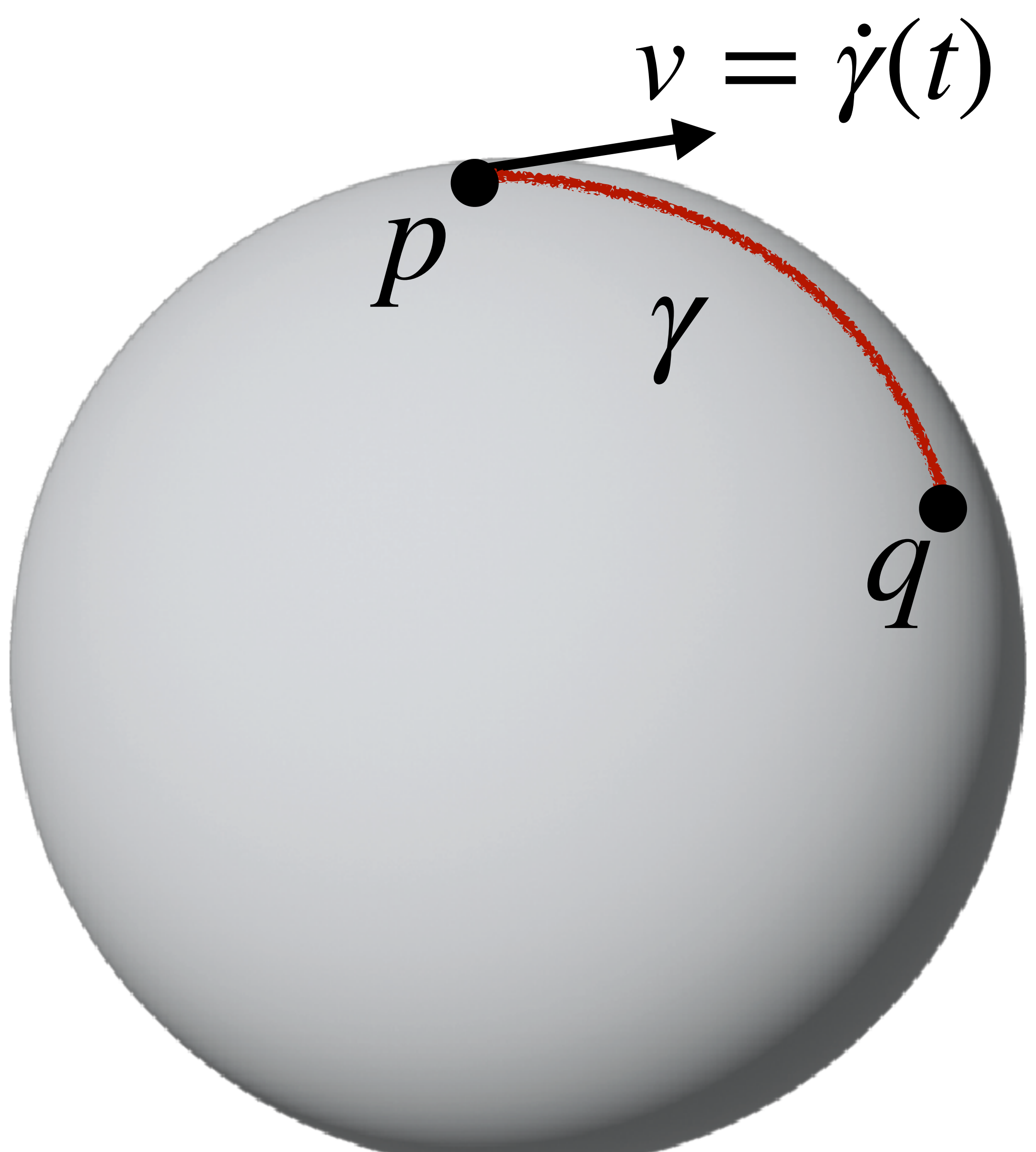
Measuring distances and geodesics

- Main idea: Measure the norm of the tangent vector $\dot{\gamma}(t)$ along the curve

$$\text{length}(\gamma) = \int_0^1 \|\dot{\gamma}(t)\|^2_{g(\gamma(t))} \mathrm{d}t = \int_0^1 \sqrt{\dot{\gamma}(t)^T G \dot{\gamma}(t) \mathrm{d}t} \longrightarrow G \text{ measures length of } \dot{\gamma}(t)$$

Distance is the shortest curve γ

$$d_g(p,q) = \inf_{\gamma} \int_0^1 \|\dot{\gamma}(t)\| dt$$



Facts:

- Shortest path is a geodesic
- It is also the “straightest”
- Geodesics minimize Kinetic Energy.

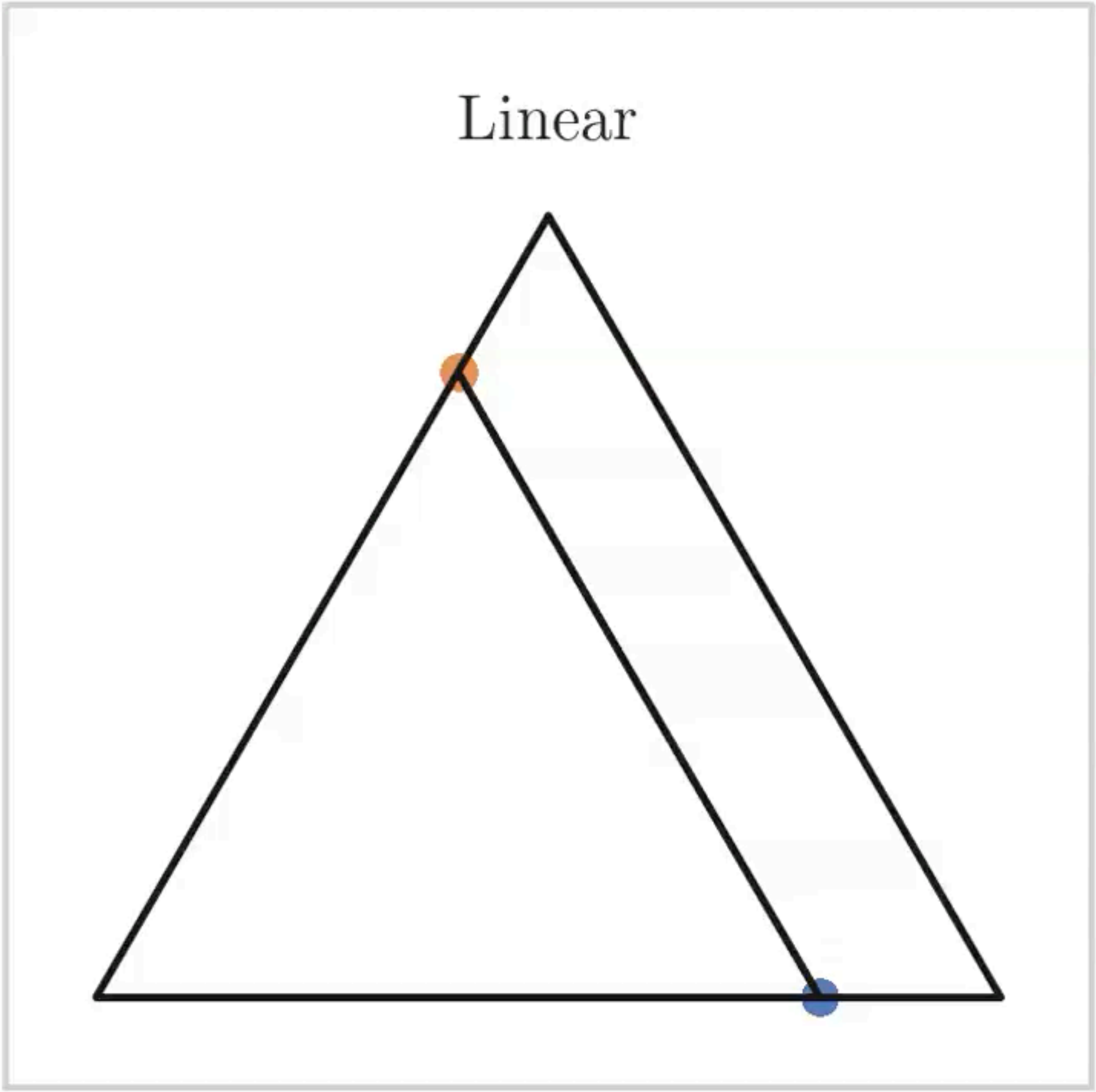
Geodesics

- Different Metrics Induce different Geodesics on the same space

$$\text{length}(\gamma) = \int_0^1 ||\dot{\gamma}(t)||^2_{g(\gamma(t))} \mathrm{d}t = \int_0^1 \sqrt{\dot{\gamma}(t)^T G \dot{\gamma}(t)} \mathrm{d}t$$

Example: Geodesics on the probability simplex

- Euclidean metric “Linear”
- Fisher-Rao metric



Distances allow us to ...

Flows

$$L_{\text{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| u_t^\theta(x) - u_t(x \,|\, z) \|_g^2$$

$$L_{\text{CFM}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| d(\hat{x}_1^\theta(x), x_1)_g \|_g^2$$

ODE $\quad dx_t = u_t(x_t)dt$

Velocity field

Diffusion

$$L_{\text{Diff}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| s_t^\theta(x) - \nabla_x p_t(x \,|\, x_{data}) \|_g^2$$

$$L_{\text{Diff}}(\theta) = \min \mathbb{E}_{t,\, q(z),\, p_t(x|z)} \| d(\epsilon_t^\theta(x), \epsilon_t)_g \|_g^2$$

SDE $\quad dx_t = f_t(x_t)dt + g_t dw_t$

Drift

Diffusion Coefficient

Brownian Motion

But How do we integrate on $\mathscr{M}$?

Inference on Manifolds

$\mathbb{R}^d$

Simulating Flows

ODE $dx_t = u_t(x_t)dt$

$$x_{t+1} = x_t + u_t^\theta \Delta t$$

$\mathcal{M}$

Can not do +

Need $x_{t+1} \in \mathcal{M}$

$\mathbb{R}^d$

Simulating Diffusion

SDE $dx_t = f_t(x_t)dt + g_t dw_t$

$$x_{t+1} = x_t + [f(x_t) - g_t^2 s_t^\theta(x_t)]\Delta t + g_t\sqrt{|\Delta t|}z_t$$

$$z_t \sim N(0,1)$$

$\mathcal{M}$

Can not do +

Need $x_{t+1} \in \mathcal{M}$

Need z_t to be Brownian motion on $\mathcal{M}$

Manifold Operations

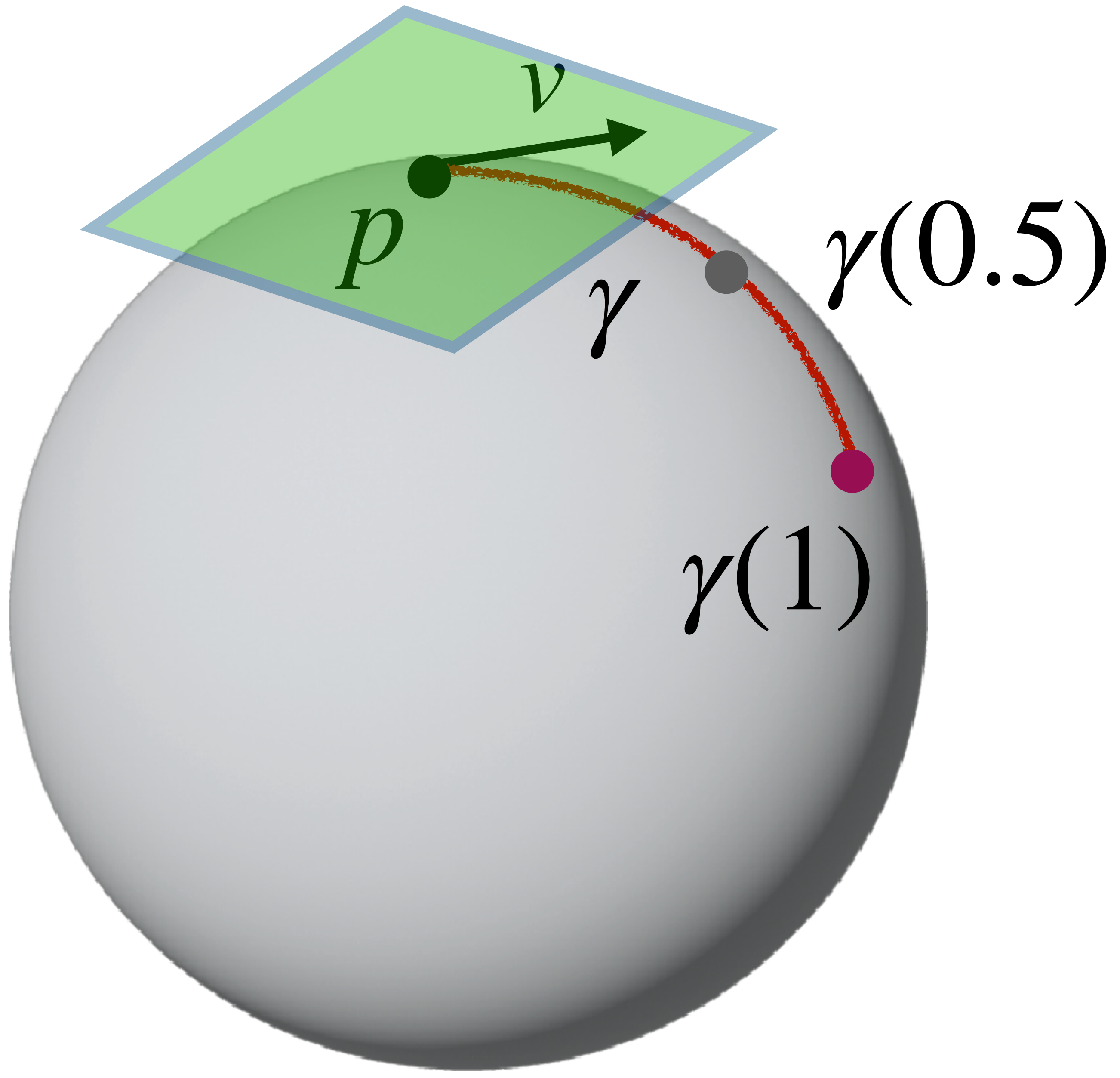
How do we move from a the tangent space back to the manifold? → Exponential Map

How do we move from the manifold to a tangent space? → Logarithmic Map

How do we move vectors between tangent spaces? → Parallel Transport

Exponential Map

- Exponential map: Takes a tangent vector $v \in T_p\mathscr{M}$ and transports it along the unique geodesic which satisfies $\gamma(0) = p$ and $\dot{\gamma}(0) = v$ to the point $\exp_p(v) = \gamma(1)$.
- Output of the exponential map is a point on $\mathscr{M}$.
- Effectively we travel a unit of time along γ.
- Conceptually like "addition" in Euclidean space, case in point $\exp_p(v) = p + v, \forall p \in \mathbb{R}^n$



$$\exp_p(v) = \cos\left(||v||_2\right)p + \sin\left(||v||_2\right)\frac{v}{||v||_2}$$

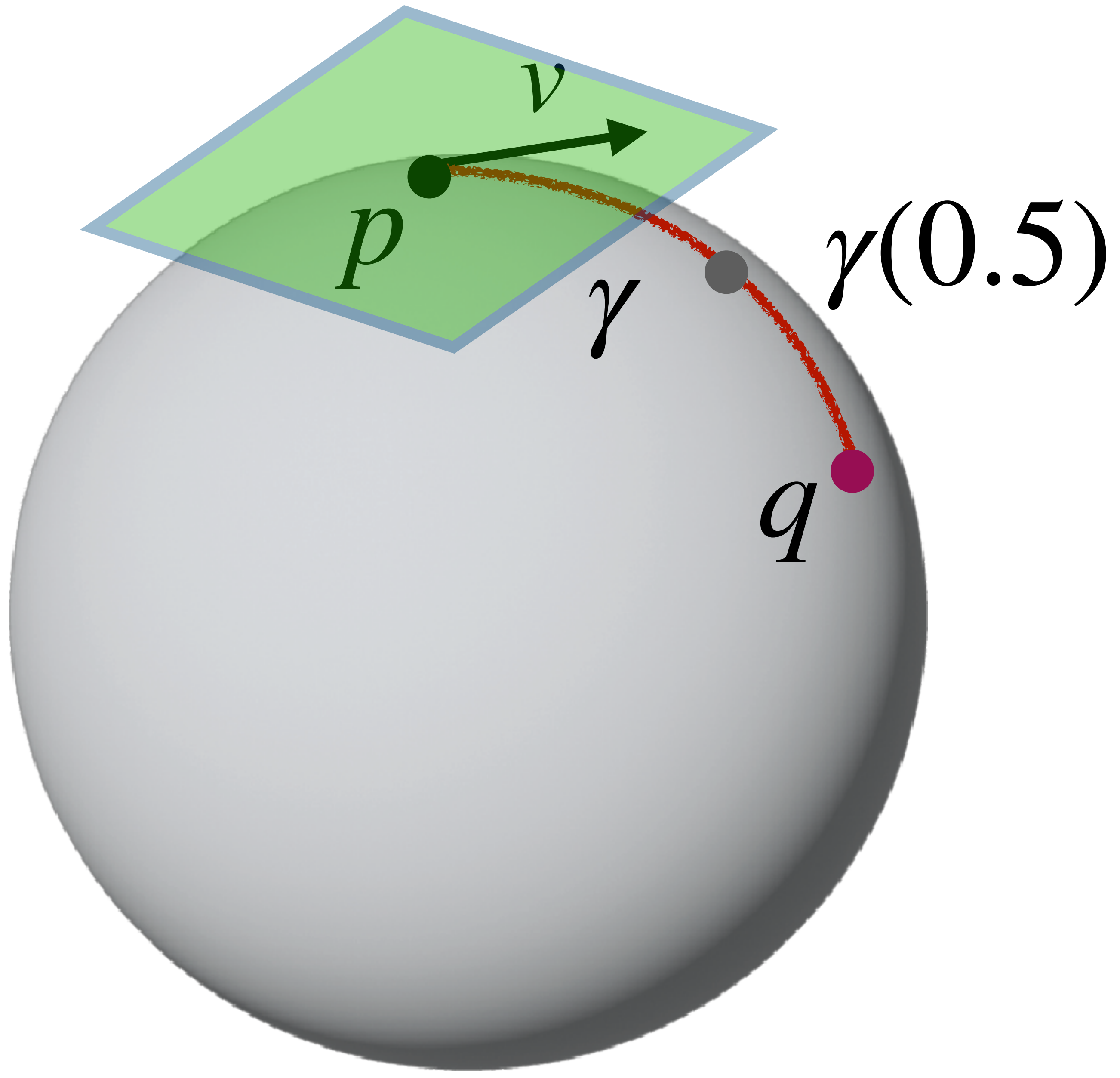
Logarithmic Map

- Logarithmic map: $\log_p : \mathcal{M} \rightarrow T_p\mathcal{M}$. Takes a point on $\mathcal{M}$ back to the tangent space of a base point by following γ.

- Output of the logarithmic map is $v \in T_p\mathcal{M}$.

- (usually) inverse of the exponential map.

- The log map is well-defined only in a neighbourhood of p where $\exp_p$ is a diffeomorphism



$$\log_p(q) = d(p,q)\frac{q - \langle p, q\rangle_2 p}{\|q - \langle p, q\rangle_2\|_2}$$

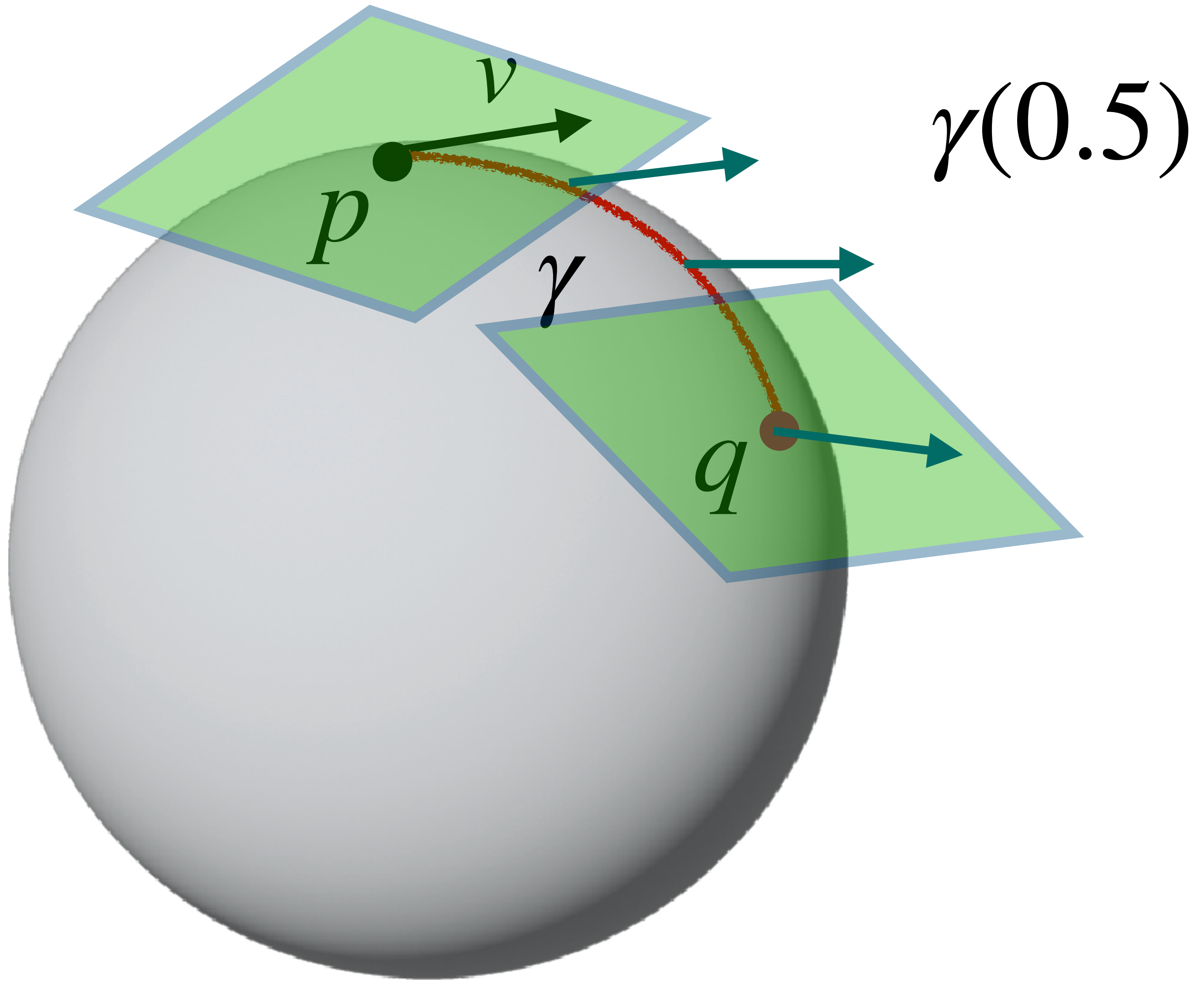
Parallel Transport

- Parallel Transport: $\log_p : \mathscr{M} \rightarrow T_p\mathscr{M}$. Moves a tangent vector v along a curve $\gamma(t)$ such that the vector remains "parallel" to it and lands at another tangent space.

