

# PolyChristoffel Networks

Obtaining latent manifolds via curvature learning

---

Mihir Talati

December 19, 2025

## Motivation & Intuition

---

## Why another dynamics model?

- Physics-style learning often wants **structure**, not just prediction:
  - invariances, conservation laws, interpretable coefficients
  - extrapolation beyond the training distribution
- Many neural dynamics models use generic nonlinearities with limited interpretability.
- Goal here: make the **equations of motion** a first-class object:

learn  $\Gamma(x)$  so that trajectories follow a geodesic equation.

## Geometric intuition: “forces” as coordinate effects

- A straight-line path in one coordinate system can look curved in another.
- Example: inertial motion in Cartesian becomes nontrivial in polar coordinates.
- The apparent “forces” come from geometry via **Christoffel symbols**.

$$\ddot{x}^i + \Gamma^i_{jk}(x) \dot{x}^j \dot{x}^k = 0$$

- Interpret  $\Gamma^i_{jk}(x)$  as **state-dependent couplings** of velocities into acceleration.

## Background: metric, connection, geodesics (minimal)

- A (pseudo-)Riemannian metric  $g(x)$  defines local inner products:

$$\langle u, v \rangle_x = u^\top g(x) v$$

- The Levi-Civita connection (torsion-free, metric-compatible) yields Christoffels:

$$\Gamma_{jk}^i(g) = \frac{1}{2} g^{i\ell} \left( \partial_j g_{k\ell} + \partial_k g_{j\ell} - \partial_\ell g_{jk} \right)$$

- Geodesics are “straightest” paths under that connection:

$$\ddot{x}^i + \Gamma_{jk}^i(x) \dot{x}^j \dot{x}^k = 0.$$

# Model

---

## Model overview (one slide)

**Inputs:** initial state/latent  $x_0$ , initial velocity  $v_0$  (and optionally observed trajectory).

**Learn:** a polynomial Christoffel field  $\Gamma(x)$  (and optionally an autoencoder).

| Observed $y_t$             | Latent/state $x_t$                                    |
|----------------------------|---|
| $x_t = E(y_t) \Rightarrow$ | $\Gamma(x)$ produces geodesic rollout $\hat{x}_{0:T}$ |
| $\hat{y}_t = D(\hat{x}_t)$ | Losses enforce dynamics + metric consistency          |

**Design principle:** keep the dynamics layer interpretable by making  $\Gamma(x)$  a low-degree polynomial.

## Parameterization: polynomial Christoffel network

Let  $x \in \mathbb{R}^d$ . Build monomial features  $\phi(x) \in \mathbb{R}^P$  (degree  $\leq p$ ):

$$\phi(x) = [1, x_1, \dots, x_d, x_1^2, x_1x_2, \dots].$$

Then define

$$\Gamma_{jk}^i(x) = \sum_{m=1}^P C_{jk,m}^i \phi_m(x).$$

- $C_{jk,m}^i$  are trainable coefficients (directly interpretable).
- Enforce torsion-free symmetry by construction:

$$\Gamma_{jk}^i = \Gamma_{kj}^i.$$



## Dynamics block: geodesic equation as a neural layer

Use first-order state  $z = (x, v)$  with  $v = \dot{x}$ :

$$\dot{x} = v, \quad \dot{v}^i = -\Gamma_{jk}^i(x) v^j v^k.$$

This defines a deterministic vector field  $f_\theta(z)$  where parameters are the polynomial coefficients.

### Interpretability:

- acceleration is quadratic in velocity and structured by  $\Gamma(x)$
- no arbitrary activation needed in the dynamics law

## Integrator: ResNet-style rollout (discrete geodesic flow)

A simple differentiable integrator (semi-implicit Euler):

$$v_{t+1} = v_t + \Delta t a(x_t, v_t), \quad x_{t+1} = x_t + \Delta t v_{t+1},$$

where

$$a^i(x, v) = -\Gamma^i_{jk}(x) v^j v^k.$$

- Equivalent to stacking residual blocks with shared parameters.
- Stable, CUDA-friendly, easy to backprop through long rollouts.
- Later swap-in: RK4 / differentiable ODE solvers if needed.

## Optional: jointly learned autoencoder with manifold constraint

When observations live in  $y$ -space (images, fields, etc.), use an encoder/decoder:

$$x_t = E_\psi(y_t), \quad \hat{y}_t = D_\psi(\hat{x}_t).$$

To enforce latent validity on a chosen manifold  $\mathcal{M}$ :

$$x \leftarrow \Pi_{\mathcal{M}}(x)$$

(e.g., sphere projection, hyperboloid retraction).

