LEARNING MANIFOLD AND ITÔ DYNAMICS WITH

Branched Neural Rough Differential Equations

Luke Thompson, Dai Shi, Lequan Lin, Junbin Gao, Andi Han

Thesis.

NRDE/log-NCDE are already the right idea:

Thesis.

NRDE/log-NCDE are already the right idea:

1. Summarise each path window by its log-signature

Thesis.

NRDE/log-NCDE are already the right idea:

1. Summarise each path window by its log-signature
2. Advance the hidden state by a coarse log-ODE step

Thesis.

NRDE/log-NCDE are already the right idea:

1. Summarise each path window by its log-signature
2. Advance the hidden state by a coarse log-ODE step

This work

We keep the log-ODE, but choose the Hopf algebra to match the calculus and state space:

$$\lambda_{s,t} = \log_{\mathcal{H}}\left(\mathrm{Sig}^{N}_{\mathcal{H}}(X_{s,t})\right),$$

$$L_{s,t} = \sum_{p \in \mathcal{B}^{\mathrm{prim}}_{\mathcal{H}}} \lambda^{p}_{s,t}\, F_W(p).$$

Thesis.

NRDE/log-NCDE are already the right idea:

1. Summarise each path window by its log-signature
2. Advance the hidden state by a coarse log-ODE step

This work

We keep the log-ODE, but choose the Hopf algebra to match the calculus and state space:

$$\lambda_{s,t} = \log_{\mathcal{H}}\left(\mathrm{Sig}^{N}_{\mathcal{H}}(X_{s,t})\right),$$

$$L_{s,t} = \sum_{p \in \mathcal{B}^{\mathrm{prim}}_{\mathcal{H}}} \lambda^{p}_{s,t}\, F_{W}(p).$$

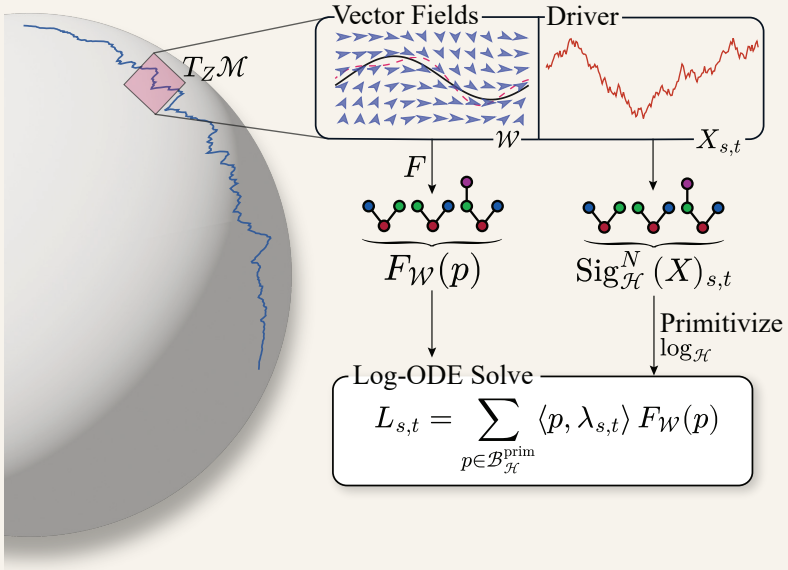


Figure: A driver segment is lifted to the chosen Hopf algebra, primitivised, and mapped through matching vector-field lifts, and solved by a geometric log-ODE step.

How the talk is organised.

§1 · Setup & the gap

From NCDEs to NRDEs.

Why the log-ODE step is the right starting point; what the shuffle algebra commits us to; the two structural limits we want to remove.

§2 · Three Hopf algebras

Matching algebra to calculus.

Shuffle, Grossman–Larson, and Munthe–Kaas–Wright; why Itô and manifold Itô require different primitive coordinates.

§3 · The B-NRDE method

Primitives to vector fields.

Signature primitivisation, pseudo-bialgebra maps, vector-field lifts, and manifold log-ODE steps on homogeneous spaces.

§4 · Training & experiments

Does the algebra matter empirically?

Branched signature kernels; rough Bergomi, sim-to-real $SO(3)$, and Itô dynamics on the SPD cone.

Neural differential equations for time series.

A continuous interpolation $X\colon [t_0, t_n] \to \mathbb{R}^d$ of observations drives a hidden state $h_t \in \mathbb{R}^u$ through a learned vector field g_θ.

Controlled dynamics

$$h_t = h_{t_0} + \int_{t_0}^{t} g_\theta(h_s)\, dX_s.$$

Applications

- Irregularly sampled time series with missing or asynchronous observations.
- Continuous-time forecasting, classification, and filtering.
- Latent dynamics driven directly by an observed input path.

Many follow-on works, some applications to manifolds, notably, $\mathrm{SO}(3)$

How NRDEs buy coarser intervals.

Over a window $I_j = [t_j, t_{j+1}]$, summarise $X|_{I_j}$ by its truncated log-signature λ_j.

The hidden state then obeys an *autonomous* ODE on I_j:

$$h_t = h_{t_j} + \int_{t_j}^{t} \tilde{g}_{\theta,\lambda_j}(h_s)\, ds, \qquad t \in I_j.$$

Here $\tilde{g}_{\theta,\lambda_j}$ is the log-ODE lift of the atomic field:

$$\tilde{g}_{\theta,\lambda_j}(h) = \sum_{\ell \in \mathcal{B}^{\mathrm{prim}}_{\mathcal{H}_{\sqcup\!\sqcup}}} \lambda_j^{\ell} F_W(\ell)(h).$$

This is the construction of Castell & Gaines (1996), applied in Morrill et al. (2021), Walker et al. (2024). The shuffle algebra $\mathcal{H}_{\sqcup\!\sqcup}$ is the algebraic substrate in these cases.

What this earns you

1. Coarser integration steps at fixed accuracy.
2. Robustness to irregular sampling within a window.
3. Higher-order local approximation, not just $O(h)$.