- Length and angles between parallel transported vectors are preserved $\langle v(t), w(t) \rangle_g = \text{constant}$.

- Parallel transport is unique.

- Parallel transport is reversible (along the same curve)



$$\mathrm{PT}_{p \rightarrow q}(v) = v - \frac{\langle q, v \rangle_2}{1 + \langle p, q \rangle_2}(p + q)$$

Manifold Operations in Action

$\mathbb{R}^d$

Target Velocity

$$\|u_t^\theta(x) - u_t(x\,|\,z)\|^2$$

Straight line!

$$\frac{d}{dt}x_t = u_t(x_t\,|\,z) = \frac{x_1 - x_t}{1-t}$$

$\mathcal{M}$

$$\|u_t^\theta(x) - u_t(x\,|\,z)\|_g^2$$

Point
on a Geodesic!

$$\frac{\log_{x_t}(x_1)}{1-t}$$

$\mathbf{SO(3)}$

$$\log(X) \approx \sum_{n=1}^{N} \frac{(-1)^{n-1}}{n}(g-I)^n$$

A rotation matrix

Very expensive to approximate!

$$r = \exp \hat{\boldsymbol{\omega}} = \cos(\omega)I + \sin(\omega)e_\omega + (1-\cos(\omega))e_\omega e_\omega^\top$$

$$\log(r) = \begin{cases} \frac{\omega}{2\sin(\omega)}(r - r^\top) & \text{if } r \neq I, \\ 0 & \text{if } r = I. \end{cases}$$

Inference on Manifolds

$\mathbb{R}^d$

Simulating Flows

ODE $dx_t = u_t(x_t)dt$

$$x_{t+1} = x_t + u_t^\theta \Delta t$$

$\mathbb{R}^d$

Simulating Diffusion

SDE $dx_t = f_t(x_t)dt + g_t dw_t$

$$x_{t+1} = x_t + [f(x_t) - g_t^2 s_t^\theta(x_t)]\Delta t + g_t \sqrt{|\Delta t|}\, z_t$$

$$z_t \sim N(0,1)$$

$\mathscr{M}$

$$dx_t = u_t(x_t)dt$$

$$x_{t+1} = \exp_{x_t}\left(u_t^\theta \Delta t\right)$$

$\mathscr{M}$

$$x_{t+1} = \exp_{x_t}\left([f(x_t) - g_t^2 s_t^\theta(x_t)]\Delta t + g_t \sqrt{|\Delta t|}\, z_t\right)$$

$$z_t \sim T_p N(0,1)$$

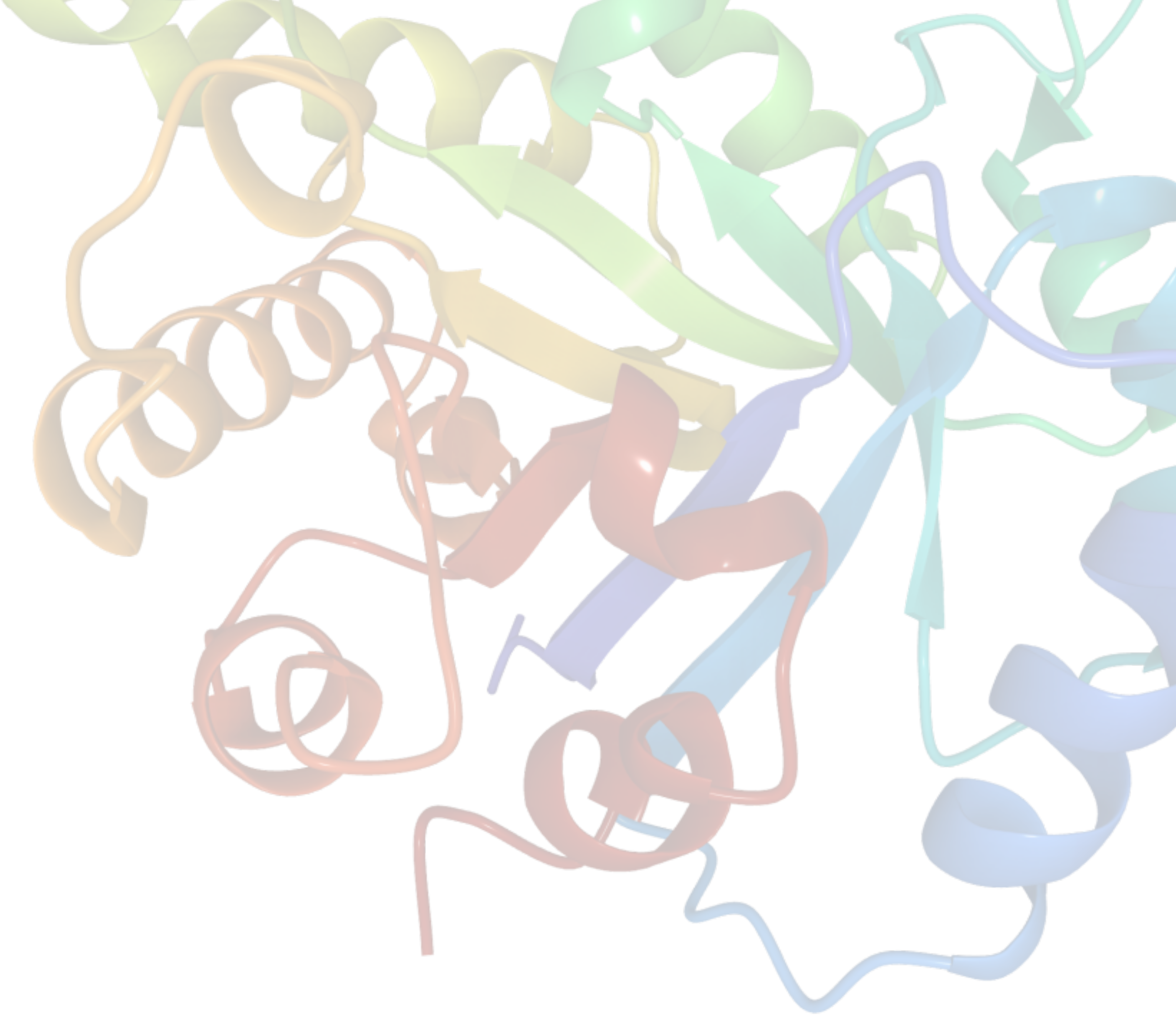
Use tangent space and exp map

Summary

- Each Manifold requires different design considerations for Generative Modeling
  - Tip: Pick a parametrization that makes it “as close” as possible to $\mathbb{R}^d$
  - Tip: Take into account that certain manifold operations might be numerically unstable, e.g. close to the boundary.
  - Tip: Diffusion seems harder to do on Manifolds than Flow Matching. Ask yourself, do you really need an SDE on a manifold?
- There is no Canonical Gaussian distribution, choice of prior is a design decision.

Part III:

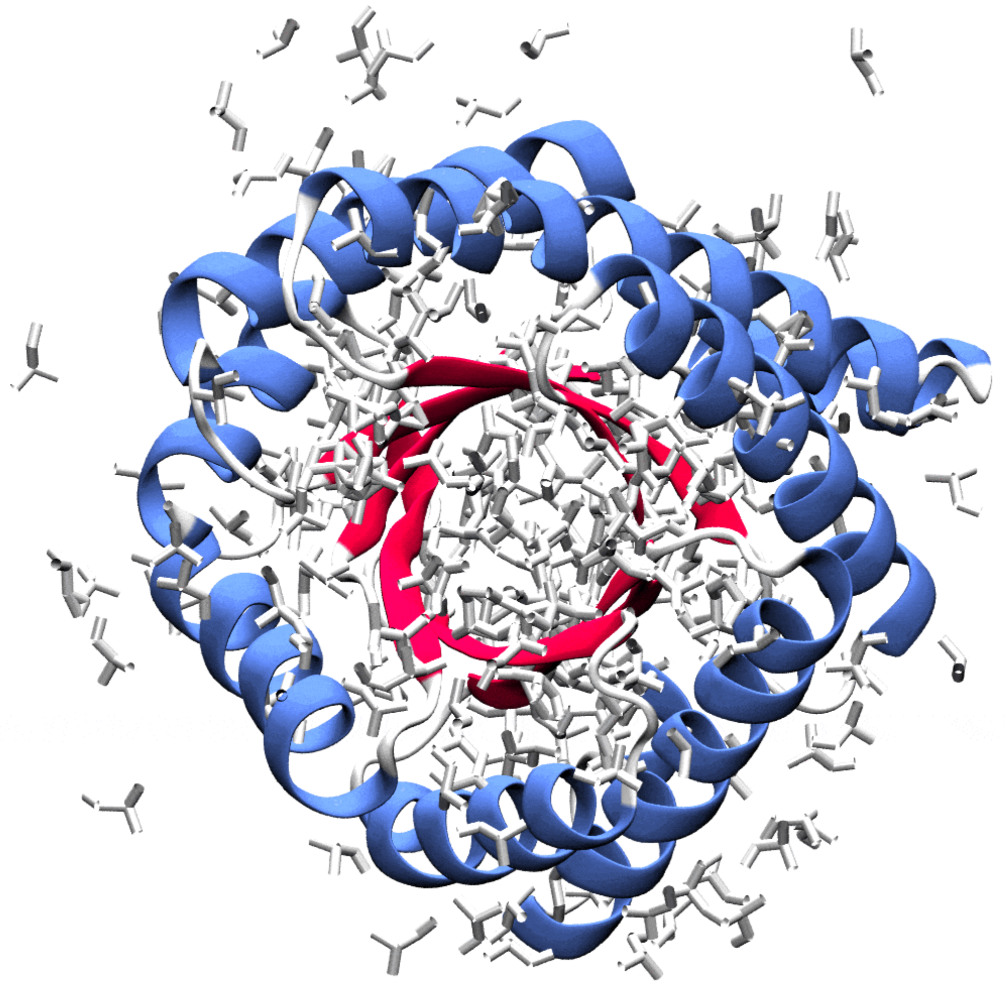
Geometric Generative Models



Modern Applications of Geometric Generative Models

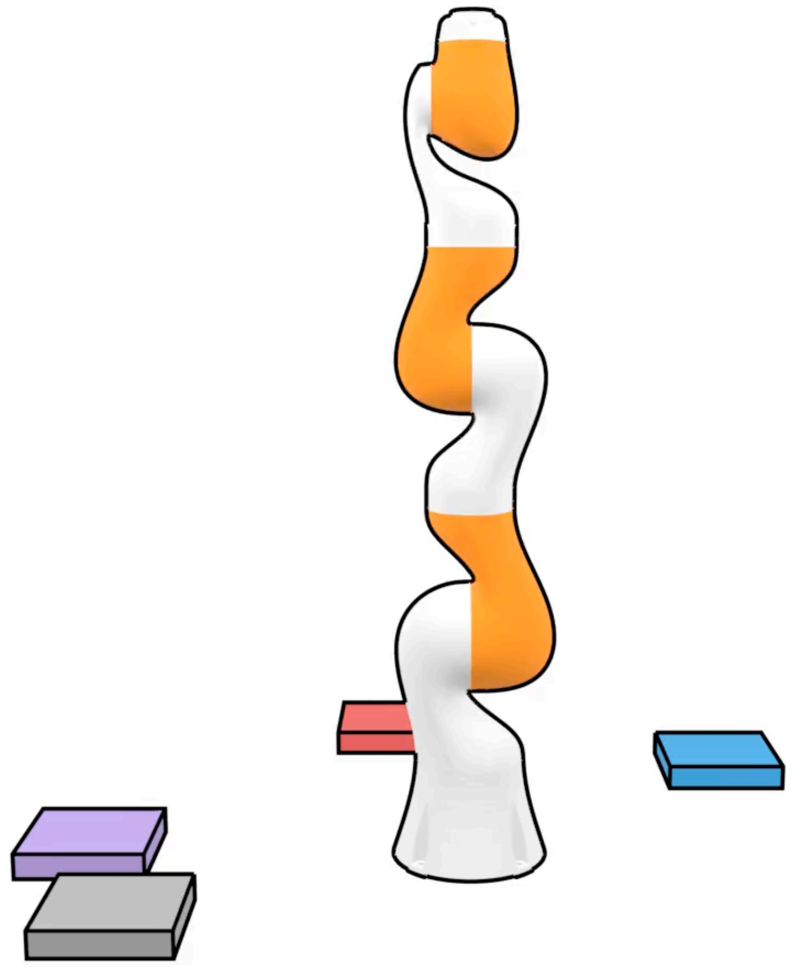
Parametrizable manifolds

Scientific Data



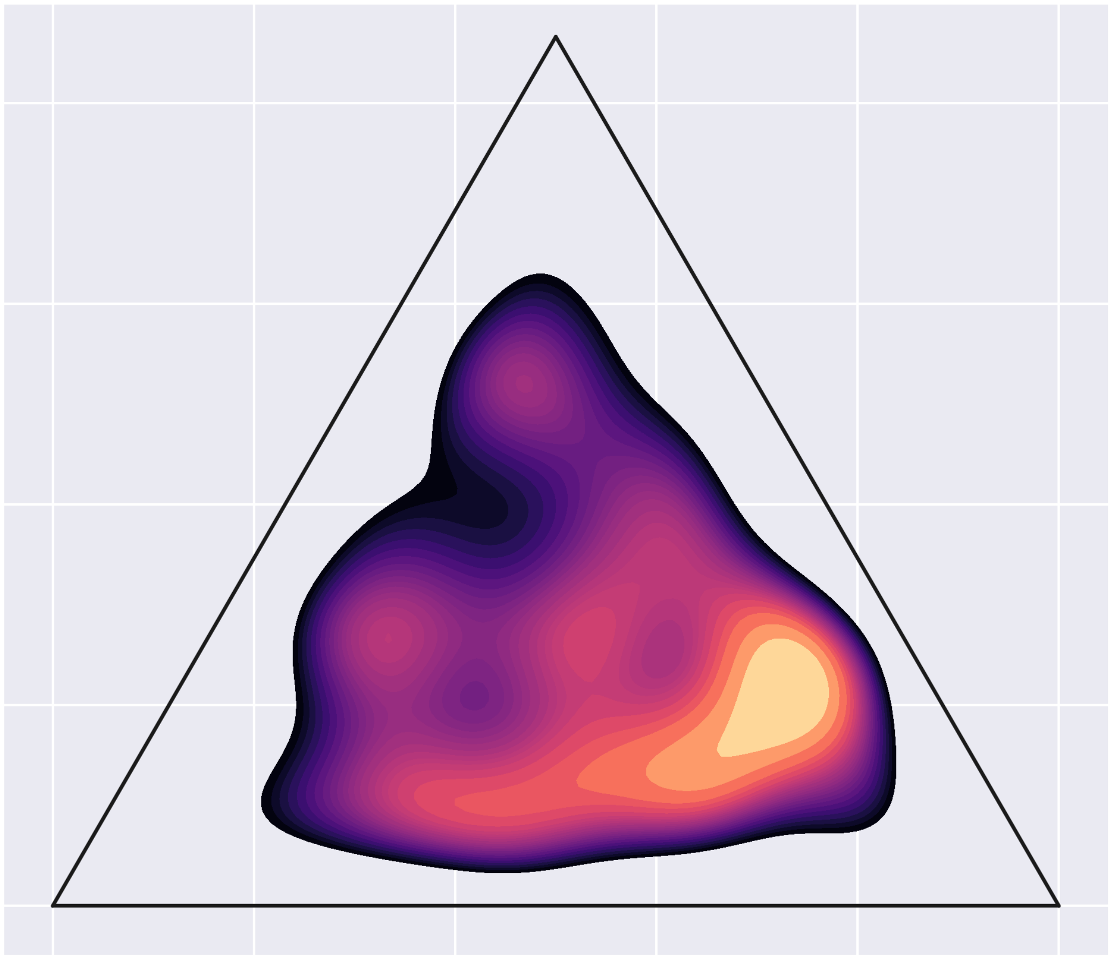
SE(3) equivariant
protein + molecule
representations