- Joint training: geometry losses backprop through  $E_\psi$ .
- Encourages a chart where dynamics are “as geodesic as possible.”

## Why metric reconstruction / pseudo-Riemannian regularization?

Learning  $\Gamma(x)$  freely can fit trajectories but yield a connection that is not Levi-Civita of any metric.

To encourage a **(pseudo-)Riemannian interpretation**, impose constraints consistent with:

- torsion-free:  $\Gamma_{jk}^i = \Gamma_{kj}^i$
- metric compatibility:  $\nabla g = 0$
- fixed signature  $(p, q)$ , non-degeneracy ( $\det g \neq 0$ )

## Metric reconstruction idea: integrate the compatibility equation

Metric compatibility in coordinates:

$$\partial_i g_{jk} = \Gamma_{ij}^{\ell} g_{\ell k} + \Gamma_{ik}^{\ell} g_{j\ell}.$$

Treat this as a (path) ODE along a curve  $x(s)$  from a basepoint  $x_0$ :

$$\frac{d}{ds} g_{jk}(s) = \left( \Gamma_{ij}^{\ell}(x(s)) g_{\ell k}(s) + \Gamma_{ik}^{\ell}(x(s)) g_{j\ell}(s) \right) \frac{dx^i}{ds}.$$

- Choose  $g(x_0) = g_0$  with fixed signature; integrate to get  $g(x)$ .
- If  $\Gamma$  is not metric-realizable, the reconstructed  $g(x)$  becomes **path-dependent**.

## Consistency loss: penalize path dependence (loop loss)

Compute  $g(x)$  via two different paths:

- Path A: straight line  $x_0 \rightarrow x$
- Path B: two-segment  $x_0 \rightarrow x_m \rightarrow x$

Then define:

$$\mathcal{L}_{\text{loop}} = \|g^{(A)}(x) - g^{(B)}(x)\|_F^2.$$

### Interpretation:

- pushes  $\Gamma(x)$  toward connections that preserve some bilinear form
- provides extra signal even if you only supervise trajectories

## Pseudo-Riemannian validity losses (practical)

Given reconstructed  $g(x)$ :

- Symmetry:

$$\mathcal{L}_{\text{sym}} = \|g - g^{\top}\|_F^2$$

- Non-degeneracy barrier (avoid singular metric):

$$\mathcal{L}_{\text{det}} = \text{softplus}(\alpha - \log |\det g|)$$

- Fixed signature  $(p, q)$  via eigenvalue sign penalties:

$$\mathcal{L}_{\text{sig}} = \sum_{i=1}^d \text{softplus}(-\beta s_i \lambda_i(g)), \quad s_i \in \{+1, -1\}.$$

## Overall objective and training loop

**Primary fit:** match trajectories (latent or decoded):

$$\mathcal{L}_{\text{traj}} = \frac{1}{T} \sum_t \|\hat{x}_t - x_t\|^2 \quad \text{or} \quad \mathcal{L}_{\text{recon}} = \frac{1}{T} \sum_t \|D(\hat{x}_t) - y_t\|^2.$$

**Regularize geometry:**

$$\mathcal{L} = \mathcal{L}_{\text{traj/recon}} + \lambda_{\text{loop}} \mathcal{L}_{\text{loop}} + \lambda_{\text{det}} \mathcal{L}_{\text{det}} + \lambda_{\text{sig}} \mathcal{L}_{\text{sig}} + \lambda \|C\|_2^2.$$

- Backprop through (i) rollout integrator and (ii) metric reconstruction.
- Coefficients  $C$  remain interpretable (polynomial terms in  $\Gamma$ ).



## Proof of Concept experiment: polar inertial motion

Use a synthetic benchmark with known Christoffels:

$$\Gamma_{\theta\theta}^r = -r, \quad \Gamma_{r\theta}^\theta = \Gamma_{\theta r}^\theta = \frac{1}{r}$$

(others = 0), derived from Euclidean plane in polar coordinates.

- Train  $\Gamma(x)$  to reproduce trajectories.
- Use diagnostics to compare learned components to ground truth.
- Verify torsion-free symmetry and reduced path dependence in reconstructed  $g$ .