The shuffle algebra has two structural limits.

LIMIT I · ITÔ SEMANTICS

Quadratic is not exposed by unaugmented signatures

Under Itô,

$$X^{(i)}_{s,t} X^{(j)}_{s,t} = \int_s^t X^{(i)}_{s,u}\, dX^{(j)}_u + \int_s^t X^{(j)}_{s,u}\, dX^{(i)}_u + [X^{(i)}, X^{(j)}]_{s,t}.$$

- The symmetric bracket term is not an independent coordinate in $\mathcal{H}_{\sqcup\!\sqcup}$
- It must be added by a Lead–Lag transformation, extended geometric lift, or branched lift

The shuffle algebra has two structural limits.

LIMIT I · ITÔ SEMANTICS

Quadratic is not exposed by unaugmented signatures

Under Itô,

$$X^{(i)}_{s,t} X^{(j)}_{s,t} = \int_s^t X^{(i)}_{s,u}\, dX^{(j)}_u + \int_s^t X^{(j)}_{s,u}\, dX^{(i)}_u + [X^{(i)}, X^{(j)}]_{s,t}.$$

- The symmetric bracket term is not an independent coordinate in $\mathcal{H}_{\sqcup\!\sqcup}$
- It must be added by a Lead–Lag transformation, extended geometric lift, or branched lift

LIMIT II · MANIFOLD STATE

Itô calculus is second order

Itô calculus has a second-order chain rule:

$$df(X_t) = \partial_i f(X_t)\, dX^i_t + \frac{1}{2}\, \partial_{ij} f(X_t)\, d[X^i, X^j]_t.$$

- Changing coordinates in Itô calculus produces an extra correction term.
- This correction uses second derivatives of the coordinate map, weighted by quadratic variation.

Stratonovich is coordinate-free, Itô is not

The shuffle algebra has two structural limits.

LIMIT I · ITÔ SEMANTICS

Quadratic is not exposed by unaugmented signatures

Under Itô,

$$X^{(i)}_{s,t} X^{(j)}_{s,t} = \int_s^t X^{(i)}_{s,u}\, dX^{(j)}_u + \int_s^t X^{(j)}_{s,u}\, dX^{(i)}_u + [X^{(i)}, X^{(j)}]_{s,t}.$$

- The symmetric bracket term is not an independent coordinate in $\mathcal{H}_{\sqcup\!\sqcup}$
- It must be added by a Lead–Lag transformation, extended geometric lift, or branched lift

LIMIT II · MANIFOLD STATE

Itô calculus is second order

Itô calculus has a second-order chain rule:

$$df(X_t) = \partial_i f(X_t)\, dX^i_t + \frac{1}{2}\, \partial_{ij} f(X_t)\, d[X^i, X^j]_t.$$

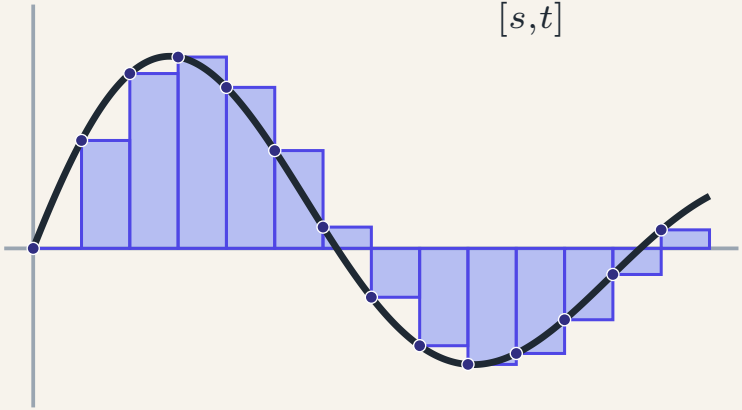
- Changing coordinates in Itô calculus produces an extra correction term.
- This correction uses second derivatives of the coordinate map, weighted by quadratic variation.

Stratonovich is coordinate-free, Itô is not

Both limits are algebraic: they live in the choice of Hopf algebra, not in the network or the solver.

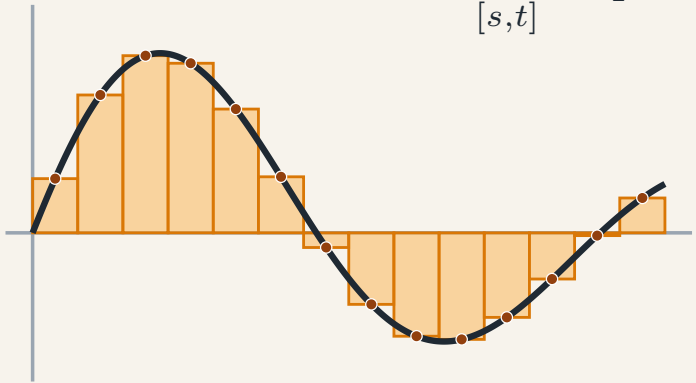
What does this look like?

$$\int X^{(i)} \, \mathrm{d}X^{(j)} =: L^2\text{lim} \sum_{[s,t]} X_s^{(i)} \, X_{s,t}^{(j)}$$



(a) Itô convention.

$$\int X^{(i)} \circ \mathrm{d}X^{(j)} =: L^2\text{lim} \sum_{[s,t]} X_{\frac{s+t}{2}}^{(i)} \, X_{s,t}^{(j)}$$



(b) Stratonovich convention.

Figure: Riemann–Stieltjes approximations under different evaluation conventions.

Our Contributions.

Keep the log-ODE step, but change the algebra so it matches the calculus and geometry of the problem.

Contributions

Our Contributions.

Keep the log-ODE step, but change the algebra so it matches the calculus and geometry of the problem.

Contributions

1. **Handle Itô NRDEs with $\mathcal{H}_{\mathrm{GL}}$.**
   Rooted trees expose quadratic variation directly.

Our Contributions.