Robotics



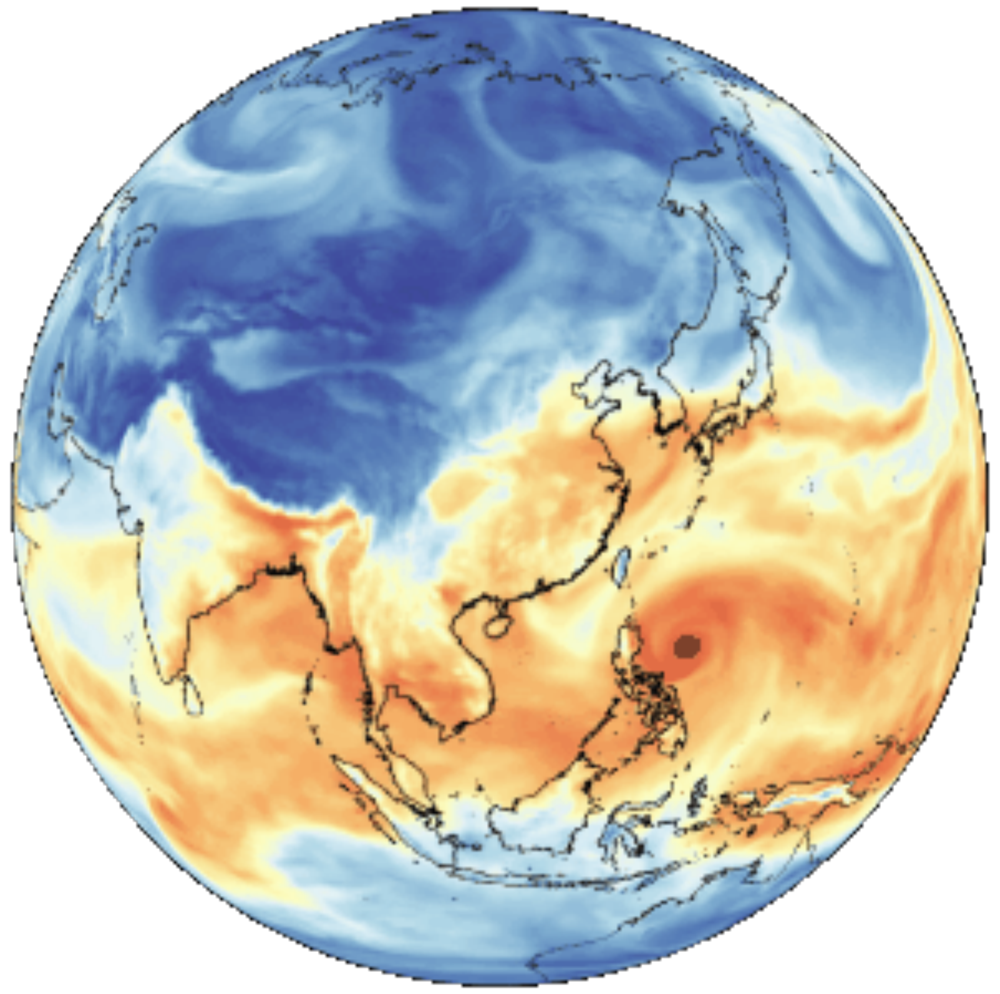
SO(2) invariant
Block stacking

Information Geometry



Fisher-Rao geometry
On the probability Simplex

Climate Modeling

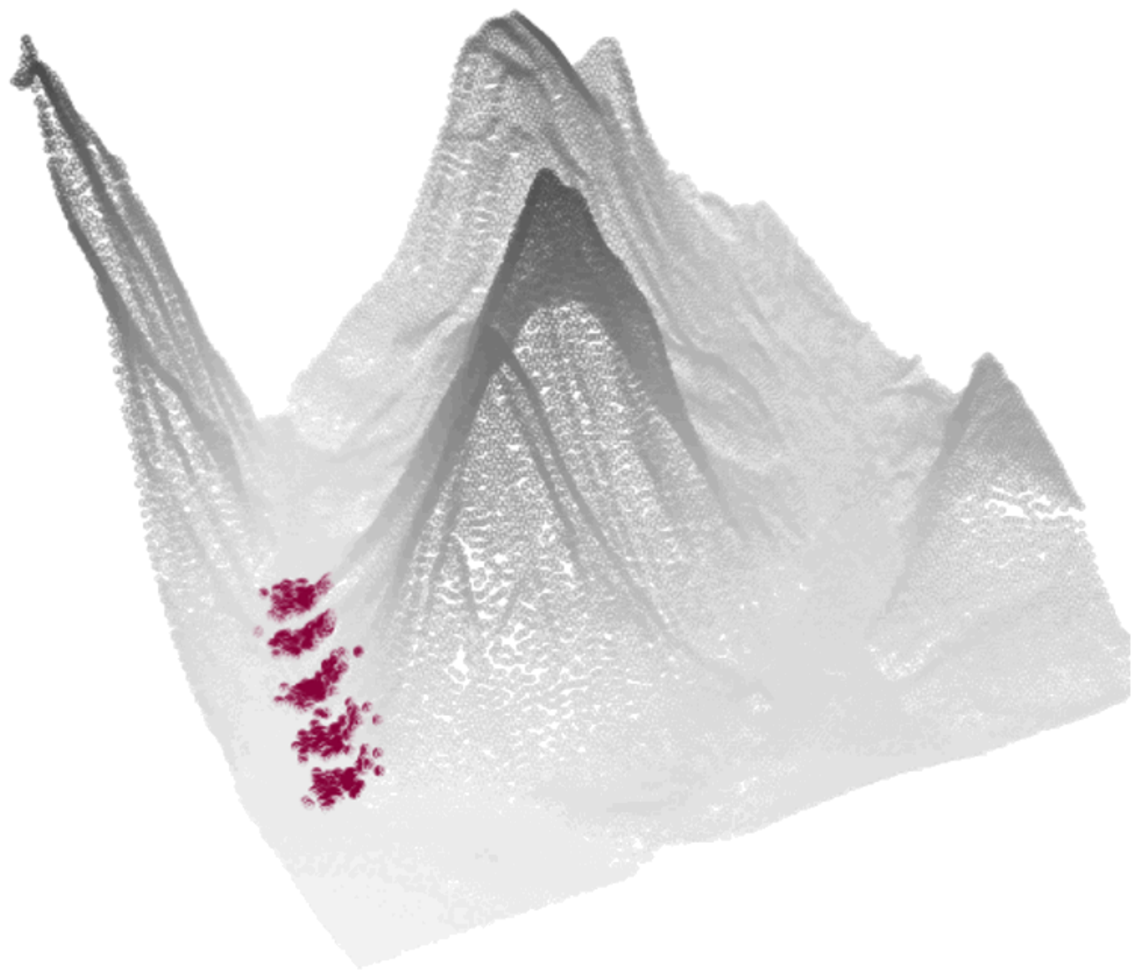


Spherical Geometry $\mathbb{S}^2$

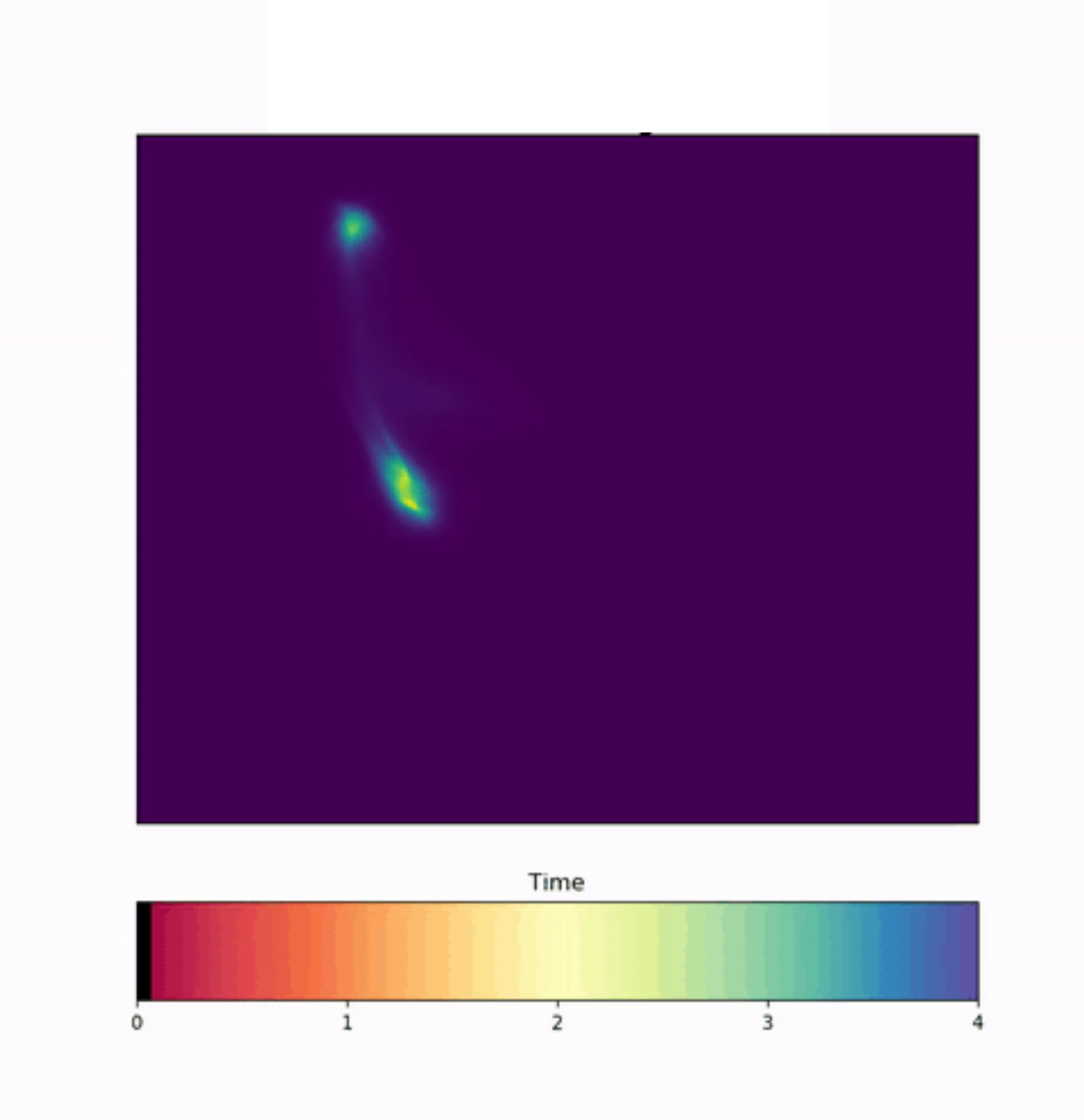
Modern Applications of Geometric Generative Models

non-parametrizable manifolds

3D surfaces



Lidar imaging



Data-driven manifolds

Euclidean to Riemannian Flow Matching

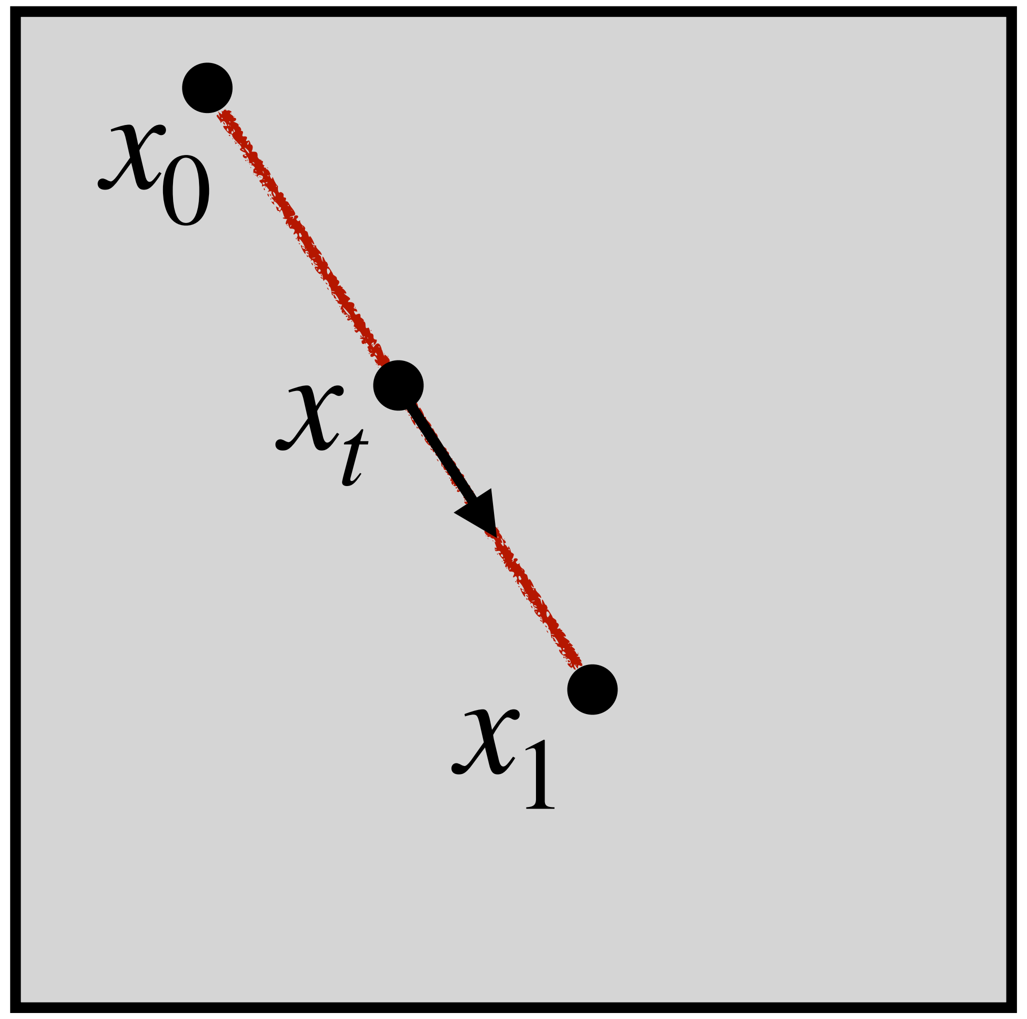
How do you define $z, q(z), x_t, u_t(x_t \mid z)$?

$$z := (x_0, x_1)$$

$$q(z) := q(x_0)q(x_1)$$

$$x_t := (1-t)x_0 + tx_1$$

$$u_t(x_t \mid z) := x_1 - x_0$$



Euclidean to Riemannian Flow Matching

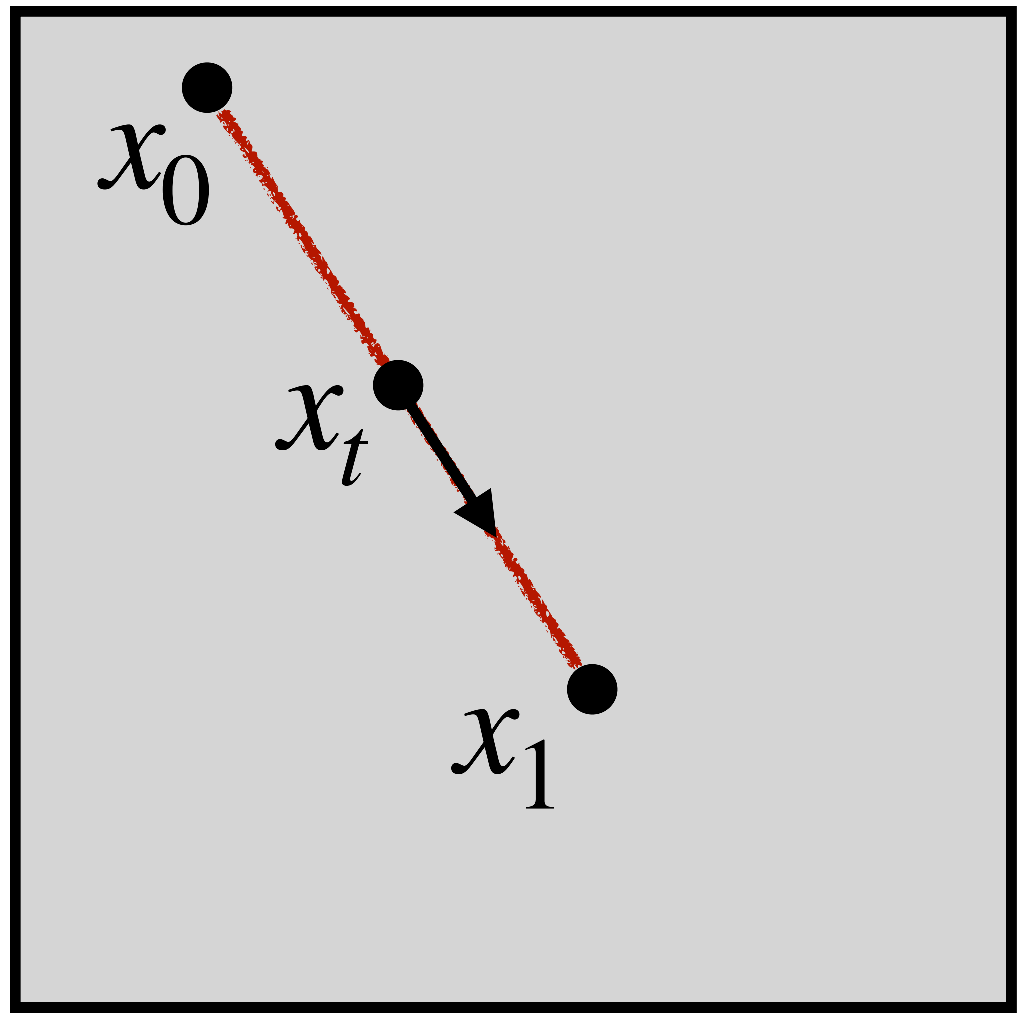
How do you define $z, q(z), x_t, u_t(x_t | z)$?

$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := (1 - t)x_0 + tx_1$

$u_t(x_t | z) := x_1 - x_0$



$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

Euclidean to Riemannian Flow Matching

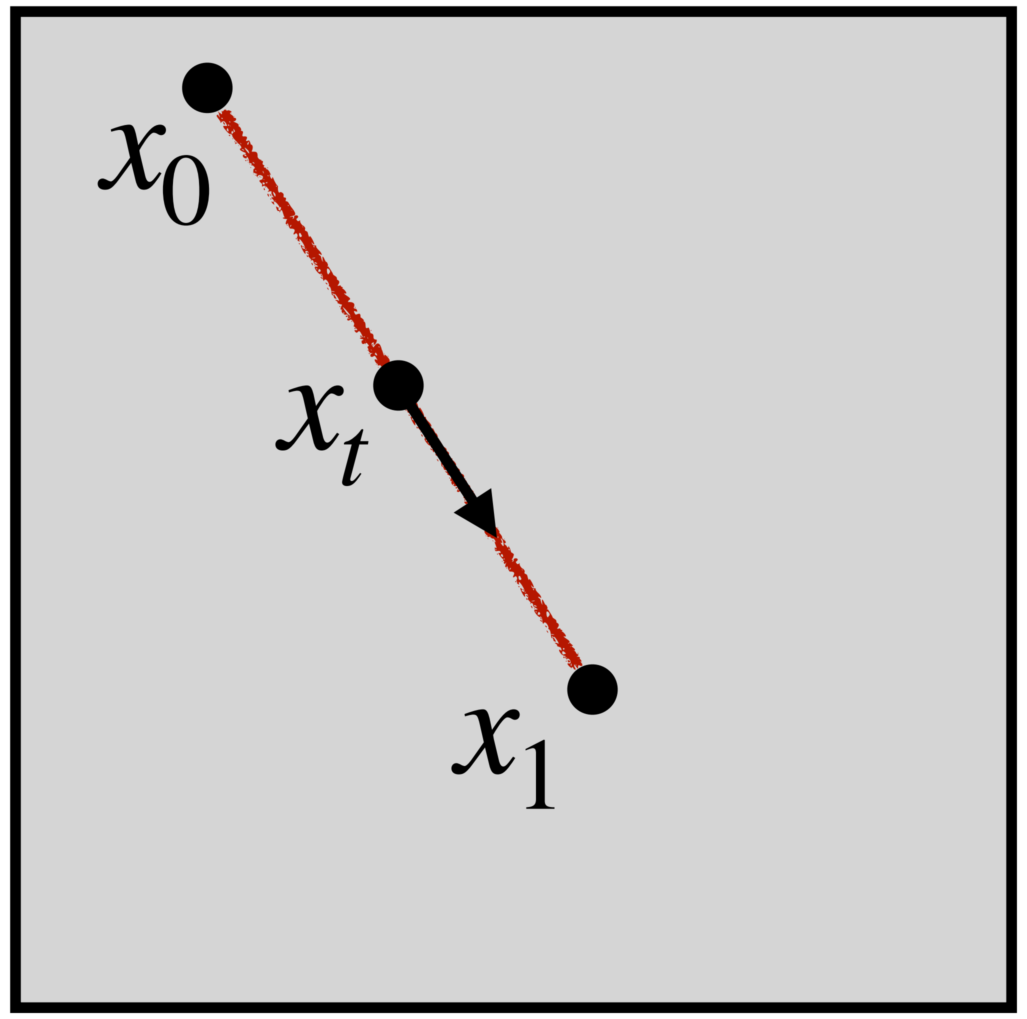
How do you define $z, q(z), x_t, u_t(x_t \mid z)$?

$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := (1 - t)x_0 + tx_1$

$u_t(x_t \mid z) := x_1 - x_0$



$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := \exp_{x_0}(t \log_{x_0} x_1)$

Euclidean to Riemannian Flow Matching

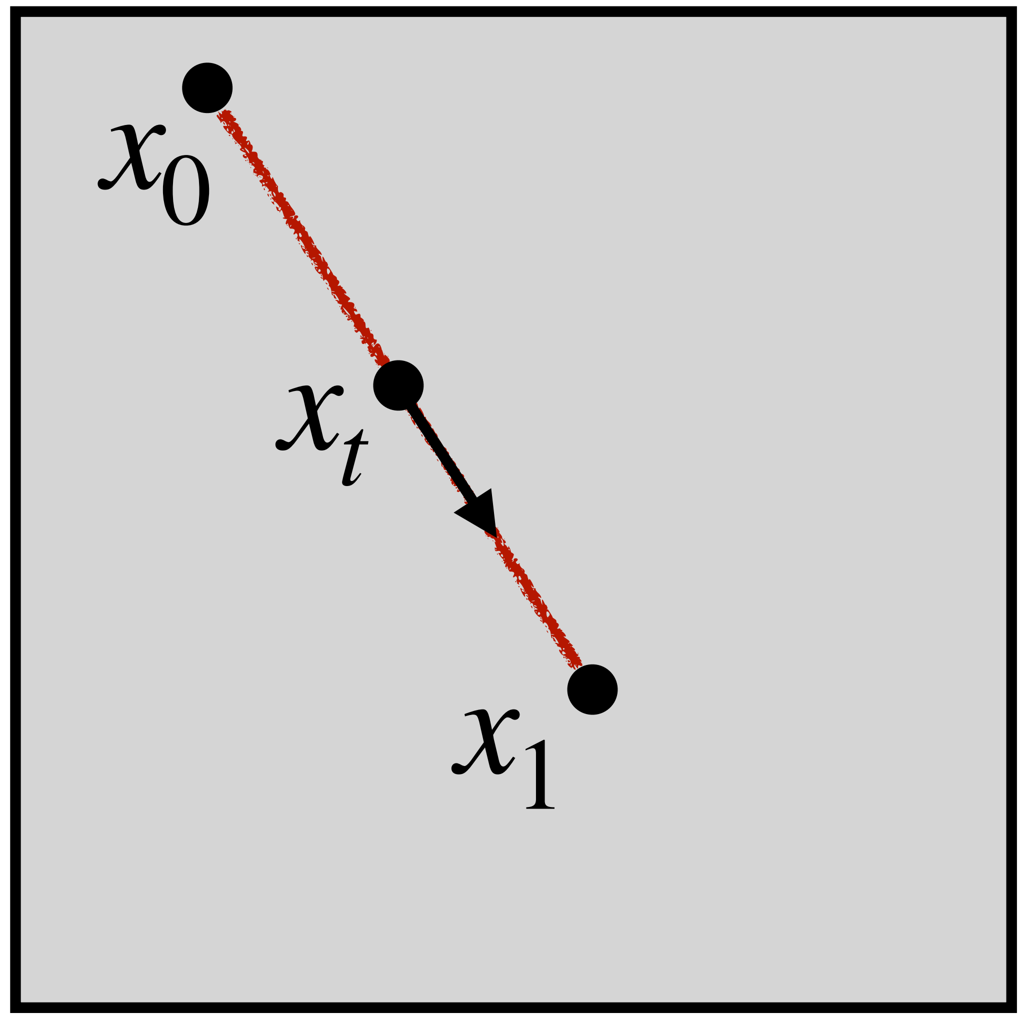
How do you define $z, q(z), x_t, u_t(x_t \mid z)$?