# Diagnostics I: Learned Christoffels vs. ground truth

## What we plot

Evaluate the learned connection on a grid  $(r, \theta)$  and extract:

$$\Gamma_{\theta\theta}^r(r, \theta), \quad \Gamma_{r\theta}^\theta(r, \theta)$$

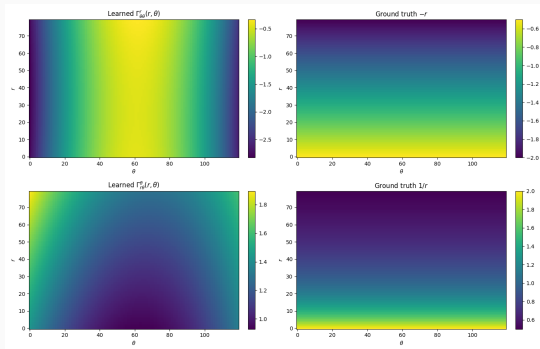
Compare to ground truth:

$$\Gamma_{\theta\theta}^r = -r, \quad \Gamma_{r\theta}^\theta = \frac{1}{r}.$$

## How to interpret

- **$\theta$ -invariance:** GT depends only on  $r$ ,  $(r, \theta)$  grid.  
so learned heatmaps should be *nearly constant across  $\theta$* .

- **Shape:**  $\Gamma_{\theta\theta}^r$  should look like a linear



**Figure 1:** Learned vs. GT Christoffel components on

# Diagnostics II: Reconstructed metric and path-dependence (loop error)

## Metric target (polar plane)

The Euclidean metric in polar coordinates is:

$$g(r, \theta) = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix}.$$

We reconstruct  $g$  from the learned  $\Gamma$  by integrating metric-compatibility:

$$\nabla g = 0 \quad \Rightarrow \quad \partial_i g_{jk} = \Gamma_{ij}^\ell g_{\ell k} + \Gamma_{ik}^\ell g_{j\ell}.$$

## How to interpret

- **Diagonal structure:**  $g_{r\theta} \approx 0$  (off-diagonal heatmap near zero).
- **Component shapes:**  $g_{rr} \approx 1$  (flat),

## Insert figure: Metric diagnostics

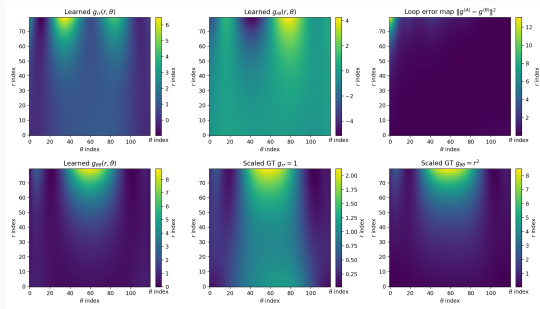
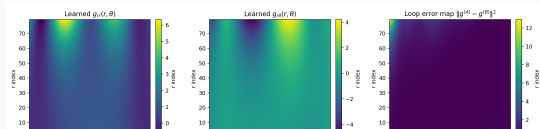


Figure 2: Learned Metric



## Future Directions

---

## Scaling up: leverage strong autoencoders (CNNs, etc.)

- Many domains already have excellent representation learners:
  - CNN autoencoders / VAEs for images and fields
  - pretrained encoders with semantic latent factors
- Proposal: **freeze or lightly finetune** an existing encoder/decoder, then learn  $\Gamma(x)$  (and optionally  $g(x)$ ) on the latent manifold to obtain:
  - interpretable latent dynamics
  - geometry-aware interpolation/extrapolation (geodesics)
  - constraints like signature / nondegeneracy for stability
- The notion of enforcing a Pseudo-Riemannian metric also defines a consistent, invariant dot product for vectors at any point on the manifold, which is also precisely how many encoders are trained (e.g in NLP)

## Research directions (compact)

- Enforce a Levi-Civita connection instead of just torsionlessness to ensure small closed loops return to the same point
- Enforce a notion of "distance" via states through known equivalencies
- **Beyond geodesics:** add forcing/dissipation in a structured way (e.g., geodesic + learned potential or Rayleigh dissipation term).
- **Better metrizability:** directly co-train  $g(x)$  and enforce  $\Gamma_{\text{poly}} \approx \Gamma(g)$  to guarantee Levi-Civita structure.
- **Topology/coverage:** multiple charts or implicit atlas, while retaining interpretability.
- **Stochastic extensions:** geodesic drift with learned diffusion (SDE in latent).

- Christoffel symbols provide an interpretable parameterization of dynamics:

$$\ddot{x}^i = -\Gamma^i_{jk}(x) \dot{x}^j \dot{x}^k$$

- Polynomial  $\Gamma(x)$  yields a small, inspectable model with calculus-friendly smoothness.
- Metric reconstruction and pseudo-Riemannian regularizers provide extra training signals and geometric validity checks.
- Natural next step: apply on strong latent spaces (e.g., CNN-based autoencoders) to learn **curved latent dynamics**.