Keep the log-ODE step, but change the algebra so it matches the calculus and geometry of the problem.

Contributions

1. **Handle Itô NRDEs with $\mathcal{H}_{\mathrm{GL}}$.**
   Rooted trees expose quadratic variation directly.
2. **Handle Manifold Itô NRDEs with $\mathcal{H}_{\mathrm{MKW}}$.**
   Planar trees retain ordered covariant derivatives.

Our Contributions.

Keep the log-ODE step, but change the algebra so it matches the calculus and geometry of the problem.

Contributions

1. **Handle Itô NRDEs with $\mathcal{H}_{\mathrm{GL}}$.**
   Rooted trees expose quadratic variation directly.
2. **Handle Manifold Itô NRDEs with $\mathcal{H}_{\mathrm{MKW}}$.**
   Planar trees retain ordered covariant derivatives.
3. **A unified NRDE construction.**
   Pseudo bialgebra maps turn word/tree coordinates into vector-field lifts.

Our Contributions.

Keep the log-ODE step, but change the algebra so it matches the calculus and geometry of the problem.

Contributions

1. **Handle Itô NRDEs with $\mathcal{H}_{\mathrm{GL}}$.**
   Rooted trees expose quadratic variation directly.
2. **Handle Manifold Itô NRDEs with $\mathcal{H}_{\mathrm{MKW}}$.**
   Planar trees retain ordered covariant derivatives.
3. **A unified NRDE construction.**
   Pseudo bialgebra maps turn word/tree coordinates into vector-field lifts.
4. **Branched signature-kernel training.**
   The objective compares enhanced laws, including bracket coordinates.

Our Contributions.

Keep the log-ODE step, but change the algebra so it matches the calculus and geometry of the problem.

Contributions

1. **Handle Itô NRDEs with $\mathcal{H}_{\mathrm{GL}}$.**
   Rooted trees expose quadratic variation directly.
2. **Handle Manifold Itô NRDEs with $\mathcal{H}_{\mathrm{MKW}}$.**
   Planar trees retain ordered covariant derivatives.
3. **A unified NRDE construction.**
   Pseudo bialgebra maps turn word/tree coordinates into vector-field lifts.
4. **Branched signature-kernel training.**
   The objective compares enhanced laws, including bracket coordinates.
5. **Roughrax**
   A JAX package to make manifold log-ODE solving accessible and easy.

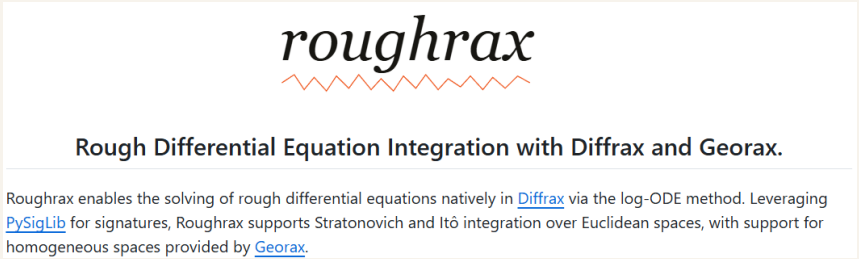


Figure: Roughrax makes our method available to the diffrax community.

PART TWO

II

Two calculi, three Hopf algebras.

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathfrak{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathfrak{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

01 · LIFT PATH

Path $\to$ Hopf algebra.

Compute the truncated $\mathcal{H}$-signature of the segment.

$$\mathrm{Sig}^{N}_{\mathcal{H}}(X_k)$$

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

01 · LIFT PATH

Path $\to$ Hopf algebra.

Compute the truncated $\mathcal{H}$-signature of the segment.

$\mathsf{Sig}^N_{\mathcal{H}}(X_k)$

$\longrightarrow$

02 · PRIMITIVISE

Hopf logarithm.

Project the segment signature onto primitive coordinates of $\mathcal{H}$.

$\lambda_k = \log_{\mathcal{H}} \mathsf{Sig}^N_{\mathcal{H}}(X_k)$

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

01 · LIFT PATH

Path $\to$ Hopf algebra.

Compute the truncated $\mathcal{H}$-signature of the segment.

$\mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

02 · PRIMITIVISE

Hopf logarithm.

Project the segment signature onto primitive coordinates of $\mathcal{H}$.

$\lambda_k = \log_{\mathcal{H}} \mathrm{Sig}^N_{\mathcal{H}}(X_k)$

Steps 01, 02 cached offline: precompute λ_k on the windowing once per dataset.

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

01 · LIFT PATH

Path $\to$ Hopf algebra.

Compute the truncated $\mathcal{H}$-signature of the segment.

$\mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

02 · PRIMITIVISE

Hopf logarithm.

Project the segment signature onto primitive coordinates of $\mathcal{H}$.

$\lambda_k = \log_{\mathcal{H}} \mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

03 · LIFT FIELDS

Pseudo bialgebra map.

Atomic neural vector fields W_θ generate primitive-indexed fields on $\mathcal{M}$.

$F_W : \mathcal{B}^{\mathrm{prim}}_{\mathcal{H}} \to \Gamma(T\mathcal{M})$

Steps 01, 02 cached offline: precompute λ_k on the windowing once per dataset.

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

01 · LIFT PATH

Path $\to$ Hopf algebra.

Compute the truncated $\mathcal{H}$-signature of the segment.

$\mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

02 · PRIMITIVISE

Hopf logarithm.

Project the segment signature onto primitive coordinates of $\mathcal{H}$.

$\lambda_k = \log_{\mathcal{H}} \mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

03 · LIFT FIELDS

Pseudo bialgebra map.

Atomic neural vector fields W_θ generate primitive-indexed fields on $\mathcal{M}$.

$F_W : \mathcal{B}^{\mathrm{prim}}_{\mathcal{H}} \to \Gamma(T\mathcal{M})$

$\to$

04 · SOLVE

Log-ODE on $\mathcal{M}$.

Heun in the flat case, or a commutator-free Lie-group integrator on manifolds.