$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := (1-t)x_0 + tx_1$

$u_t(x_t \mid z) := x_1 - x_0$



$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := \exp_{x_0}(t \log_{x_0} x_1)$

$u_t(x_t \mid z) := (\log_{x_t} x_1)/(1-t)$

Euclidean to Riemannian Flow Matching

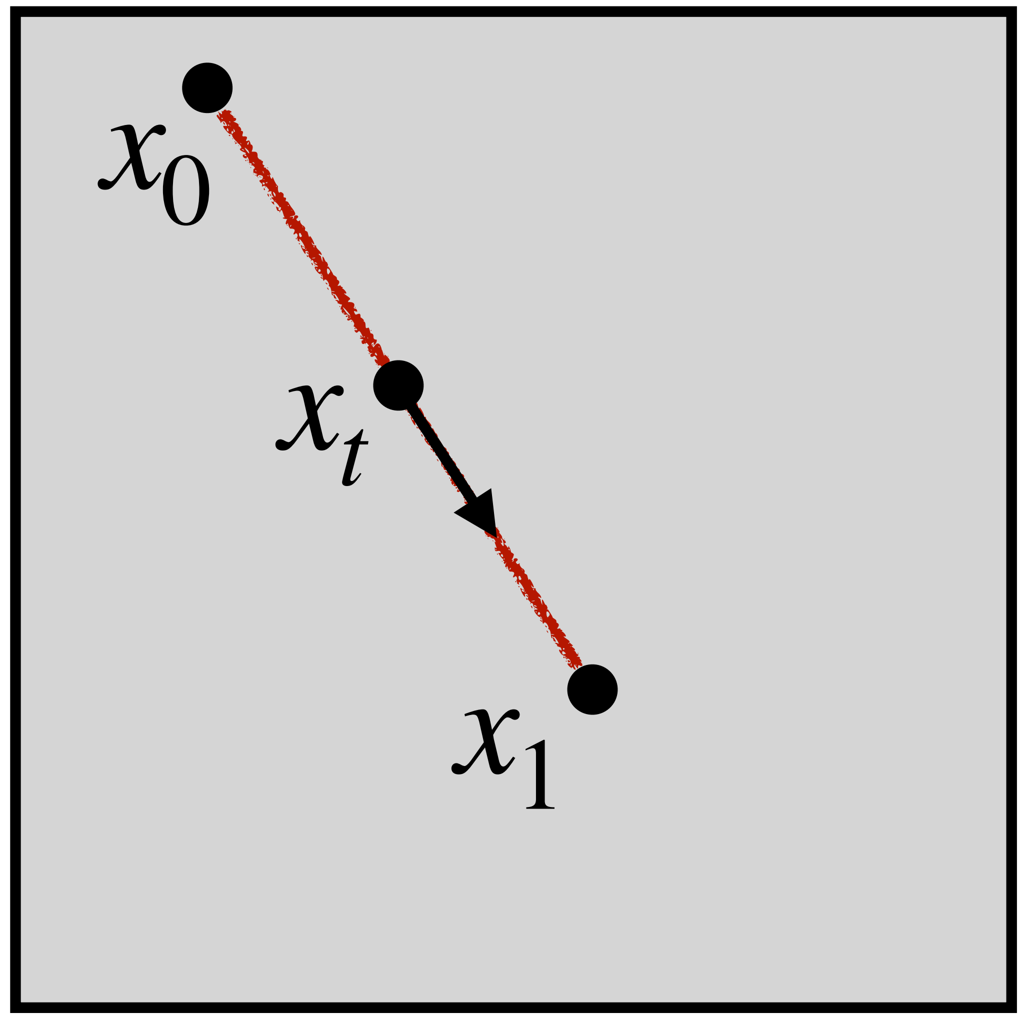
How do you define $z, q(z), x_t, u_t(x_t \mid z)$?

$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := (1 - t)x_0 + tx_1$

$u_t(x_t \mid z) := x_1 - x_0$

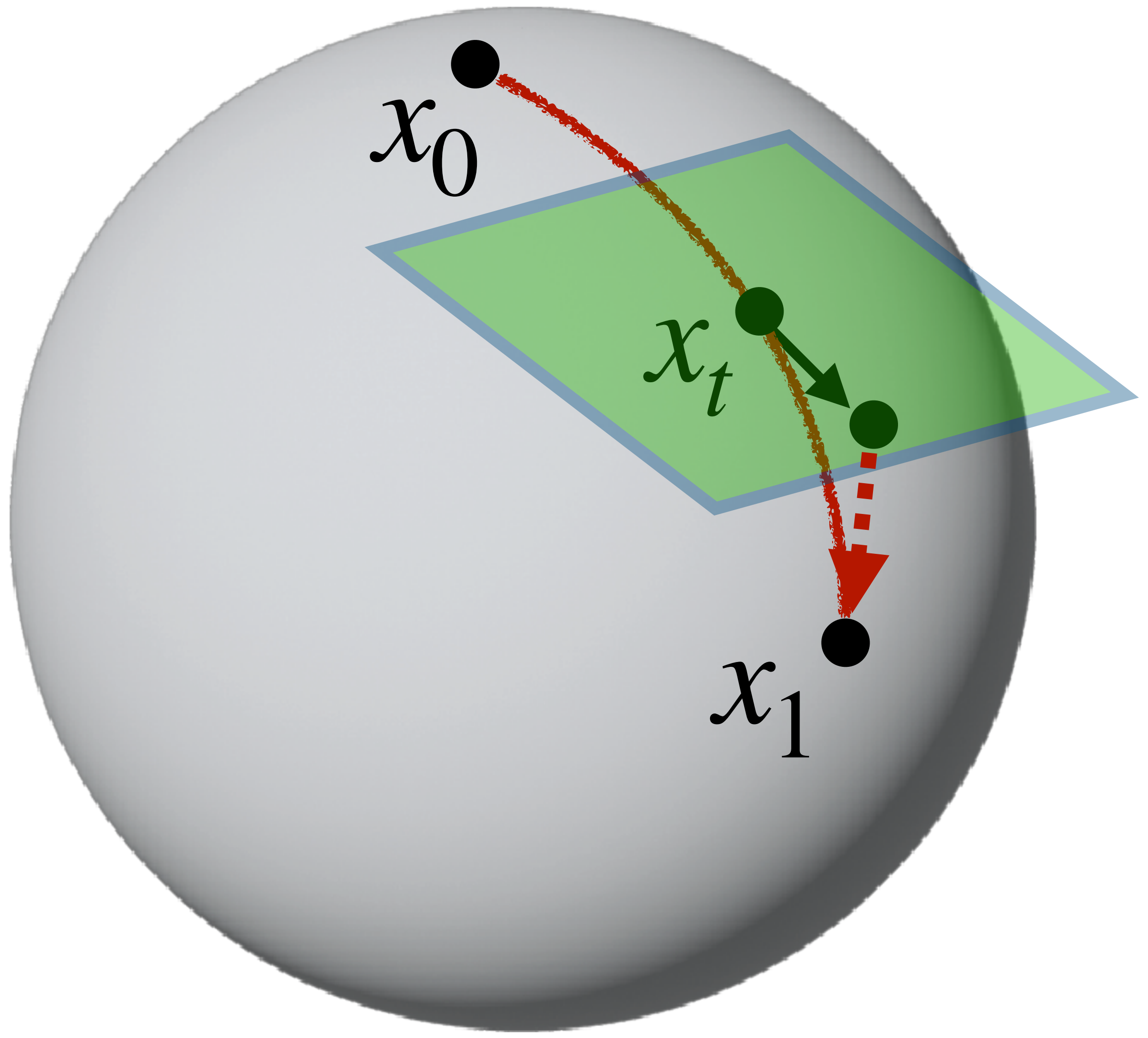


$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := \exp_{x_0}(t \log_{x_0} x_1)$

$u_t(x_t \mid z) := (\log_{x_t} x_1)/(1 - t)$



Euclidean to Riemannian Flow Matching

How do you define $z, q(z), x_t, u_t(x_t \mid z)$?

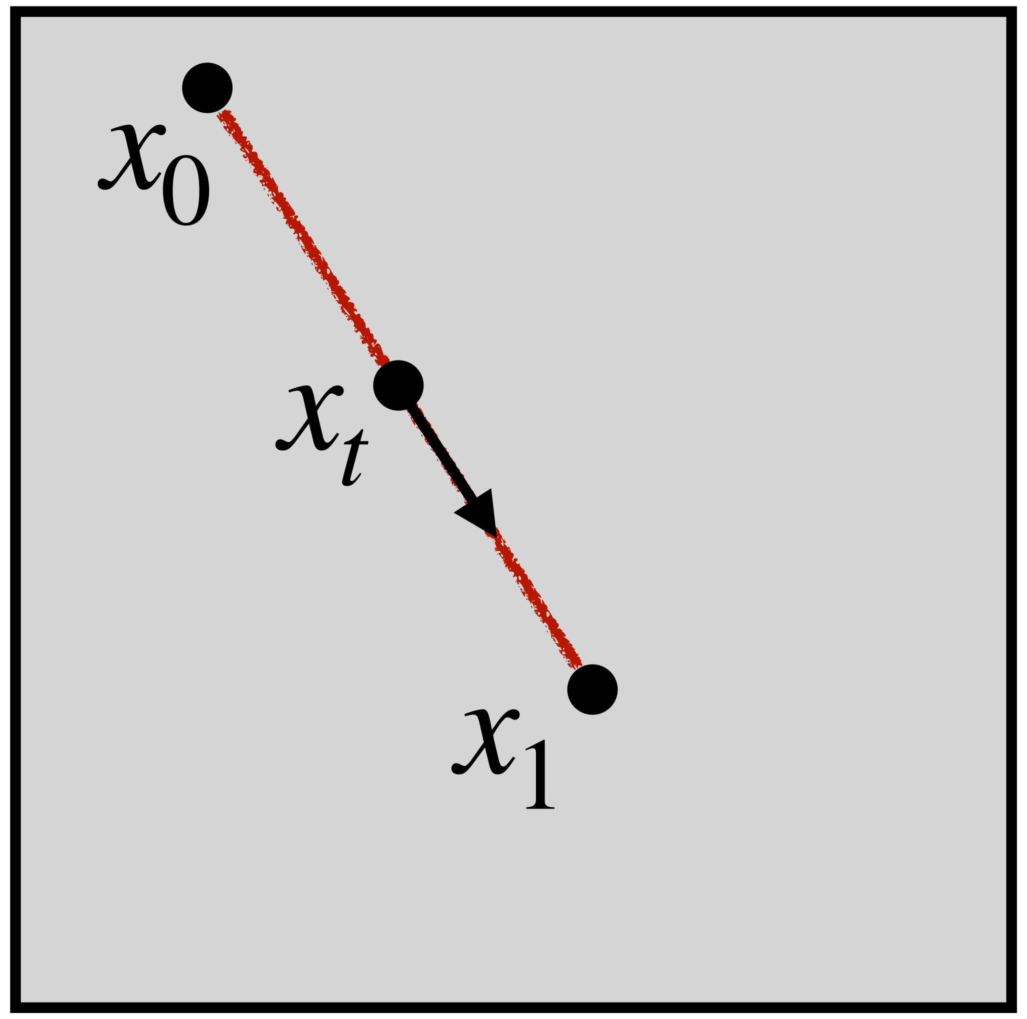
$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

Reminder in Euclidean!
$\exp_a b = a + b$
$\log_a b = b - a$

$x_t := (1 - t)x_0 + tx_1 \quad = x_0 + t(x_1 - x_0)$

$u_t(x_t \mid z) := x_1 - x_0 \quad = (x_1 - x_t)/(1 - t)$

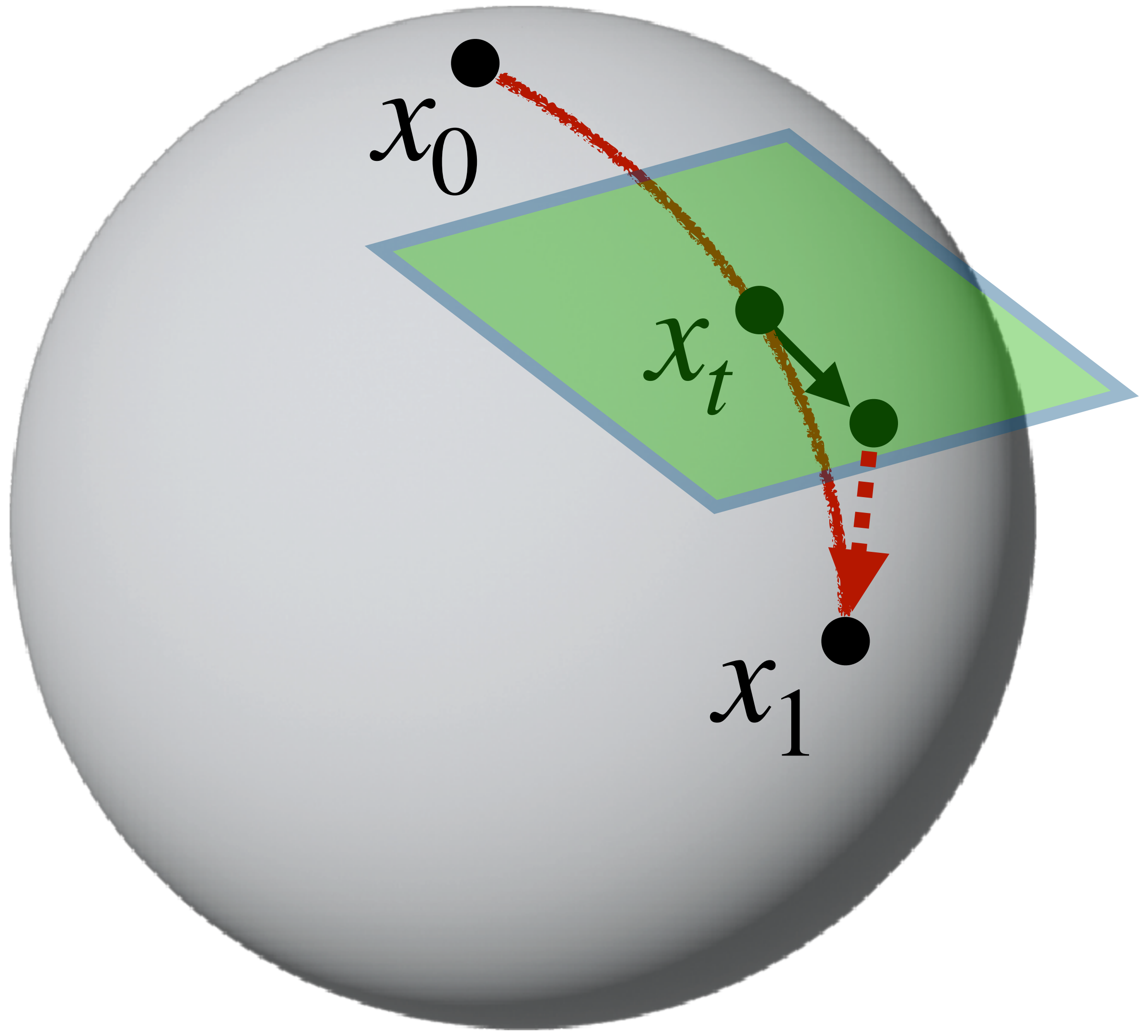


$z := (x_0, x_1)$

$q(z) := q(x_0)q(x_1)$

$x_t := \exp_{x_0}(t \log_{x_0} x_1)$

$u_t(x_t \mid z) := (\log_{x_t} x_1)/(1 - t)$



Euclidean to Riemannian Flow Matching

How do you define $z, q(z), x_t, u_t(x_t \mid z)$?

* Requires efficient $\exp$ and $\log$ map

$$z := (x_0, x_1)$$

$$q(z) := q(x_0)q(x_1)$$

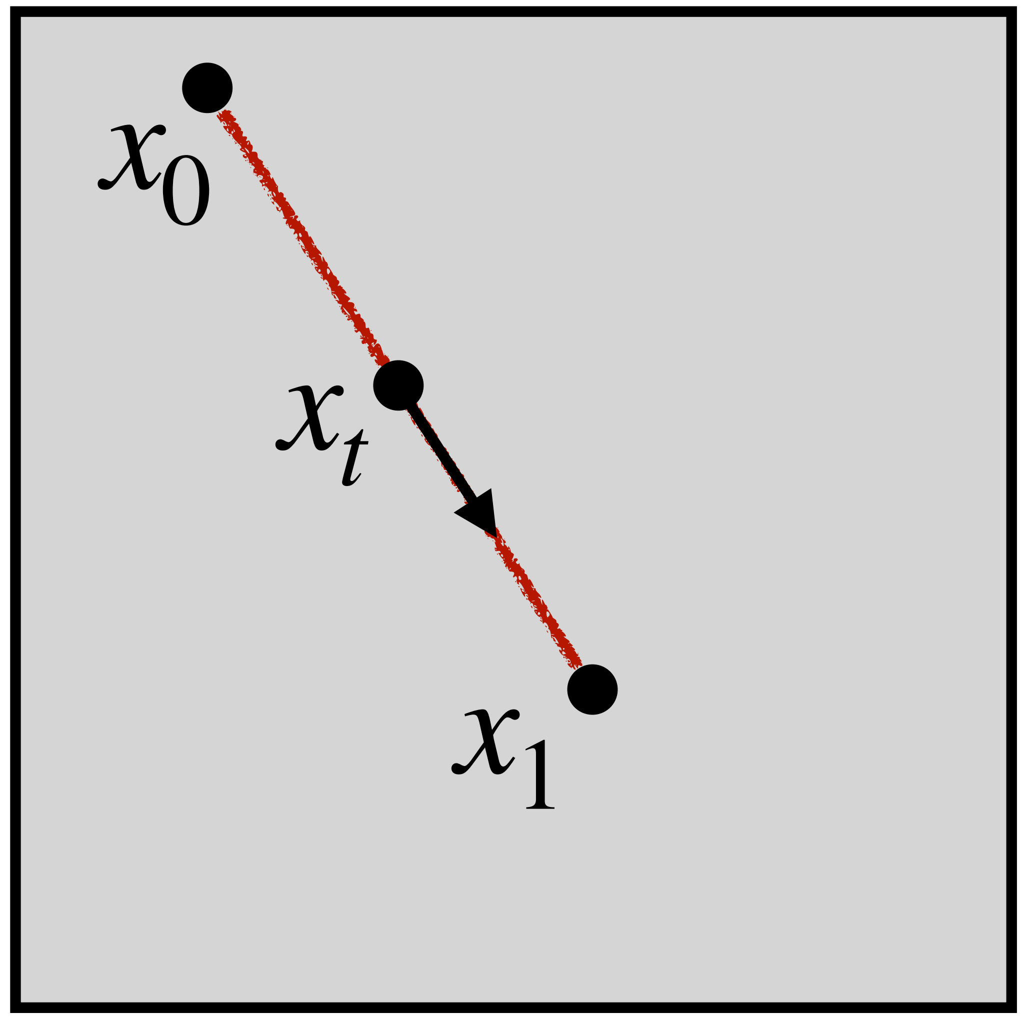
Reminder in Euclidean!

$\exp_a b = a + b$

$\log_a b = b - a$

$$x_t := (1 - t)x_0 + tx_1 \quad = x_0 + t(x_1 - x_0)$$

$$u_t(x_t \mid z) := x_1 - x_0 \quad = (x_1 - x_t)/(1 - t)$$

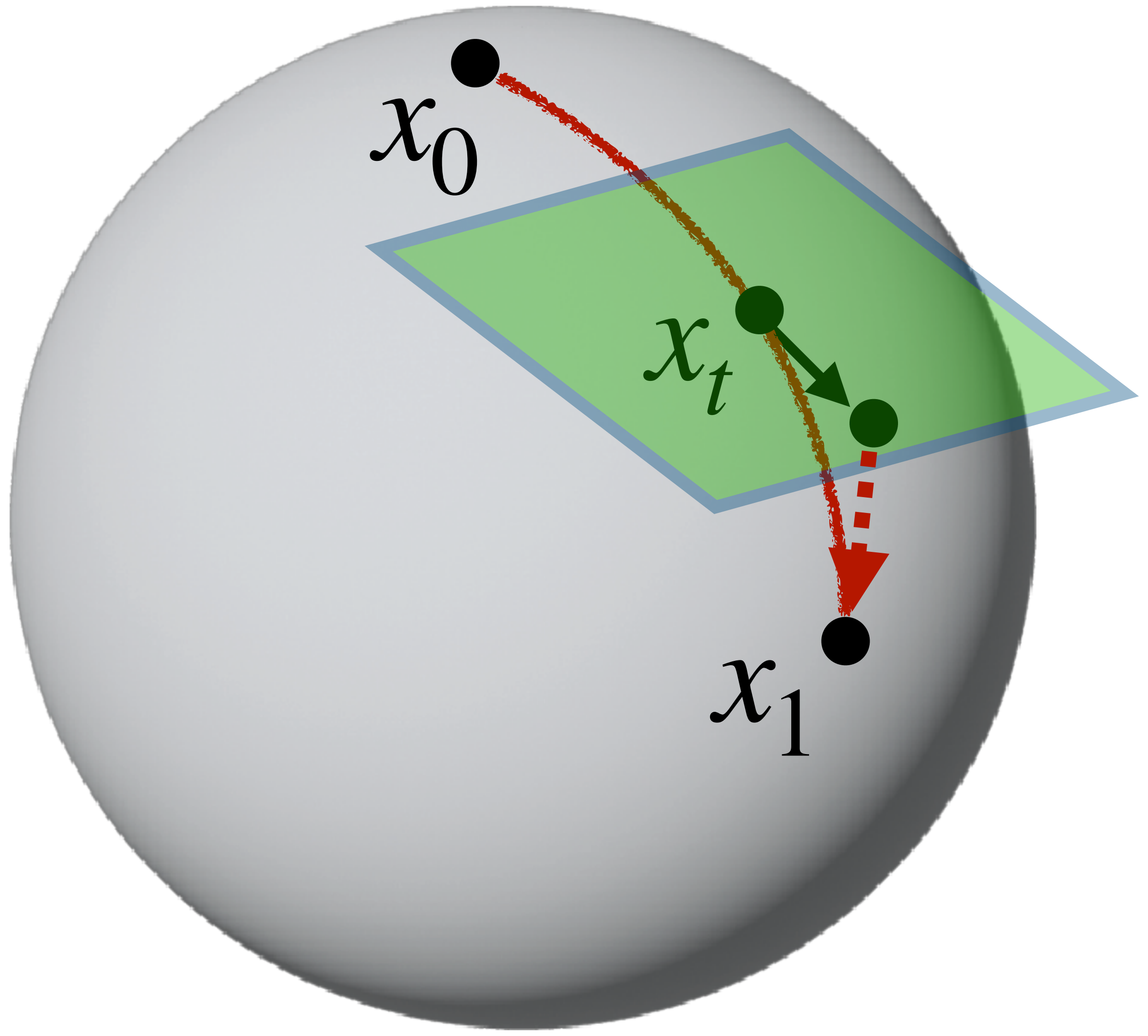


$$z := (x_0, x_1)$$

$$q(z) := q(x_0)q(x_1)$$

$$x_t := \exp_{x_0}(t \log_{x_0} x_1)$$

$$u_t(x_t \mid z) := (\log_{x_t} x_1)/(1 - t)$$



Likelihood computation

Log-likelihood computation

$$\log p_1(x_1) = \log p(x_0) + \int_1^0 \mathrm{div}_g(u_t(x_t))dt$$

$$x_t = x_1 + \int_1^t u_s(x_s)ds$$

Riemannian Divergence

$$\mathrm{div}_g(X) = \nabla \circ X = \frac{1}{\sqrt{\det g}} \sum_i^n \frac{\partial}{\partial x_i}(\sqrt{\det g} X^i)$$

Allows calculation of the log likelihood by integrating divergence over time!

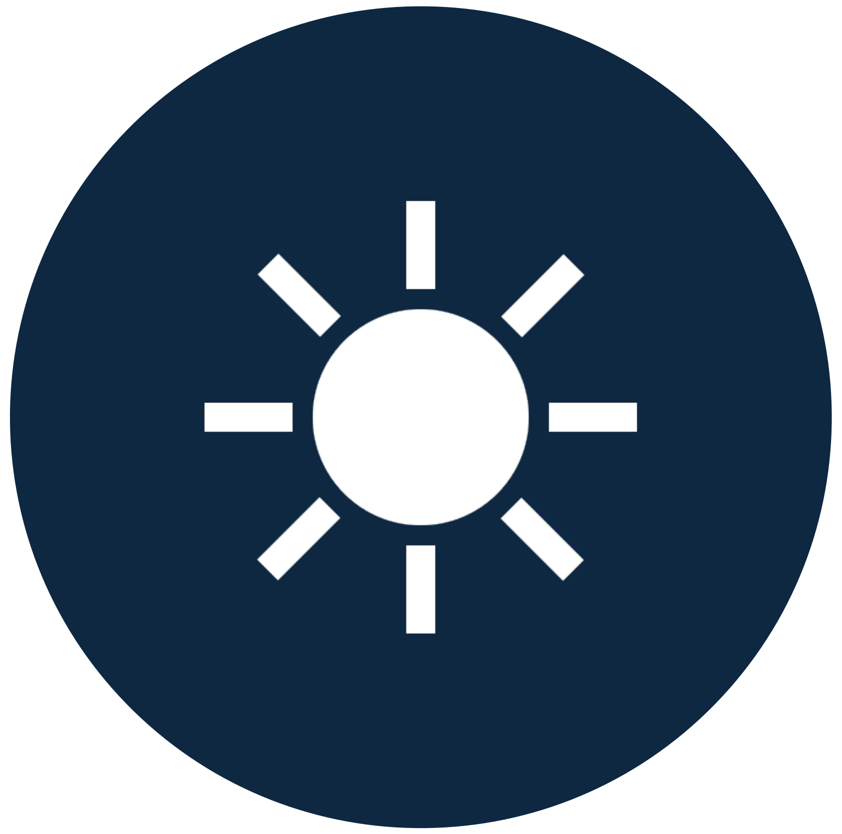
A Case Study:
The protein design problem

Given a set of desired properties, create a protein sequence that satisfies those properties.

Why Design Proteins?

MEDICINE

VACCINES



CLIMATE

What might you want to design for?

- Structure
- Binding / Interaction affinity
  - Strength
  - Specificity
- Stability
- Flexibility
- Evading the immune system
- Activating the immune system

Property —> Structure + Sequence

What are proteins and why do we care?

- Molecules found in all life
- Human DNA contains code for ~20k unique types of proteins
- All stem from the same 20 “amino acid” building blocks
- Sequence of amino acids determine the 3D structure and therefore function of the protein

Sequence:
MVKSYELIAGWFTPHQMVKS

Structure:

Protein Representations

- Can be represented as a cloud of atoms in $\mathbb{R}^{(N\times\sim19.2)\times3}$
- Backbone is represented as elements of $\mathrm{SE}(3)^N$ (local orientations around C_α atoms)
- Sidechains are represented as elements of $\mathrm{SO}(2)^{N\times7}$ (torsion angles)

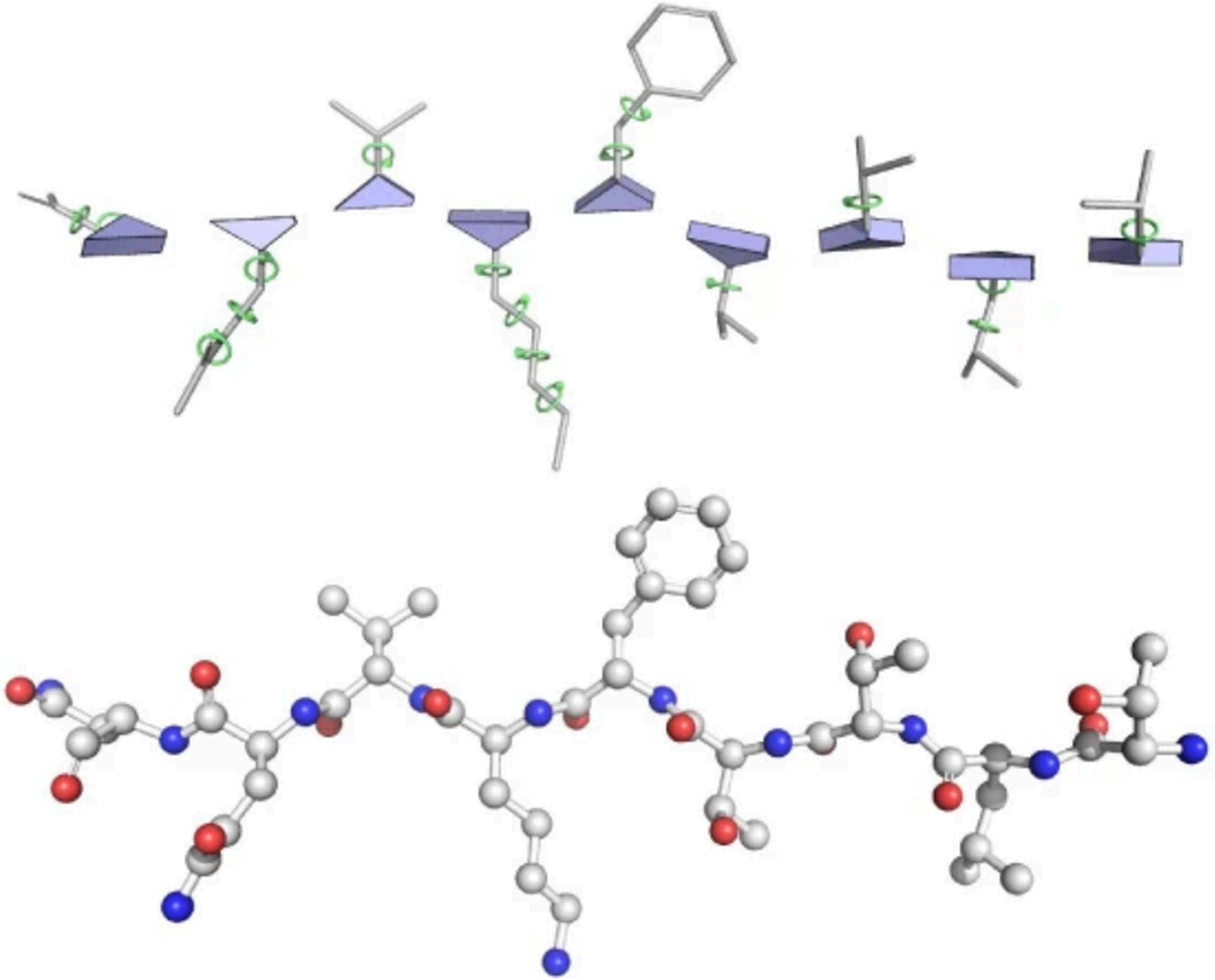
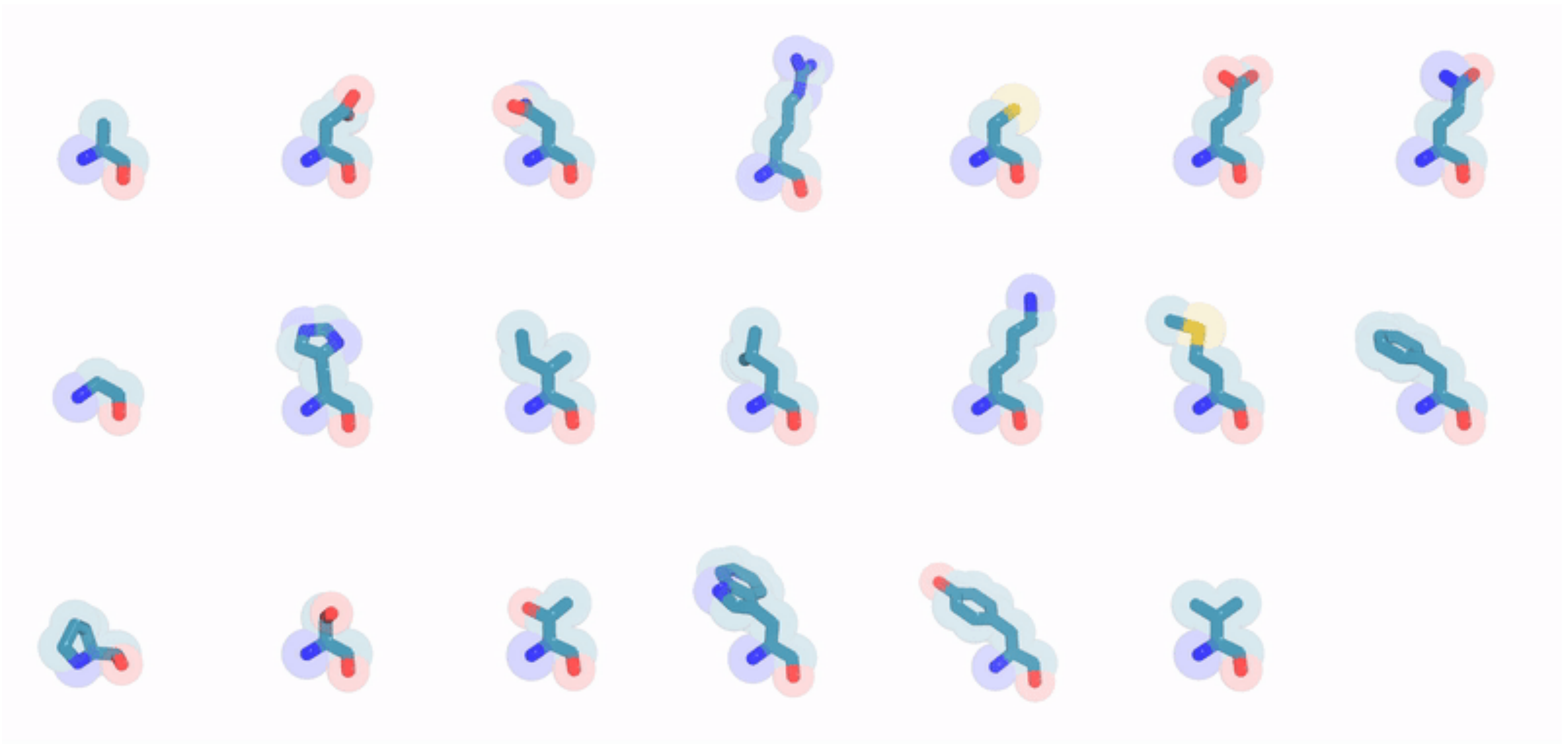


Image credit: https://www.ipd.uw.edu/what-is-protein-design/

Protein Representations

- Can be represented as a cloud of atoms in $\mathbb{R}^{(N\times\sim 19.2)\times 3}$
- Backbone is represented as elements of $\mathrm{SE}(3)^N$ (local orientations around C_α atoms)
- Sidechains are represented as elements of $\mathrm{SO}(2)^{N\times 7}$ (torsion angles)

20 amino acids



Backbone structure

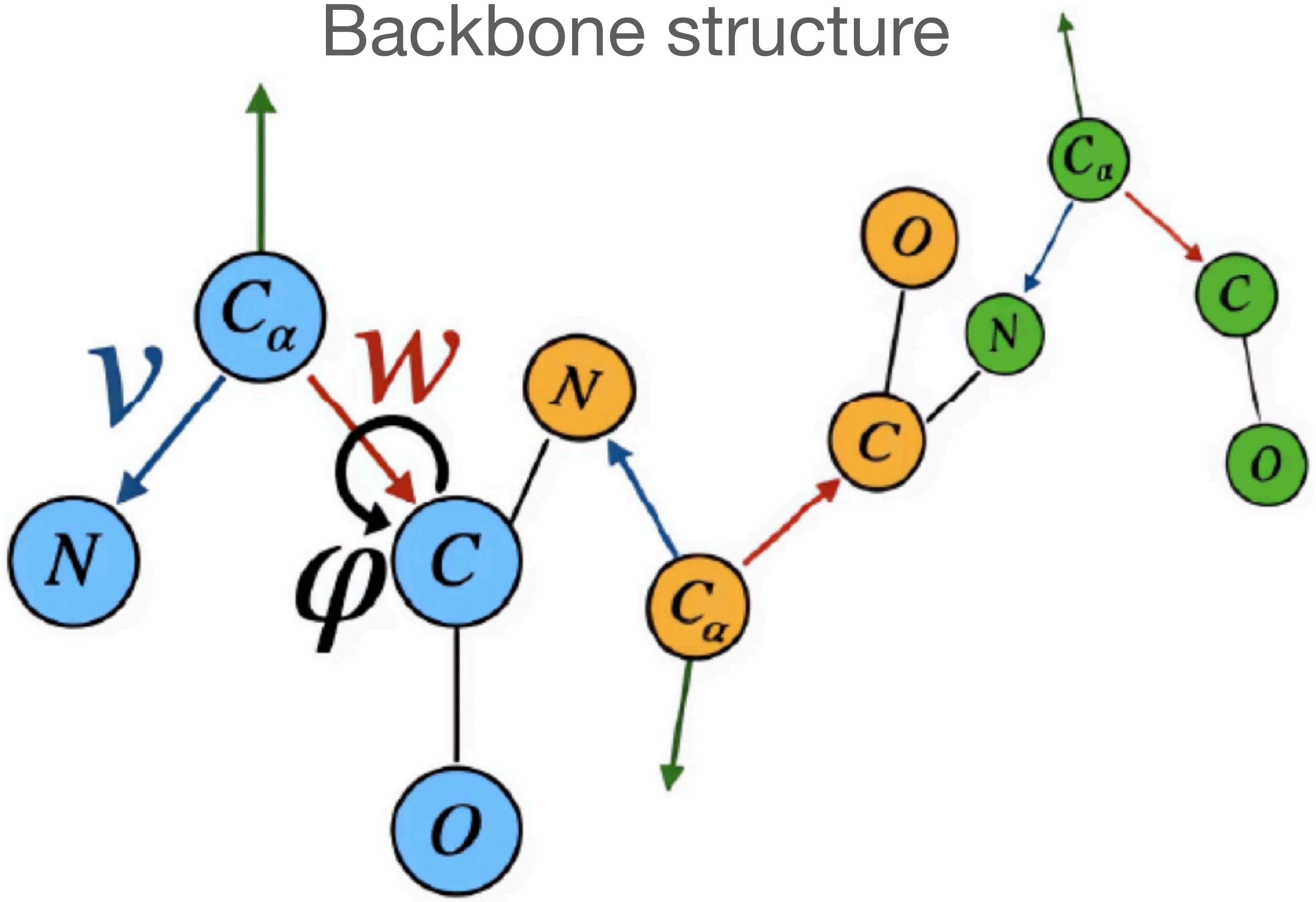


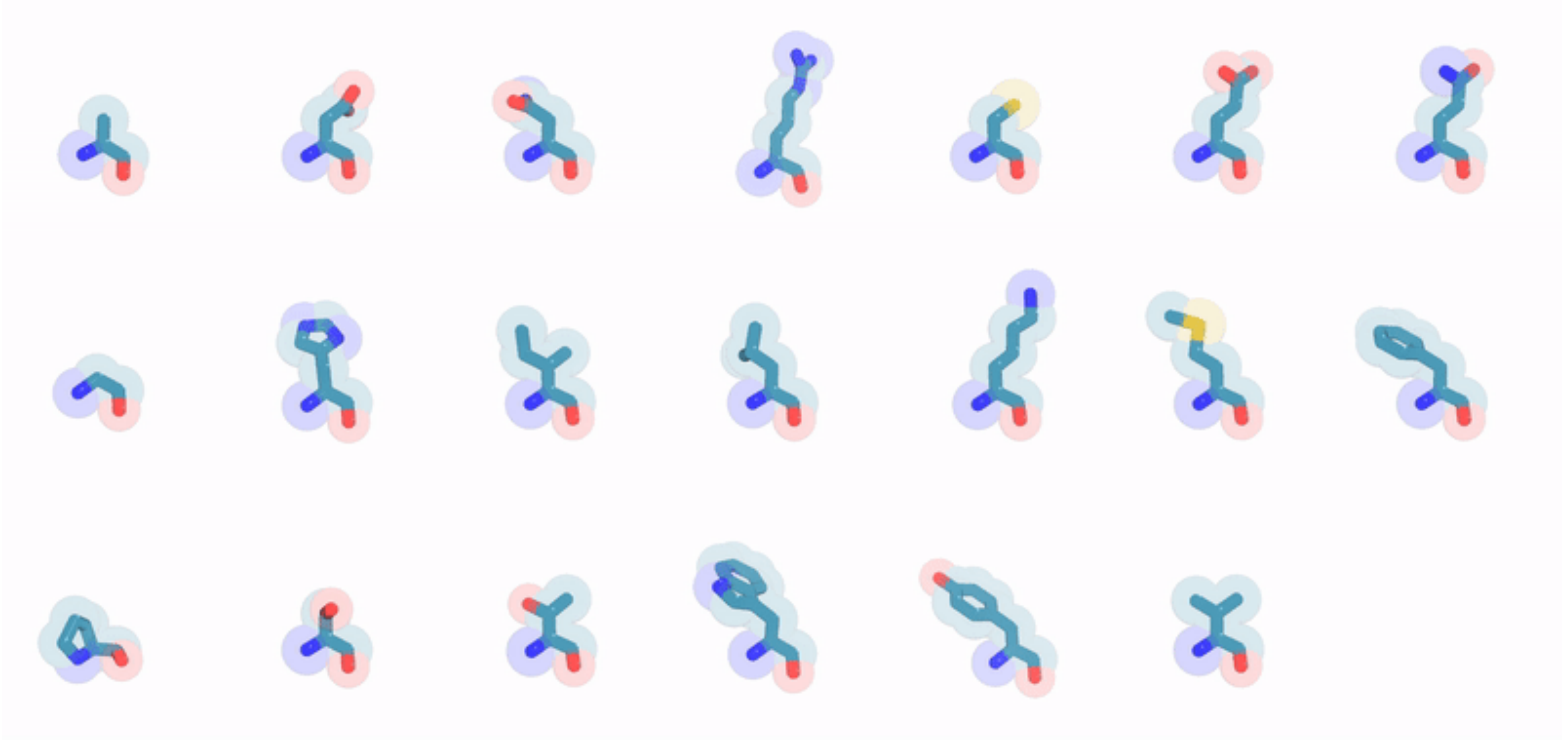
Image credit: https://www.ipd.uw.edu/what-is-protein-design/

Protein Representations

- Can be represented as a cloud of atoms in $\mathbb{R}^{(N\times\sim19.2)\times3}$
- Backbone is represented as elements of $\mathrm{SE}(3)^N$ (local orientations around C_α atoms)
- Sidechains are represented as elements of $\mathrm{SO}(2)^{N\times7}$ (torsion angles)

But Why?