$\dot{Z}_\tau = L_k(Z_\tau), \qquad Z_0 = Y_k$

Steps 01, 02 cached offline: precompute λ_k on the windowing once per dataset.

One log-ODE template, three algebra choices.

The key idea is that every model in this talk follows the same four-stage pipeline. What changes is only the Hopf algebra $\mathcal{H} \in \{\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}, \mathcal{H}_{\mathrm{GL}}, \mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}\}$, which determines the primitive coordinates and the matching lifted vector fields.

01 · LIFT PATH

Path $\to$ Hopf algebra.

Compute the truncated $\mathcal{H}$-signature of the segment.

$\mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

02 · PRIMITIVISE

Hopf logarithm.

Project the segment signature onto primitive coordinates of $\mathcal{H}$.

$\lambda_k = \log_{\mathcal{H}} \mathrm{Sig}^N_{\mathcal{H}}(X_k)$

$\to$

03 · LIFT FIELDS

Pseudo bialgebra map.

Atomic neural vector fields W_θ generate primitive-indexed fields on $\mathcal{M}$.

$F_W : \mathcal{B}^{\mathrm{prim}}_{\mathcal{H}} \to \Gamma(T\mathcal{M})$

$\to$

04 · SOLVE

Log-ODE on $\mathcal{M}$.

Heun in the flat case, or a commutator-free Lie-group integrator on manifolds.

$\dot{Z}_\tau = L_k(Z_\tau), \qquad Z_0 = Y_k$

Steps 01, 02 cached offline: precompute λ_k on the windowing once per dataset.

Steps 03, 04 occur online via autograd for the VFs or Diffrax for the solve.

Each calculus & Space has an Algebra

We will work with the cocommutative graded duals, so the pseudo bialgebra Leibniz identity will make the maps produce a first-order vector field.

HOPF ALGEBRA	PRIMITIVE BASIS	STRATONOVICH	EUCLIDEAN ITÔ	MANIFOLD ITÔ

Each calculus & Space has an Algebra

We will work with the cocommutative graded duals, so the pseudo bialgebra Leibniz identity will make the maps produce a first-order vector field.

HOPF ALGEBRA	PRIMITIVE BASIS	STRATONOVICH	EUCLIDEAN ITÔ	MANIFOLD ITÔ
Tensor $\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}$ (Morrill et al. '21)	Lyndon words	✓	×	×

COCOMMUTATIVE DUAL OF

the shuffle algebra.
Primitives = free Lie algebra.

Each calculus & Space has an Algebra

We will work with the cocommutative graded duals, so the pseudo bialgebra Leibniz identity will make the maps produce a first-order vector field.

HOPF ALGEBRA	PRIMITIVE BASIS	STRATONOVICH	EUCLIDEAN ITÔ	MANIFOLD ITÔ
Tensor $\mathcal{H}_{\sqcup\!\sqcup}^{\mathfrak{g}}$ (Morrill et al. '21)	Lyndon words	✓	×	×
Grossman–Larson $\mathcal{H}_{\mathrm{GL}}$ **(this work)**	Rooted trees	×	✓	×

COCOMMUTATIVE DUAL OF

the shuffle algebra.
Primitives = free Lie algebra.

COCOMMUTATIVE DUAL OF

the Butcher–Connes–Kreimer Hopf algebra.
Primitives = free pre-Lie algebra.

Each calculus & Space has an Algebra

We will work with the cocommutative graded duals, so the pseudo bialgebra Leibniz identity will make the maps produce a first-order vector field.

HOPF ALGEBRA	PRIMITIVE BASIS	STRATONOVICH	EUCLIDEAN ITÔ	MANIFOLD ITÔ
Tensor $\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}$ (Morrill et al. '21)	Lyndon words	✓	×	×
Grossman–Larson $\mathcal{H}_{\mathrm{GL}}$ **(this work)**	Rooted trees	×	✓	×
Munthe–Kaas–Wright $\mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}$ **(this work)**	Planar rooted trees	×	✓	✓

COCOMMUTATIVE DUAL OF

the shuffle algebra.
Primitives = free Lie algebra.

COCOMMUTATIVE DUAL OF

the Butcher–Connes–Kreimer Hopf algebra.
Primitives = free pre-Lie algebra.

COCOMMUTATIVE DUAL OF

the MKW coordinate Hopf algebra.
Primitives = free post-Lie algebra.

Of Sigs & Logsigs

1. Signature side

A path segment is first summarised by a truncated signature in the chosen Hopf algebra:

$$\mathbf{X}_{s,t}^{\mathcal{H},N} = \mathrm{Sig}_{\mathcal{H}}^{N}(X_{s,t}) \in \mathcal{H}_{\leq N}.$$

In coordinates, this means

$$X_{s,t}^{H,N} = \sum_{p \in B_{H,\leq N}} X_{s,t}^{p}\, p.$$

So...

- The basis $\mathcal{B}_{\mathcal{H}}$ determines which iterated integrals
- The product & coproduct determine the algebra

Of Sigs & Logsigs

1. Signature side

A path segment is first summarised by a truncated signature in the chosen Hopf algebra:

$$\mathbf{X}_{s,t}^{\mathcal{H},N} = \mathrm{Sig}_{\mathcal{H}}^{N}(X_{s,t}) \in \mathcal{H}_{\leq N}.$$

In coordinates, this means

$$X_{s,t}^{H,N} = \sum_{p \in B_{H,\leq N}} X_{s,t}^{p}\, p.$$

So...