20 amino acids



Backbone structure

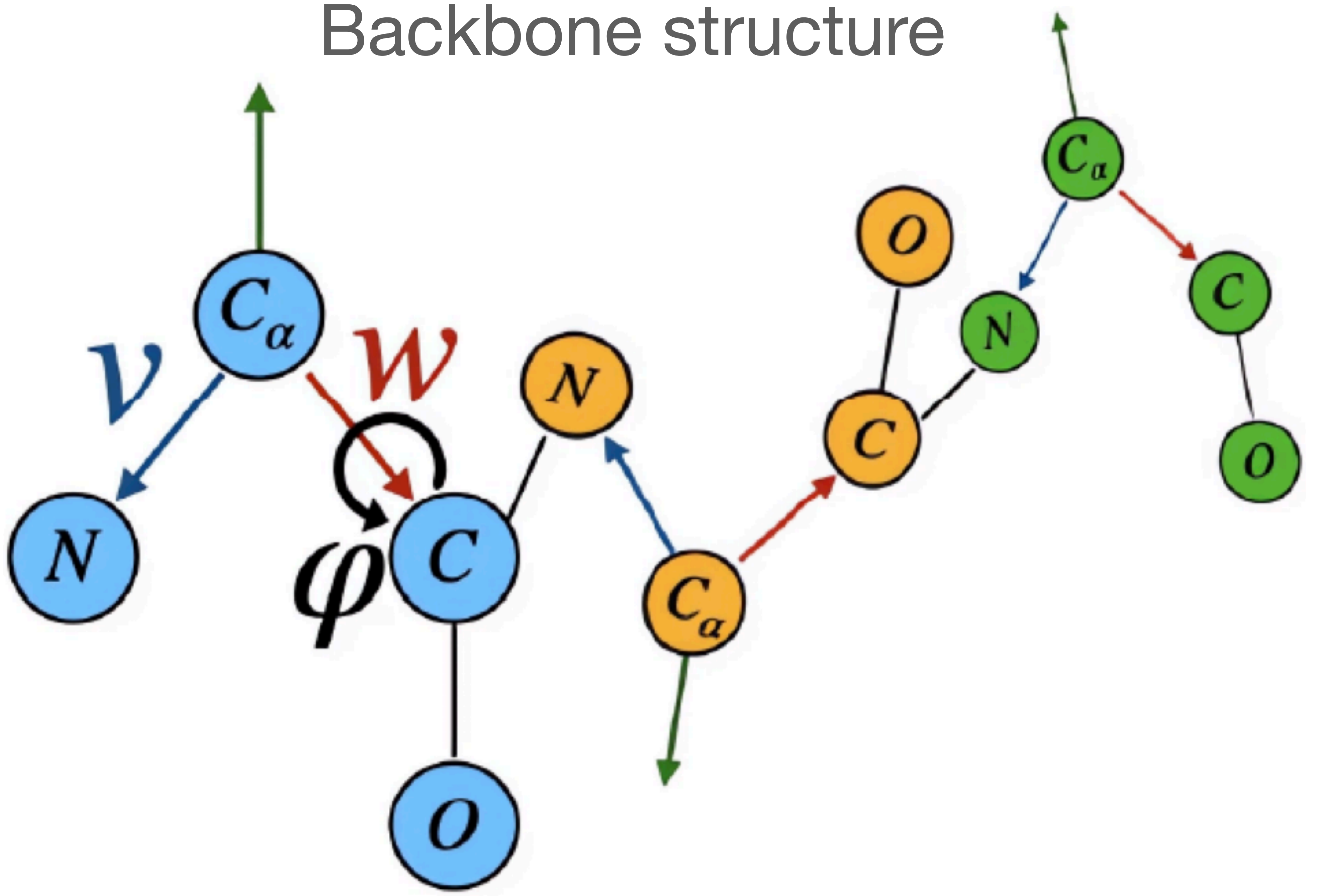


Image credit: https://www.ipd.uw.edu/what-is-protein-design/

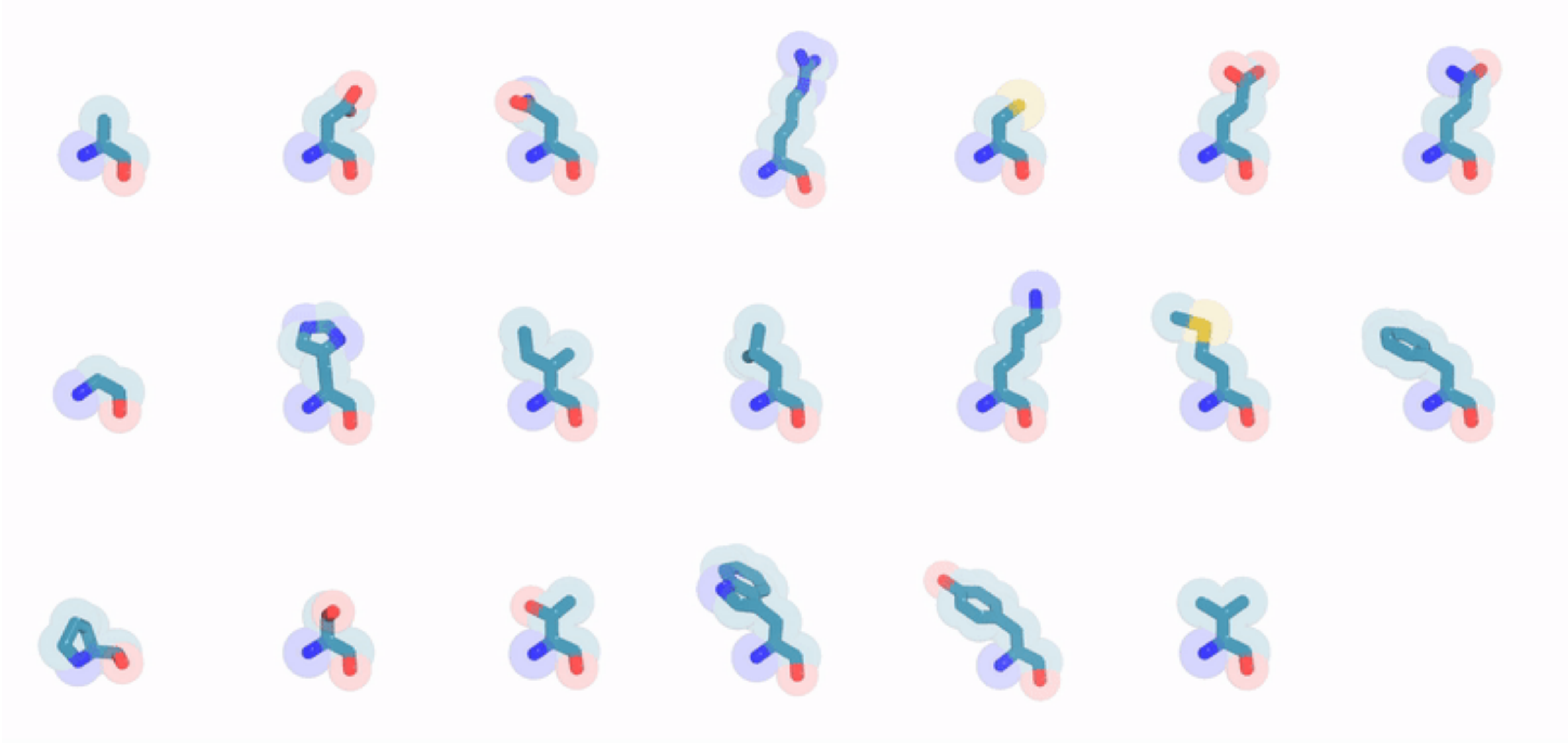
Protein Representations

- Can be represented as a cloud of atoms in $\mathbb{R}^{(N\times\sim 19.2)\times 3}$
- Backbone is represented as elements of $\mathrm{SE}(3)^N$ (local orientations around C_α atoms)
- Sidechains are represented as elements of $\mathrm{SO}(2)^{N\times 7}$ (torsion angles)

But Why?

Leverages prior knowledge on bond lengths and amino acid structure (order of atom types)

20 amino acids



Backbone structure

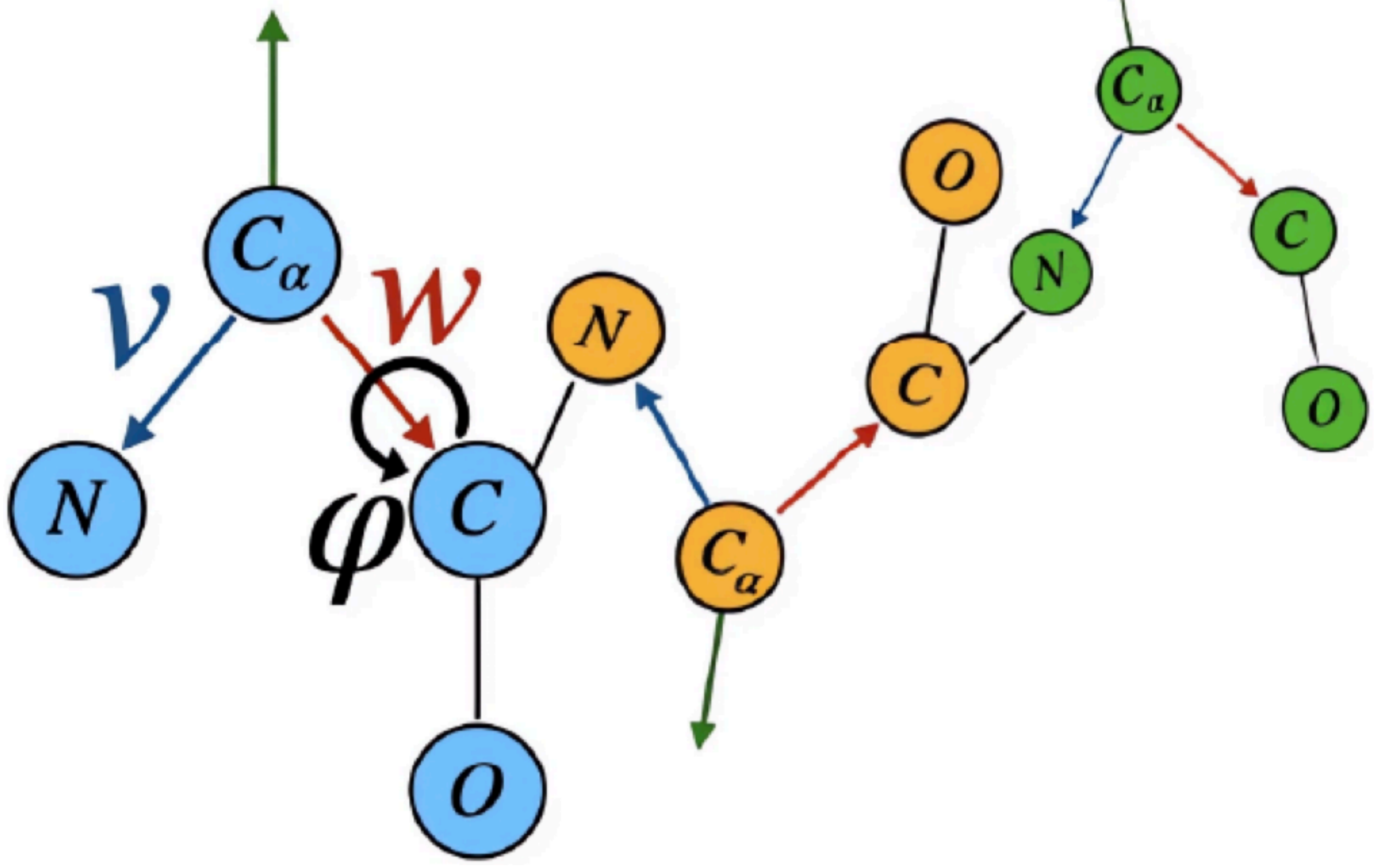
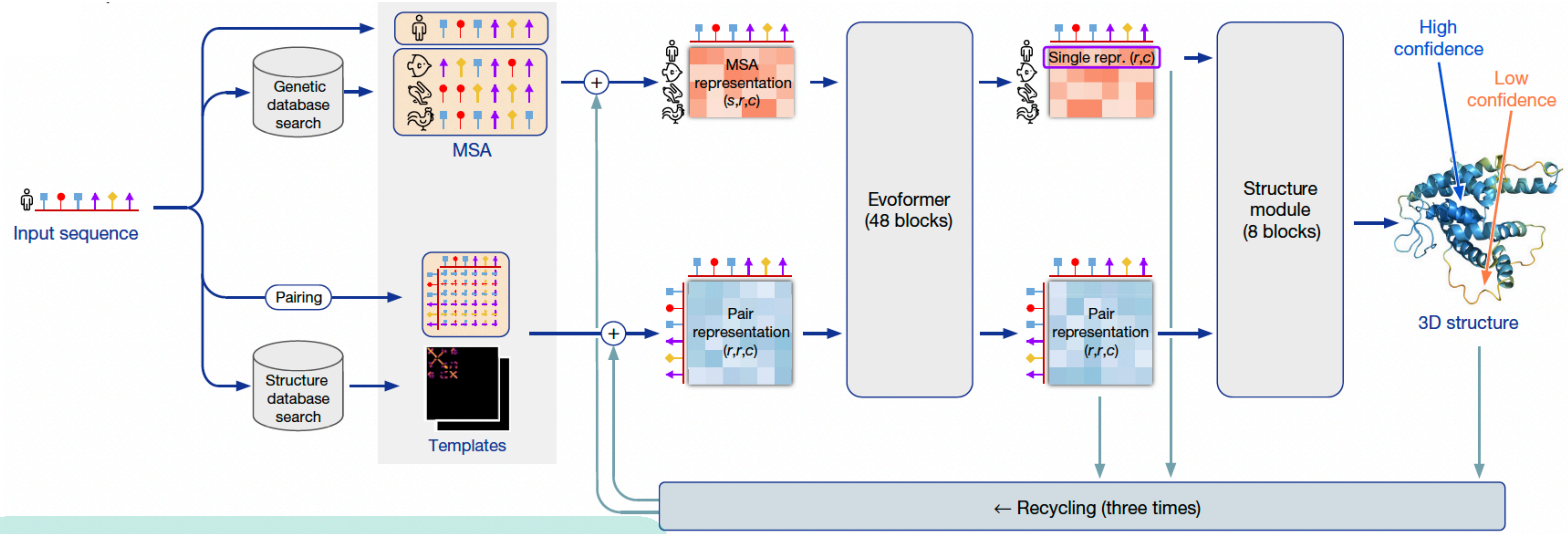


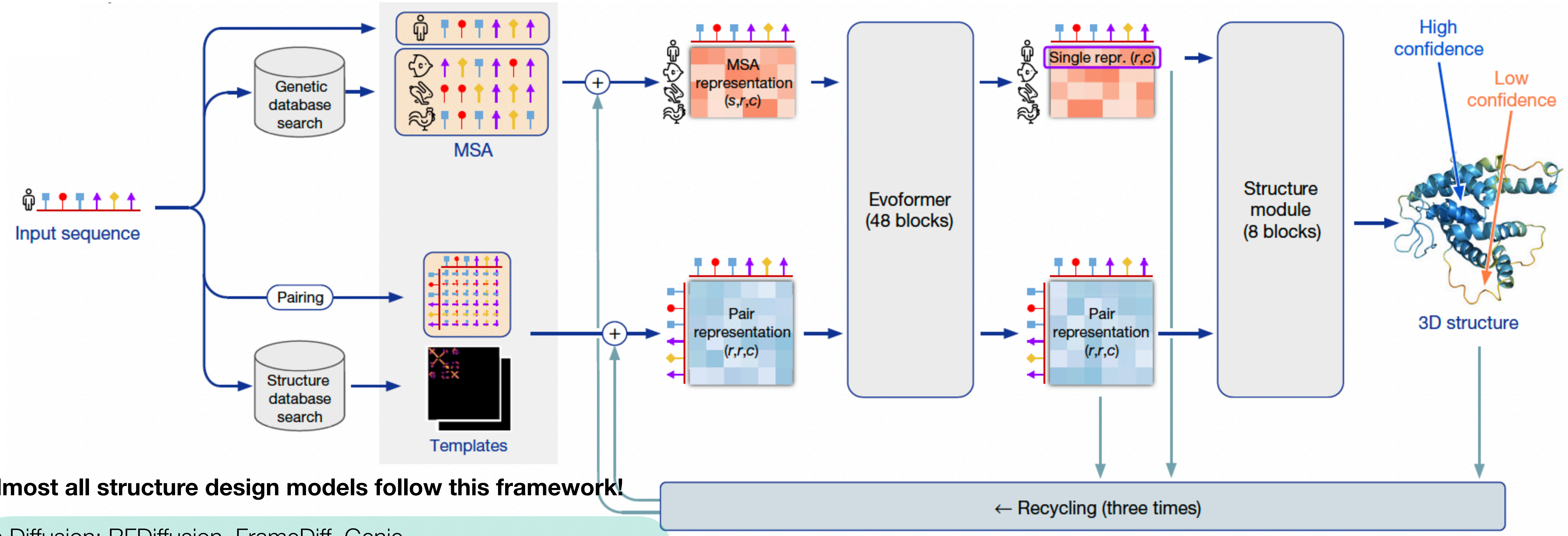
Image credit: https://www.ipd.uw.edu/what-is-protein-design/

AlphaFold 2



Input sequence
Genetic database search
MSA
Pairing
Structure database search
Templates
MSA representation (s,r,c)
Pair representation (r,r,c)
Evoformer (48 blocks)
Single repr. (r,c)
Pair representation (r,r,c)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)

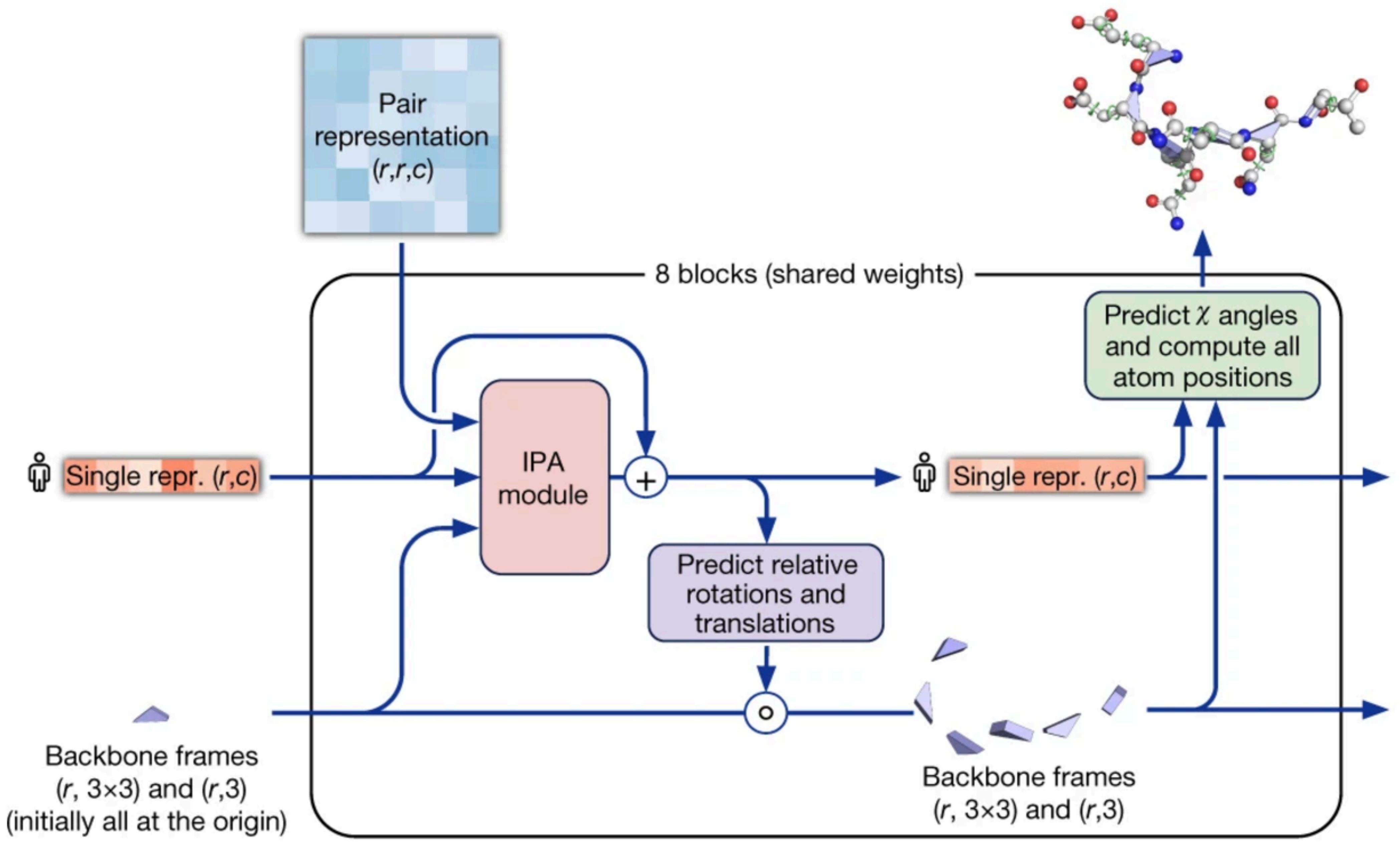
AlphaFold 2



Almost all structure design models follow this framework!

- Diffusion: RFDiffusion, FrameDiff, Genie
- Flow-based: FoldFlow, FrameFlow
- Improved flow-based: Proteus, MultiFlow, PepFlow, and PPFlow
- Sequence + structure: Genie 2 and FoldFlow 2

AlphaFold 2 — Structure Module



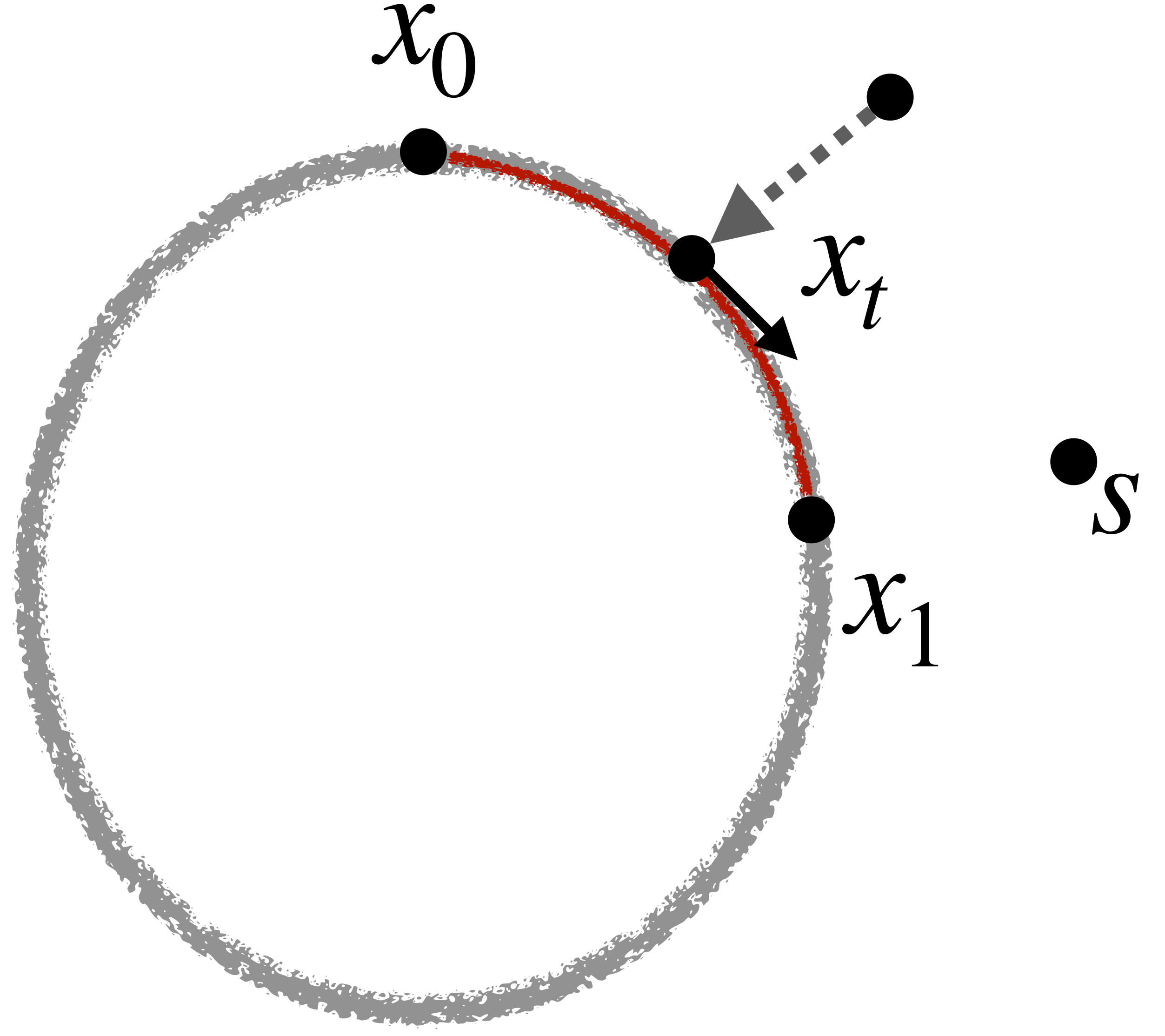

```python
def forward(self, s, z):
    """
    Args:
        "s": [*, N_res, C_s] single representation
        "z": [*, N_res, N_res, C_z] pair representation
    Returns:
        "rigids": [*, N_res, 7] rigid transformation
        "angles": [*, N_res, 7, 2] angles
        "s": [*, N_res, C_s] single representation
    """
    rigids = self.identity(s.shape[:-1])
    for i in range(self.no_blocks):
        s = s + self.ipa(s, z, rigids)
        rigids = rigids.compose_q_update_vec(self.bb_update(s))
        rigids = rigids.stop_rot_gradient()
    angles = self.angle_resnet(s)
    return rigids, angles, s
```

AlphaFold 2 — Structure Module

- "rigids" are elements of $\mathrm{SE}(3)$ and a concatenation of
    - Translation $\mathbb{R}^3$
    - Quaternion $\mathrm{SO}(3)$
- Directly parameterized

```python
def forward(self, s, z):
    """
    Args:
        "s": [*, N_res, C_s] single representation
        "z": [*, N_res, N_res, C_z] pair representation
    Returns:
        "rigids": [*, N_res, 7] rigid transformation
        "angles": [*, N_res, 7, 2] angles
        "s": [*, N_res, C_s] single representation
    """
    rigids = self.identity(s.shape[:-1])
    for i in range(self.no_blocks):
        s = s + self.ipa(s, z, rigids)
        rigids = rigids.compose_q_update_vec(self.bb_update(s))
        rigids = rigids.stop_rot_gradient()
    angles = self.angle_resnet(s)
    return rigids, angles, s
```

AlphaFold 2 — Structure Module



“angles” are elements of $\mathrm{SO}(2)^7$

Projections on the unit circle of $\mathbb{R}^{7\times 2}$

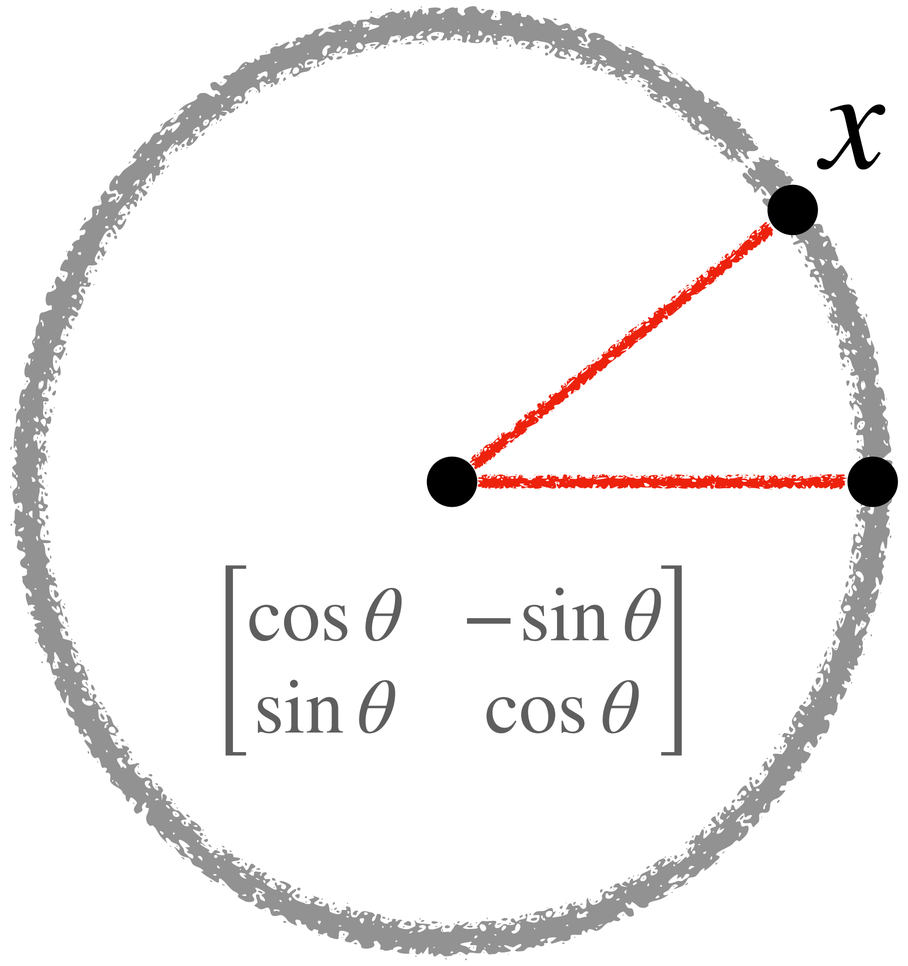
```
def forward(self, s):
    """
    Args:
        s: [*, C_hidden] single embedding
    Returns:
        [*, no_angles, 2] predicted angles
    """
    for l in self.layers:
        s = l(s)
    s = self.linear_out(self.relu(s))
    s = s.view(s.shape[:-1] + (-1, 2))
    norm_denom = torch.norm(s, dim=-1, keepdim=True)
    return s / torch.clamp(norm_denom, min=self.eps)
```

```
def forward(self, s, z):
    """
    Args:
        "s": [*, N_res, C_s] single representation
        "z": [*, N_res, N_res, C_z] pair representation
    Returns:
        "rigids": [*, N_res, 7] rigid transformation
        "angles": [*, N_res, 7, 2] angles
        "s": [*, N_res, C_s] single representation
    """
    rigids = self.identity(s.shape[:-1])
    for i in range(self.no_blocks):
        s = s + self.ipa(s, z, rigids)
        rigids = rigids.compose_q_update_vec(self.bb_update(s))
        rigids = rigids.stop_rot_gradient()
    angles = self.angle_resnet(s)
    return rigids, angles, s
```

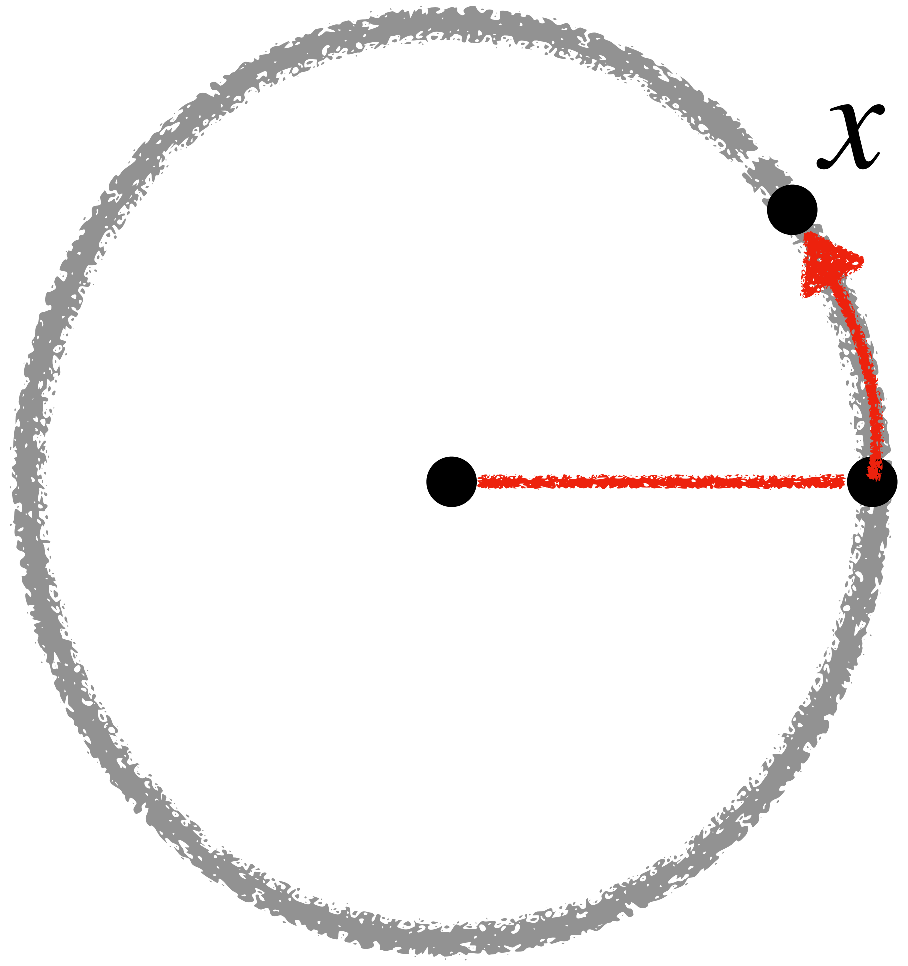
Flow Matching on a Torus

What is the parameterization space?

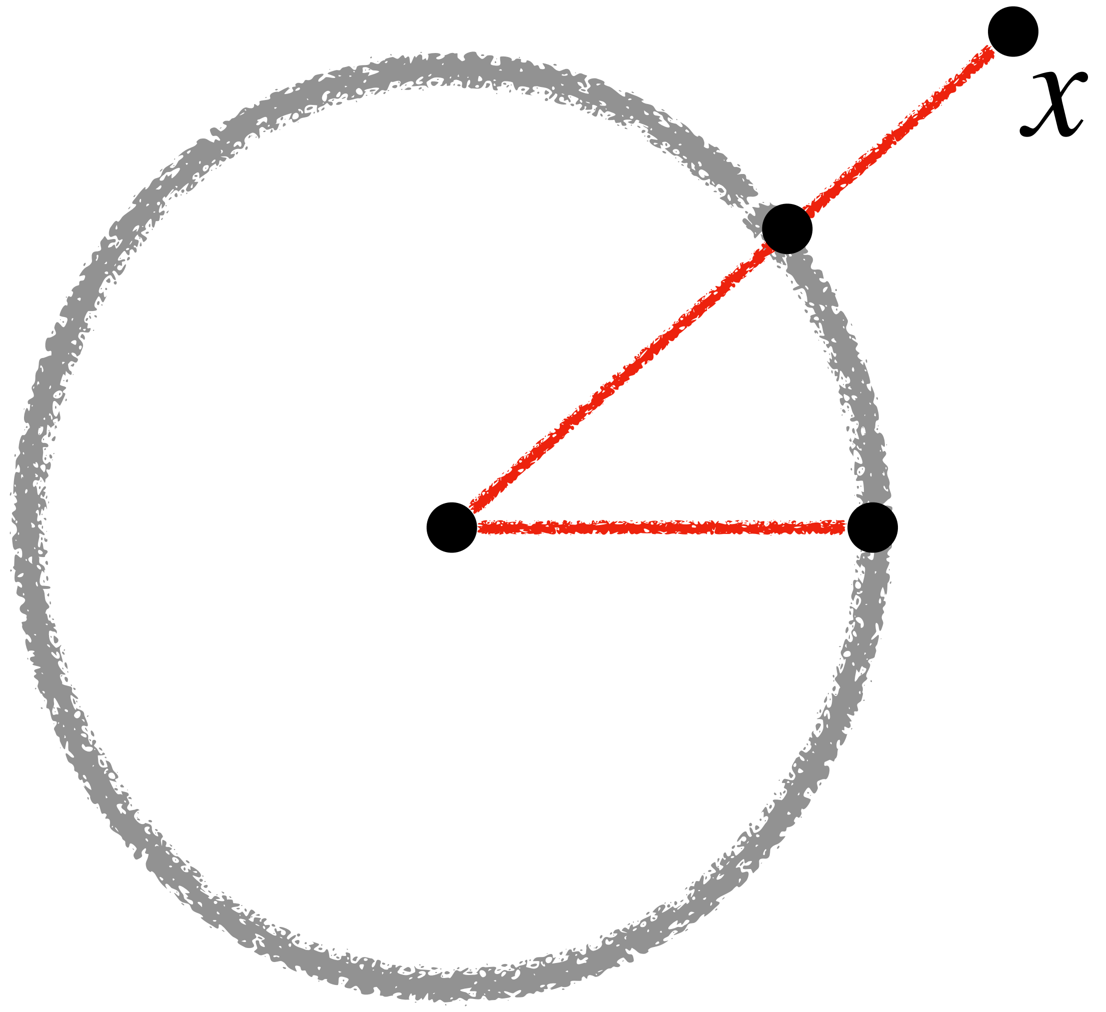
2 x 2 Rotation matrices $\mathbf{SO}(2)$



$[-\pi, \pi]$ interval



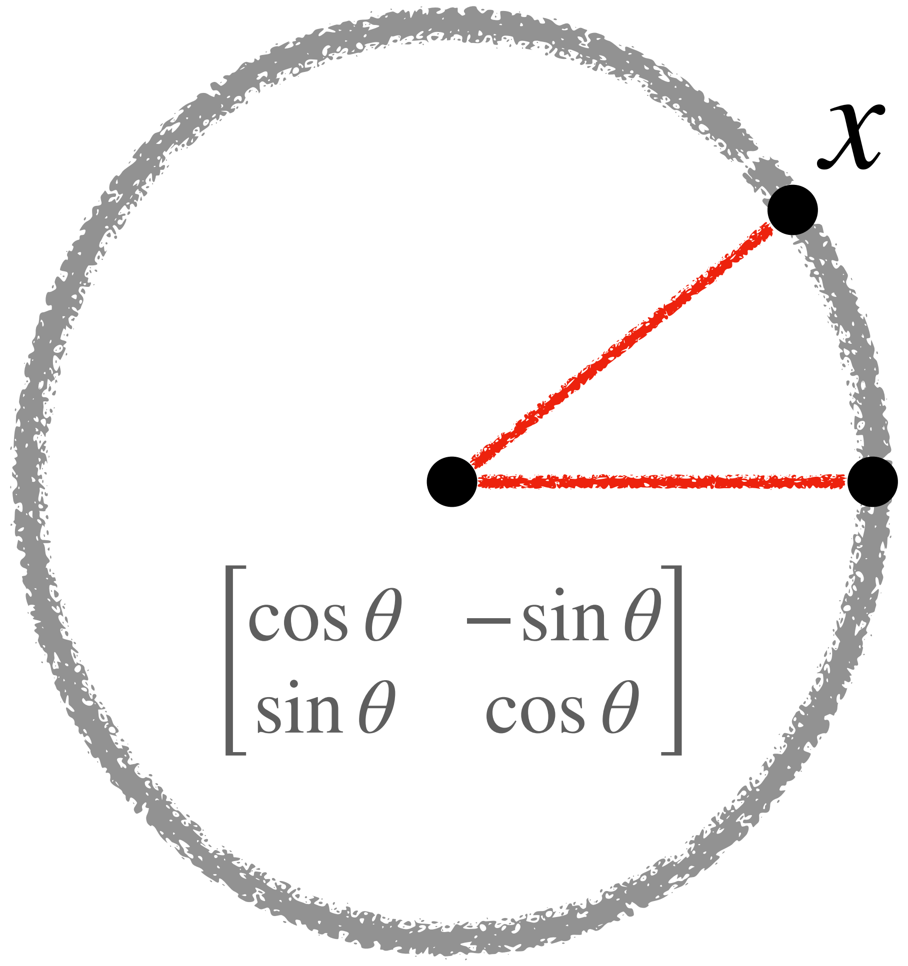
Unit vectors in $\mathbb{R}^2$



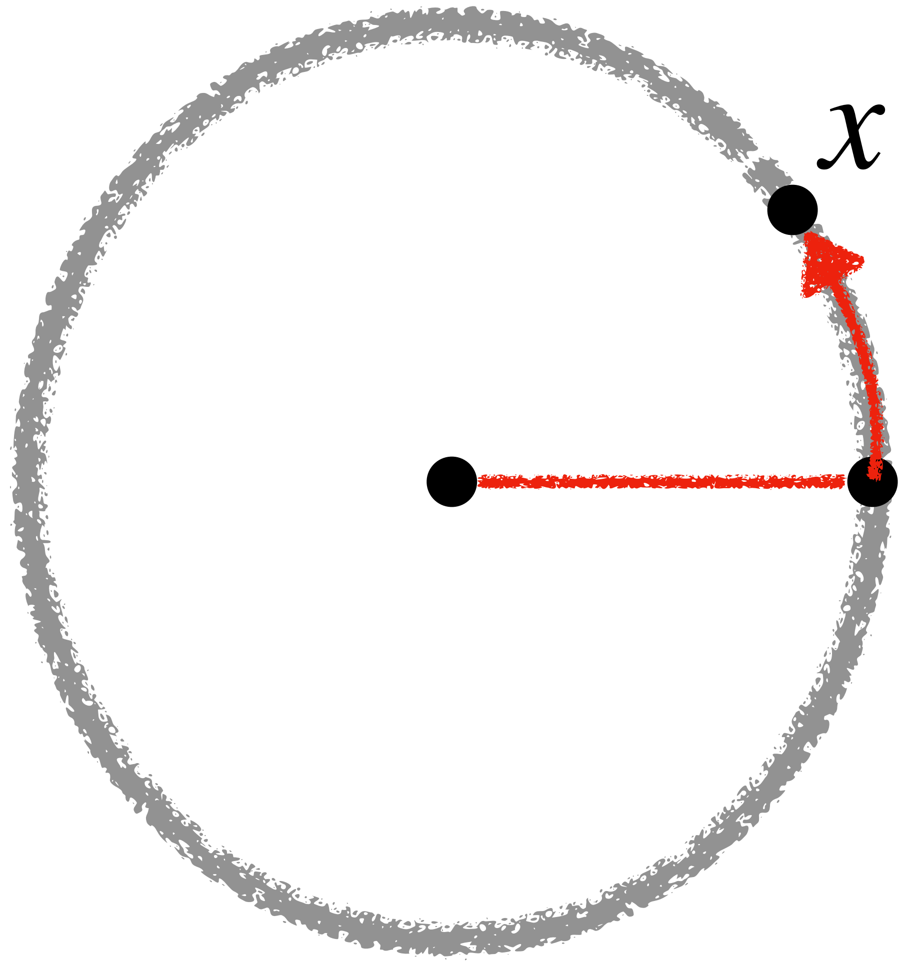
Flow Matching on a Torus

What is the parameterization space?