- The basis $\mathcal{B}_{\mathcal{H}}$ determines which iterated integrals
- The product & coproduct determine the algebra

2. Logarithm side

The log-signature step in an NRDE is not tied to words or to the shuffle setting. It only needs a group-like signature in a Hopf algebra:

$$\mathbf{X}_{s,t}^{\mathcal{H}} \in \mathcal{H}, \qquad \Delta \mathbf{X}_{s,t}^{\mathcal{H}} = \mathbf{X}_{s,t}^{\mathcal{H}} \otimes \mathbf{X}_{s,t}^{\mathcal{H}}.$$

The Hopf logarithm sends group-like elements to primitive elements:

$$\log_{\mathcal{H}}(g) := \sum_{n \geq 1} \frac{(-1)^{n-1}}{n} (g-1)^{\star n}.$$

Thus, the segment is represented in primitive coordinates in the chosen algebra:

$$\lambda_{s,t} = \log_{\mathcal{H}} \mathbf{X}_{s,t}^{\mathcal{H}} \in \mathrm{Prim}(\mathcal{H}).$$

NRDE & log-NCDE are the case $\mathcal{H} = \mathcal{H}_{\sqcup\!\sqcup}$;
B-NRDE asks what if we change $\mathcal{H}$?

The same coordinates must act on the state.

From the driver side, each segment has become a primitive expansion

$$\lambda_{s,t} = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}} \lambda_{s,t}^{p}\, p.$$

The same coordinates must act on the state.

The network does *not* learn one vector field for every primitive. As in log-NCDE, it learns only d atomic vector fields for a d-dimensional driver:

$$W_\theta(z) = \big(W_\theta^{(1)}(z), \dots, W_\theta^{(d)}(z)\big), \qquad W_\theta^{(i)}(z) \in T_z\mathcal{M}.$$

The same coordinates must act on the state.

The network does *not* learn one vector field for every primitive. As in log-NCDE, it learns only d atomic vector fields for a d-dimensional driver:

$$W_\theta(z) = \big(W_\theta^{(1)}(z), \dots, W_\theta^{(d)}(z)\big), \qquad W_\theta^{(i)}(z) \in T_z\mathcal{M}.$$

How can we get these vector fields into the same basis as the $\log_{\mathcal{H}}$-signature?

The same coordinates must act on the state.

The network does *not* learn one vector field for every primitive. As in log-NCDE, it learns only d atomic vector fields for a d-dimensional driver:

$$W_\theta(z) = \big(W_\theta^{(1)}(z), \dots, W_\theta^{(d)}(z)\big), \qquad W_\theta^{(i)}(z) \in T_z\mathcal{M}.$$

How can we get these vector fields into the same basis as the $\log_{\mathcal{H}}$-signature?

The missing ingredient

This is provided by a pseudo bialgebra map

$$F_{W_\theta} : \mathcal{H} \to \mathrm{Diff}(\mathcal{M}).$$

When restricted to primitives, it gives vector fields:

$$p \in \mathrm{Prim}(\mathcal{H}) \quad \longmapsto \quad F_{W_\theta}(p) \in \Gamma(T\mathcal{M}).$$

Thus the segment update is driven by the matched log-ODE field

$$L_{s,t}(z) = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}} \lambda_{s,t}^{p}\, F_{W_\theta}(p)(z).$$

How do the primitive coordinates act?

A pseudo bialgebra map preserves the product/composition structure and the coproduct/Leibniz structure. Hence primitives act as derivations, i.e. as first-order vector fields.

Historically, the primitive algebras came before their enveloping Hopf algebras, i.e. BCK/GL and MKW.

How do the primitive coordinates act?

A pseudo bialgebra map preserves the product/composition structure and the coproduct/Leibniz structure. Hence primitives act as derivations, i.e. as first-order vector fields.

Free Lie algebra

PRIMITIVE BASIS

Lyndon words: minimal among their non-trivial cyclic rotations.

$$abac \quad \cancel{acab} \quad \cancel{baca}$$
$$aabc \quad abbc \quad \cancel{bcab}$$

Shuffle / geometric / Stratonovich.

Historically, the primitive algebras came before their enveloping Hopf algebras, i.e. BCK/GL and MKW.

How do the primitive coordinates act?

A pseudo bialgebra map preserves the product/composition structure and the coproduct/Leibniz structure. Hence primitives act as derivations, i.e. as first-order vector fields.

Free Lie algebra

PRIMITIVE BASIS
Lyndon words: minimal among their non-trivial cyclic rotations.

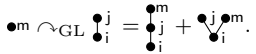
$$abac \quad \cancel{acab} \quad \cancel{baca}$$
$$aabc \quad abbc \quad \cancel{bcab}$$

Shuffle / geometric / Stratonovich.

Pre-Lie algebra

PRIMITIVE BASIS
Decorated non-planar rooted trees.

PRODUCT
All-vertex grafting:



Grossman–Larson / Euclidean Itô.

Historically, the primitive algebras came before their enveloping Hopf algebras, i.e. BCK/GL and MKW.

How do the primitive coordinates act?

A pseudo bialgebra map preserves the product/composition structure and the coproduct/Leibniz structure. Hence primitives act as derivations, i.e. as first-order vector fields.

Free Lie algebra

PRIMITIVE BASIS

Lyndon words: minimal among their non-trivial cyclic rotations.

abac ~~*acab*~~ ~~*baca*~~

aabc *abbc* ~~*bcab*~~

Shuffle / geometric / Stratonovich.

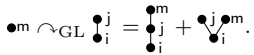
Pre-Lie algebra

PRIMITIVE BASIS

Decorated non-planar rooted trees.

PRODUCT

All-vertex grafting:

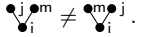


Grossman–Larson / Euclidean Itô.

Post-Lie algebra

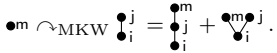
PRIMITIVE BASIS

Decorated planar rooted trees; child order is part of the data.



PRODUCT

Left grafting:



Munthe–Kaas–Wright / manifold Itô.

Historically, the primitive algebras came before their enveloping Hopf algebras, i.e. BCK/GL and MKW.

Where is quadratic variation in the log-signature?

In the branched Itô signature, depth-two ladder trees store ordered Itô iterated integrals. After taking the Hopf logarithm, their symmetric part is exactly the quadratic variation.

Depth-two ladder coefficients

The ladder

$$\begin{smallmatrix} \bullet^{\mathrm{j}} \\ \bullet_{\mathrm{i}} \end{smallmatrix}$$

corresponds to the ordered Itô integral

$$\mathbf{X}^{\begin{smallmatrix} \bullet^{\mathrm{j}} \\ \bullet_{\mathrm{i}} \end{smallmatrix}}_{s,t} = \int_s^t X^{(j)}_{s,u}\, dX^{(i)}_u.$$

After the Hopf logarithm,

$$\lambda^{\begin{smallmatrix} \bullet^{\mathrm{j}} \\ \bullet_{\mathrm{i}} \end{smallmatrix}}_{\mathrm{GL},s,t} = \mathbf{X}^{\begin{smallmatrix} \bullet^{\mathrm{j}} \\ \bullet_{\mathrm{i}} \end{smallmatrix}}_{s,t} - \frac{1}{2} X^{(i)}_{s,t} X^{(j)}_{s,t}.$$

Where is quadratic variation in the log-signature?

In the branched Itô signature, depth-two ladder trees store ordered Itô iterated integrals. After taking the Hopf logarithm, their symmetric part is exactly the quadratic variation.

Depth-two ladder coefficients

The ladder

$$\begin{smallmatrix}\bullet^{\mathrm{j}}\\ |\\ \bullet_{\mathrm{i}}\end{smallmatrix}$$

corresponds to the ordered Itô integral

$$\mathbf{X}^{\mathrm{j}\,\mathrm{i}}_{s,t} = \int_s^t X^{(j)}_{s,u}\, dX^{(i)}_u.$$

After the Hopf logarithm,

$$\lambda^{\mathrm{j}\,\mathrm{i}}_{\mathrm{GL},s,t} = \mathbf{X}^{\mathrm{j}\,\mathrm{i}}_{s,t} - \frac{1}{2} X^{(i)}_{s,t} X^{(j)}_{s,t}.$$

Quadratic variation is the symmetric ladder part

Adding the two ordered ladders removes the ordered area information and leaves the bracket:

$$\boxed{\lambda^{\mathrm{j}\,\mathrm{i}}_{\mathrm{GL},s,t} + \lambda^{\mathrm{i}\,\mathrm{j}}_{\mathrm{GL},s,t} = -\langle X^{(i)}, X^{(j)} \rangle_{s,t}.}$$

In particular, the diagonal entries are

$$\boxed{\lambda^{\mathrm{i}\,\mathrm{i}}_{\mathrm{GL},s,t} = -\frac{1}{2} \langle X^{(i)} \rangle_{s,t}.}$$

Quadratic variation is not added as a separate channel; it is already visible in the symmetric depth-two ladder log-coordinates.

Why Munthe–Kaas–Wright, not planar CK?

Planarity records the order of children, but it does not by itself encode the covariant chain rule. The product must reproduce the ordered identity

$$\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla^2 V_i(V_m, V_j) + \underbrace{\nabla_{\nabla_{V_m} V_j} V_i}_{\text{Correction}}.$$

This says differentiate V_i in direction V_j, then take this result and differentiate it in direction V_m.

Why Munthe–Kaas–Wright, not planar CK?

Planarity records the order of children, but it does not by itself encode the covariant chain rule. The product must reproduce the ordered identity

$$\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla^2 V_i(V_m, V_j) + \underbrace{\nabla_{\nabla_{V_m} V_j} V_i}_{\text{Correction}}.$$

This says differentiate V_i in direction V_j, then take this result and differentiate it in direction V_m.

but...

Why Munthe–Kaas–Wright, not planar CK?

Planarity records the order of children, but it does not by itself encode the covariant chain rule. The product must reproduce the ordered identity

$$\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla^2 V_i(V_m, V_j) + \underbrace{\nabla_{\nabla_{V_m} V_j} V_i}_{\text{Correction}}.$$

This says differentiate V_i in direction V_j, then take this result and differentiate it in direction V_m.

but...

V_j itself is a vector field, so it changes as we move in the direction V_m. The product must include exactly this correction, and no extra term.

Why Munthe–Kaas–Wright, not planar CK?

We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Why Munthe–Kaas–Wright, not planar CK?

We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:

Why Munthe–Kaas–Wright, not planar CK?

We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:

Why Munthe–Kaas–Wright, not planar CK?

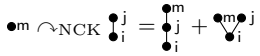
We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:



Why Munthe–Kaas–Wright, not planar CK?

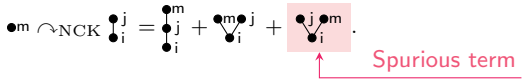
We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:



Why Munthe–Kaas–Wright, not planar CK?

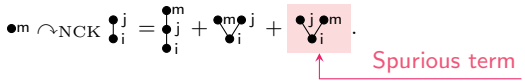
We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:



The total lifted field is therefore

$$\begin{aligned} F_{\mathrm{NCK}} &= \underbrace{F\left(\begin{smallmatrix} m \\ j \\ i \end{smallmatrix}\right) + F\left(\begin{smallmatrix} m & j \\ & i \end{smallmatrix}\right)}_{\text{correct target}} + \underbrace{F\left(\begin{smallmatrix} j & m \\ & i \end{smallmatrix}\right)}_{\text{extra term}} \\ &= \underbrace{\nabla_{\nabla_{V_m} V_j} V_i + \nabla^2 V_i(V_m, V_j)}_{=\nabla_{V_m}(\nabla_{V_j} V_i)=0} + \underbrace{\nabla^2 V_i(V_j, V_m)}_{=-V_j} \\ &= -V_j \neq 0. \end{aligned}$$

Why Munthe–Kaas–Wright, not planar CK?