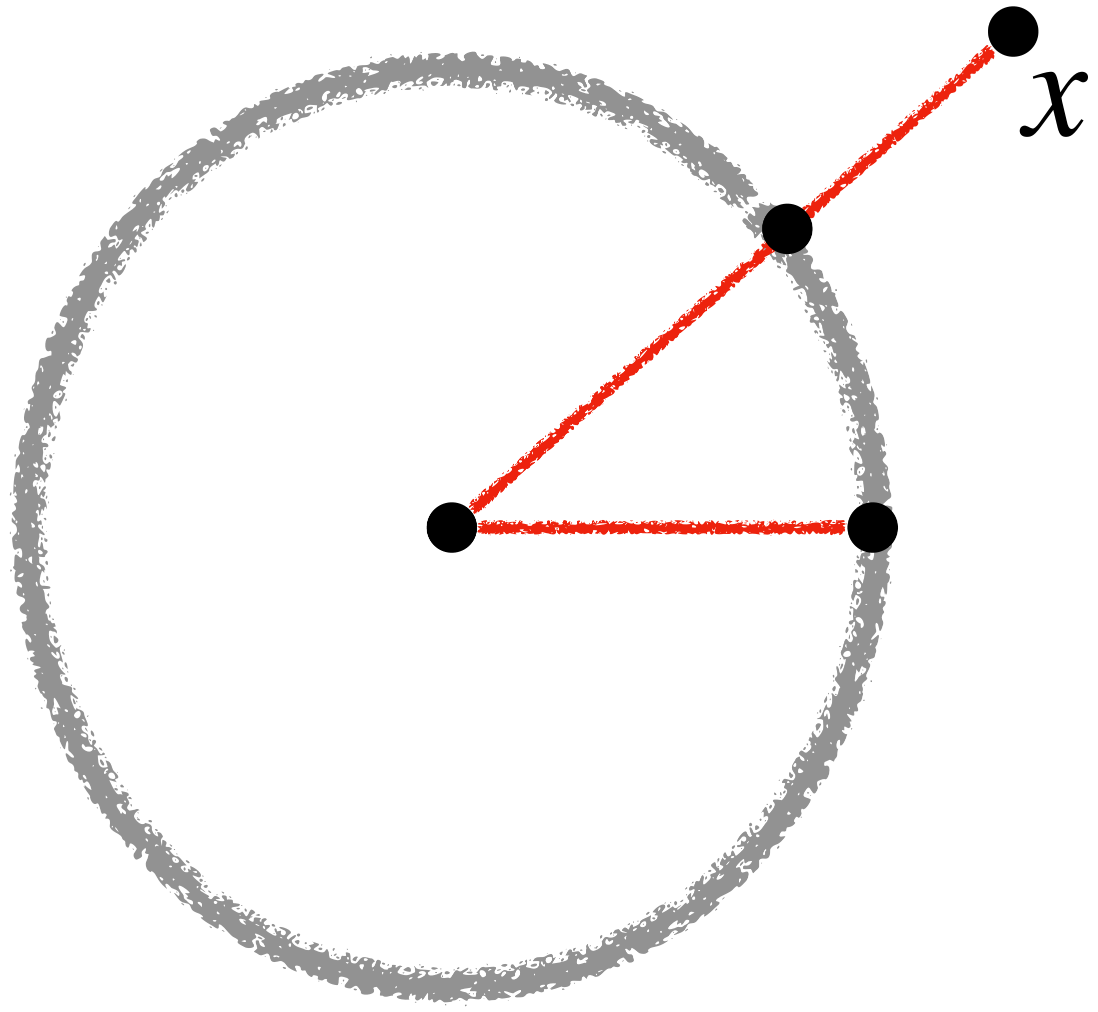
2 x 2 Rotation matrices $\mathbf{SO}(2)$



$[-\pi, \pi]$ interval



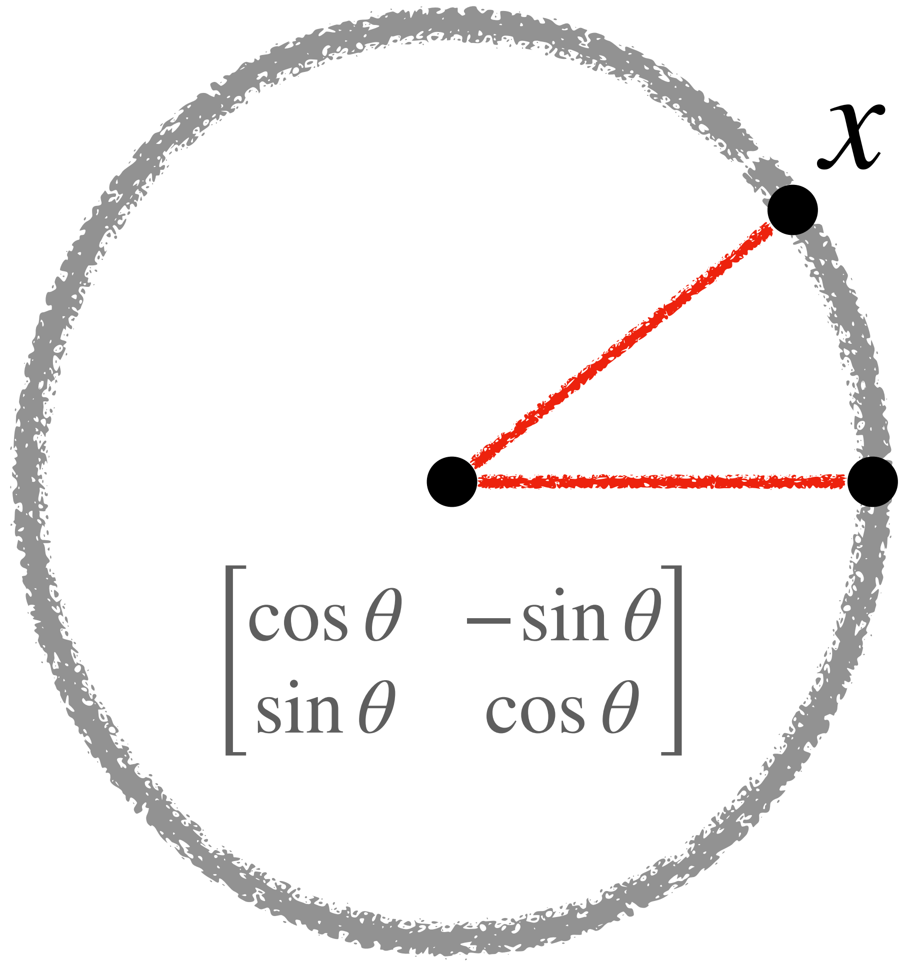
Unit vectors in $\mathbb{R}^2$



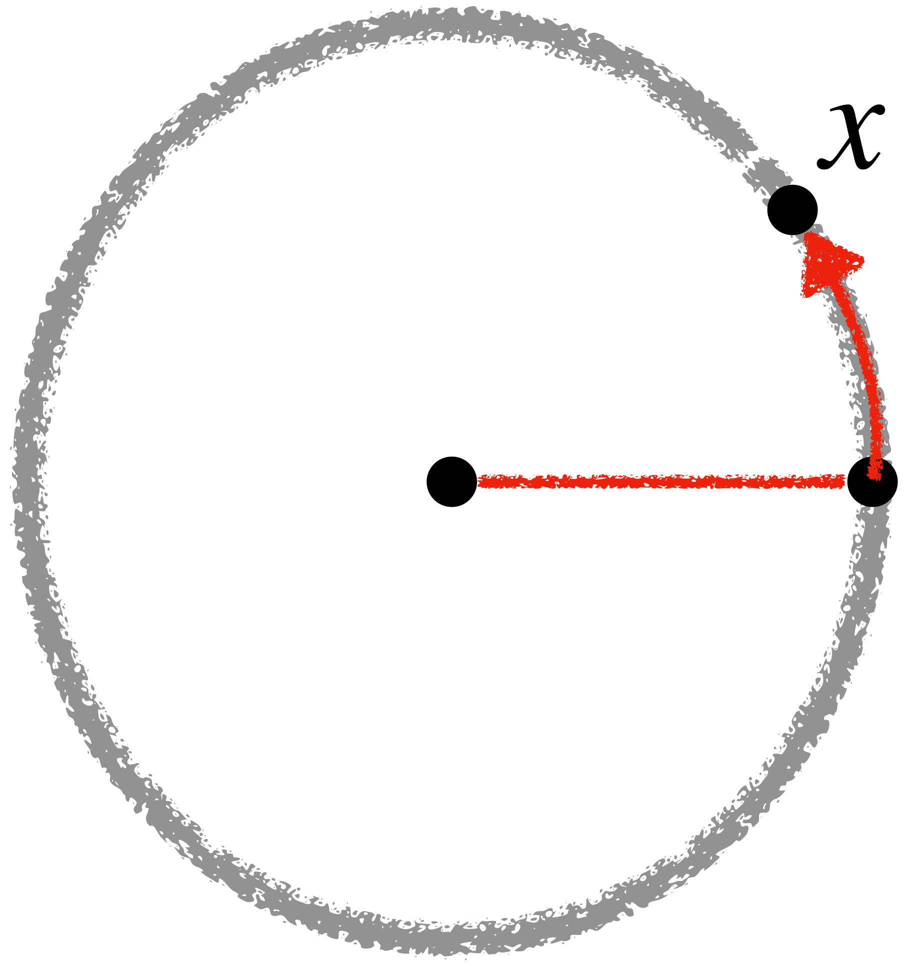
Flow Matching on a Torus

What is the parameterization space?

2 x 2 Rotation matrices $\mathbf{SO}(2)$

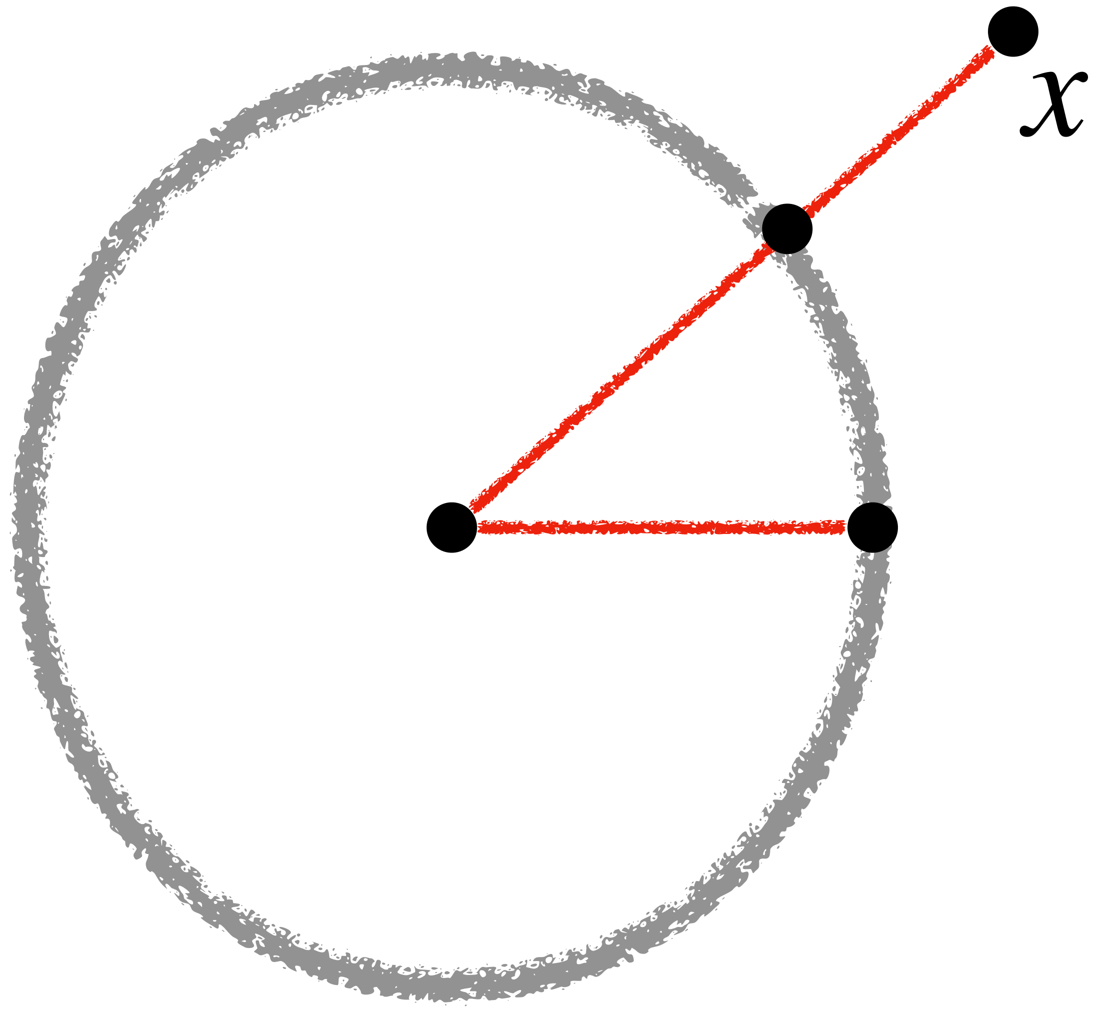


$[-\pi, \pi]$ interval



Simplest but has a discontinuity

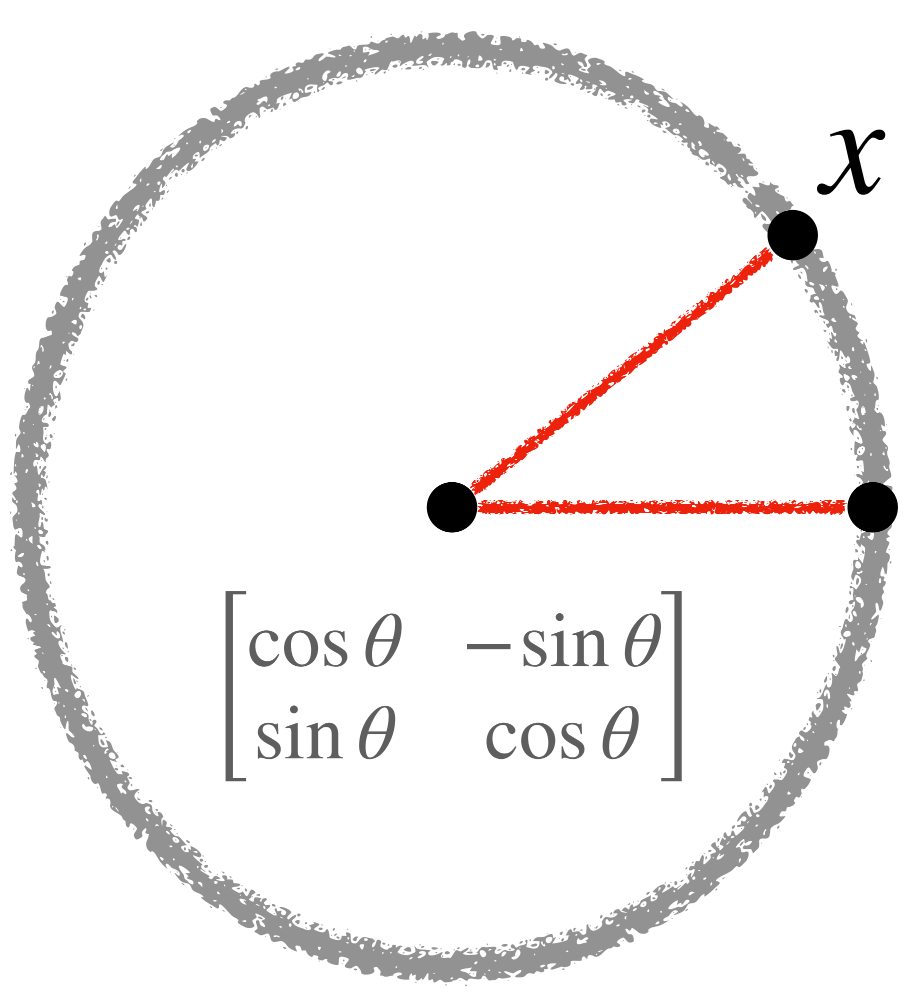
Unit vectors in $\mathbb{R}^2$



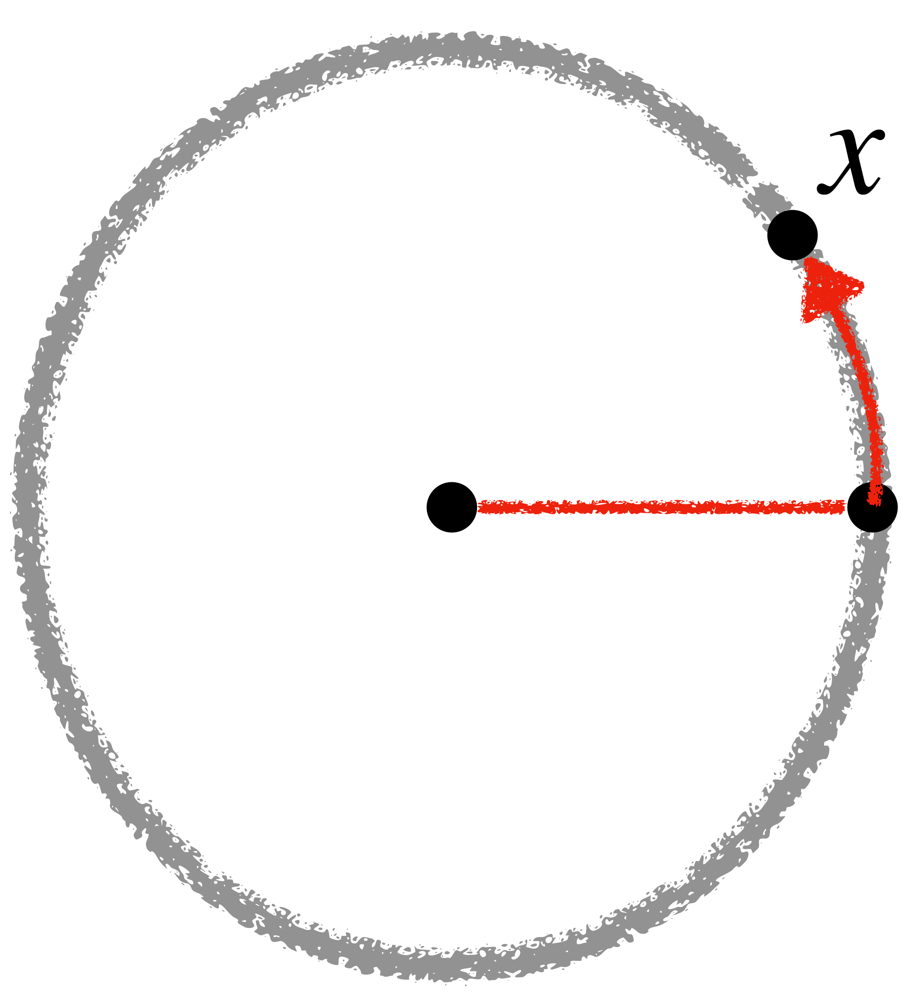
Flow Matching on a Torus

What is the parameterization space?

2 x 2 Rotation matrices $\mathbf{SO}(2)$

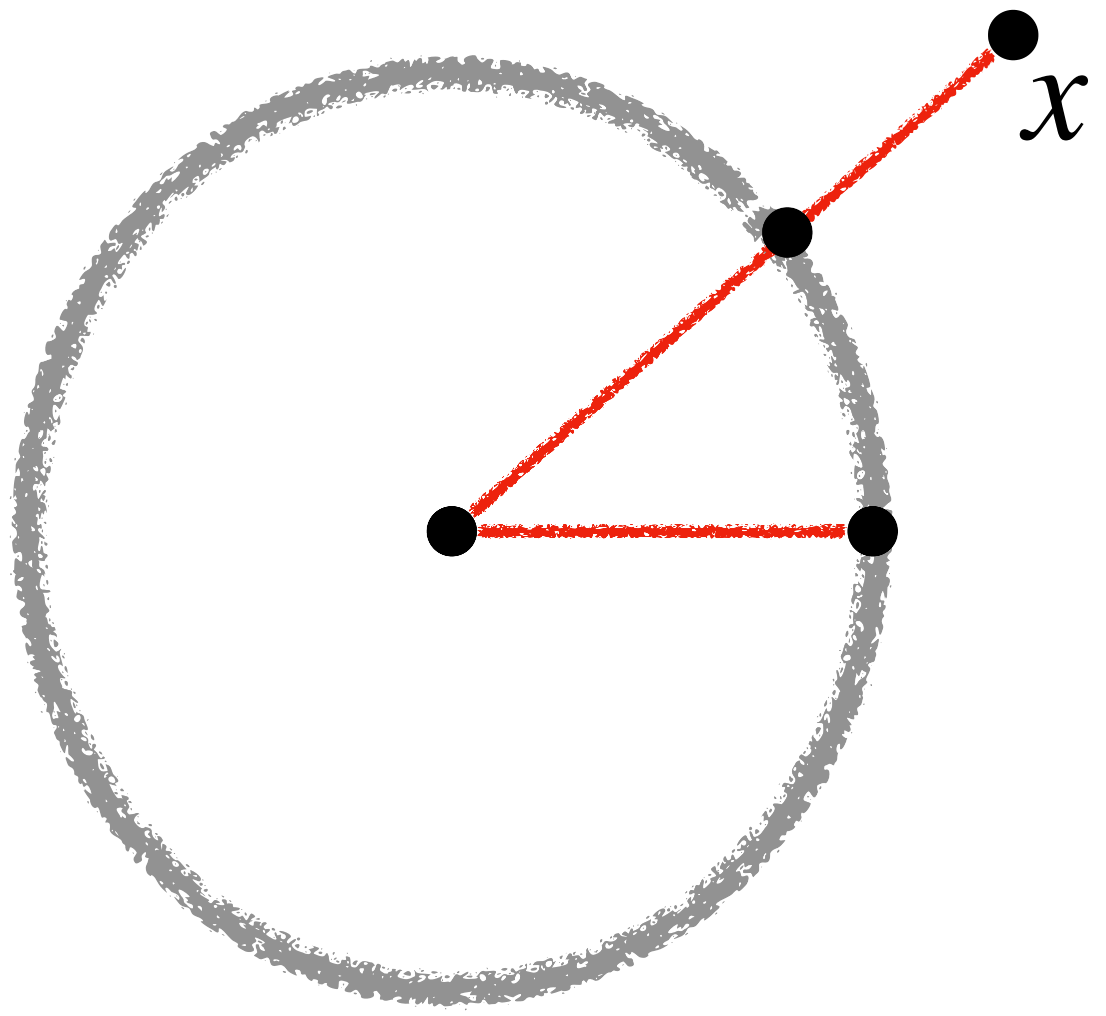


$[-\pi, \pi]$ interval



Simplest but has a discontinuity

Unit vectors in $\mathbb{R}^2$



Higher dimensional but no discontinuity

Practical conclusions in a Riemannian setting

- Flows preferred over diffusion due to ease of construction on manifolds
- Manifolds can be split into parametrizable and non-parametrizable
- Parametrization matters! Making it more like Euclidean is generally good.

Open problems in Geometric Generative Models

- Do you need equivariance?
- How does the parameterization of the model affects the learning dynamics?
- What are the guiding principles for when geometric generative models should be used?
- What is the next paradigm for geometric generative models after flows / diffusion?
- Efficient algorithms for non-parametrizable manifolds?