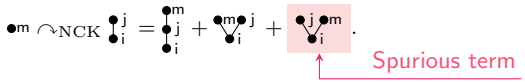
We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:



The total lifted field is therefore

$$\begin{aligned} F_{\mathrm{NCK}} &= \underbrace{F\left(\begin{smallmatrix} m \\ j \\ i \end{smallmatrix}\right) + F\left(\begin{smallmatrix} m & j \\ & i \end{smallmatrix}\right)}_{\text{correct target}} + \underbrace{F\left(\begin{smallmatrix} j & m \\ & i \end{smallmatrix}\right)}_{\text{extra term}} \\ &= \underbrace{\nabla_{\nabla_{V_m} V_j} V_i + \nabla^2 V_i(V_m, V_j)}_{=\nabla_{V_m}(\nabla_{V_j} V_i)=0} + \underbrace{\nabla^2 V_i(V_j, V_m)}_{=-V_j} \\ &= -V_j \neq 0. \end{aligned}$$

The naive planar CK product changes the whole lifted vector field.

Why Munthe–Kaas–Wright, not planar CK?

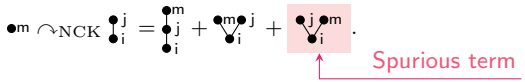
We test this at a fixed point $y \in \mathcal{M}$, with

$$\nabla_{V_j} V_i = V_j, \qquad \nabla_{V_m} V_j = 0, \qquad \nabla_{V_m} V_i = 0, \qquad \nabla_{V_j} V_m = V_j.$$

Thus, the desired value is $\nabla_{V_m}(\nabla_{V_j} V_i) = \nabla_{V_m} V_j = 0$.

Naive planar CK: wrong total

Ordinary planar admissible cuts allow one extra term:



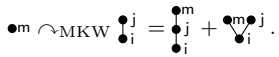
The total lifted field is therefore

$$\begin{aligned} F_{\mathrm{NCK}} &= \underbrace{F\left(\begin{smallmatrix}\bullet^{m}\\ \bullet^{j}\\ \bullet_{i}\end{smallmatrix}\right) + F\left(\begin{smallmatrix}\bullet^{m}\bullet^{j}\\ \bullet_{i}\end{smallmatrix}\right)}_{\text{correct target}} + \underbrace{F\left(\begin{smallmatrix}\bullet^{j}\bullet^{m}\\ \bullet_{i}\end{smallmatrix}\right)}_{\text{extra term}} \\ &= \underbrace{\nabla_{\nabla_{V_m} V_j} V_i + \nabla^2 V_i(V_m, V_j)}_{=\nabla_{V_m}(\nabla_{V_j} V_i)=0} + \underbrace{\nabla^2 V_i(V_j, V_m)}_{=-V_j} \\ &= -V_j \neq 0. \end{aligned}$$

The naive planar CK product changes the whole lifted vector field.

MKW: the corrected covariant product

MKW left grafting omits precisely that extra cut:



Under the elementary differential map,

$$\begin{aligned} F_{\mathrm{MKW}} &= F\left(\begin{smallmatrix}\bullet^{m}\\ \bullet^{j}\\ \bullet_{i}\end{smallmatrix}\right) + F\left(\begin{smallmatrix}\bullet^{m}\bullet^{j}\\ \bullet_{i}\end{smallmatrix}\right) \\ &= \nabla_{\nabla_{V_m} V_j} V_i + \nabla^2 V_i(V_m, V_j) \\ &= \nabla_{V_m}(\nabla_{V_j} V_i) \\ &= \nabla_{V_m} V_j \\ &= 0. \end{aligned}$$

$\mathcal{H}_{\mathrm{MKW}}$ only keeps the terms required by covariant differentiation.

Vector field lifts: Evaluating Pseudo Bialgebra Maps

We use a shared automatic differentiation scheme for all Hopf algebras. Recall, we have atomic vector fields $W_\theta^{(1)}, \dots, W_\theta^{(d)}$

Lyndon Words

1. Enumerate Lyndon words ℓ via Duval's algorithm
2. Store the standard factorisation

$$\ell = uv, \qquad v \text{ the longest Lyndon suffix},$$

3. Recurse

$$F_W(\ell) = \begin{cases} W^{(i)}, & \ell = i, \\ [F_W(u), F_W(v)], & \ell = [u, v], \end{cases}$$

Vector field lifts: Evaluating Pseudo Bialgebra Maps

We use a shared automatic differentiation scheme for all Hopf algebras. Recall, we have atomic vector fields $W_\theta^{(1)}, \dots, W_\theta^{(d)}$

Lyndon Words

1. Enumerate Lyndon words ℓ via Duval's algorithm
2. Store the standard factorisation

$$\ell = uv, \qquad v \text{ the longest Lyndon suffix},$$

3. Recurse

$$F_W(\ell) = \begin{cases} W^{(i)}, & \ell = i, \\ [F_W(u), F_W(v)], & \ell = [u, v], \end{cases}$$

Trees

1. Enumerate coloured trees via Beyer-Hedetniemi, $p = (\tau, c)$
2. Store the child list at each node

$$C(v) = (u_1, \dots, u_k),$$

ordered for $\mathcal{H}^{\mathrm{g}}_{\mathrm{MKW}}$, and fixed once for $\mathcal{H}_{\mathrm{GL}}$.

3. Evaluate subtrees bottom-up

$$V_v = \begin{cases} W^{(c(v))}, & v \text{ is a leaf}, \\ \nabla^k_{V_{u_1}, \dots, V_{u_k}} W^{(c(v))}, & C(v) = (u_1, \dots, u_k). \end{cases}$$

4. Return the field at the root

$$F_W(p) = V_r.$$

Manifold Log-ODE

Building the Log-ODE

Now that the logsig and VFs are in the same primitive basis, the truncated log-signature defines an intrinsic vector field:

$$L_k(Y) = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}} \lambda_k^p F_W(p)(Y), \qquad L_k(Y) \in T_Y M.$$

Manifold Log-ODE

Building the Log-ODE

Now that the logsig and VFs are in the same primitive basis, the truncated log-signature defines an intrinsic vector field:

$$L_k(Y) = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}} \lambda_k^p F_W(p)(Y), \qquad L_k(Y) \in T_Y M.$$

The log-ODE step is the time-one flow of this vector field:

$$\dot{Z}_\tau = L_k(Z_\tau), \qquad Z_0 = Y_k, \qquad \tau \in [0, 1],$$

and we set $Y_{k+1} \approx Z_1$.

Manifold Log-ODE

Building the Log-ODE

Now that the logsig and VFs are in the same primitive basis, the truncated log-signature defines an intrinsic vector field:

$$L_k(Y) = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}} \lambda_k^p F_W(p)(Y), \qquad L_k(Y) \in T_Y M.$$

The log-ODE step is the time-one flow of this vector field:

$$\dot{Z}_\tau = L_k(Z_\tau), \qquad Z_0 = Y_k, \qquad \tau \in [0, 1],$$

and we set $Y_{k+1} \approx Z_1$.

On Manifolds

The connection enters the lifts $F_W(p)$ through the iterated covariant derivatives.
The action/retraction enters the numerical solve.

Manifold Log-ODE

Building the Log-ODE

Now that the logsig and VFs are in the same primitive basis, the truncated log-signature defines an intrinsic vector field:

$$L_k(Y) = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}} \lambda_k^p F_W(p)(Y), \qquad L_k(Y) \in T_Y M.$$

The log-ODE step is the time-one flow of this vector field:

$$\dot{Z}_\tau = L_k(Z_\tau), \qquad Z_0 = Y_k, \qquad \tau \in [0,1],$$

and we set $Y_{k+1} \approx Z_1$.

On Manifolds

The connection enters the lifts $F_W(p)$ through the iterated covariant derivatives.
The action/retraction enters the numerical solve.

An implementation detail

Why not ambient RK? For homogeneous spaces, we use the group action and take stages of the form

$$\exp(\Omega) \cdot Y_k \in M.$$

Thus every stage is a valid manifold point.

For rotations, an ambient RK stage has the form

$$R_k + hK \notin SO(3),$$

so it creates numerical defects $R^\top R - I \neq 0$, $\det R - 1 \neq 0$.

Geometric integrators eliminate this error; the remaining error is the intended solver/signature truncation error.

How does this look in code with Roughrax?

How does this look in code with Roughrax?

Take the signature:

```
coarse_ts = fine_ts[::32] # 32-long signature windows
control = SignatureInterpolation(
    driver,
    coarse_ts,
    depth=3,
    solution="stratonovich",
)
```

How does this look in code with Roughrax?

Take the signature:

```
coarse_ts = fine_ts[::32] # 32-long signature windows
control = SignatureInterpolation(
    driver,
    coarse_ts,
    depth=3,
    solution="stratonovich",
)
```

Define your RoughTerm™:

```
term = RoughTerm(
    lambda: some_vf,
    control,
    Euclidean(),
)
```

How does this look in code with Roughrax?

Take the signature:

```
coarse_ts = fine_ts[::32] # 32-long signature windows
control = SignatureInterpolation(
    driver,
    coarse_ts,
    depth=3,
    solution="stratonovich",
)
```

Define your RoughTerm™:

```
term = RoughTerm(
    lambda: some_vf,
    control,
    Euclidean(),
)
```

And solve!

```
from diffrax import Heun

sol = diffrax.diffeqsolve(
    term,
    LogODE(Heun()), # 2nd order RK in log-ODE
    t0=float(coarse_ts[0]),
    t1=float(coarse_ts[-1]),
    dt0=None,
    y0=jnp.asarray(1.0),
    stepsize_controller=diffrax.StepTo(coarse_ts),
    saveat=diffrax.SaveAt(ts=coarse_ts),
)
```

And on a manifold

Take the signature:

```
coarse_ts = fine_ts[::32] # 32-long signature windows
control = SignatureInterpolation(
    driver,
    coarse_ts,
    depth=3,
    solution="ito",  # Planarly branched, MKW
)
```

Define your RoughTerm™:

```
term = RoughTerm(
    lambda: some_so3_vf,
    control,
    SO(3),
)
```

And solve!

```
from diffrax import Heun
from georax import RKMK # Another package of mine

sol = diffrax.diffeqsolve(
    term,
    LogODE(RKMK(Heun())), # Lie group 2nd order RK
    t0=float(coarse_ts[0]),
    t1=float(coarse_ts[-1]),
    dt0=None,
    y0=jnp.eye(3),
    stepsize_controller=diffrax.StepTo(coarse_ts),
    saveat=diffrax.SaveAt(ts=coarse_ts),
)
```

Performance of the Geometric Log-ODE

Video Demo

Complexity: what changes when we change the algebra?

Primitive basis size drives the algebraic cost

At depth N, the log-ODE field has one lifted vector field per primitive:

$$L_{s,t} = \sum_{p \in \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}(N)} \lambda_{s,t}^{p} \, F_W(p).$$

The primitive counts are

$$b_{\sqcup\!\sqcup}(n) = \frac{1}{n} \sum_{k \mid n} \mu(k) d^{n/k}, \quad b_{\mathrm{GL}}(n) = t_n d^n, \quad b_{\mathrm{MKW}}(n) = C_{n-1} d^n.$$

$$\left| \mathcal{B}_{\mathcal{H}}^{\mathrm{prim}}(N) \right| = \sum_{n=1}^{N} b_{\mathcal{H}}(n).$$

- Shuffle: Lyndon words.
- Grossman–Larson: non-planar rooted trees.
- Munthe–Kaas–Wright: planar rooted trees.

Complexity: what changes when we change the algebra?

Derivative cost: Euclidean versus homogeneous-space

Normalise one primal evaluation of the atomic vector field f_θ to C_θ. Using one JVP $\approx 3C_\theta$:

$$\mathcal{C}^{(N)}_{\sqcup\!\sqcup} = 3\left|\mathcal{B}^{\mathrm{prim}}_{\sqcup\!\sqcup}(N-1)\right| C_\theta, \qquad \mathcal{C}^{(N)}_{\mathrm{GL}} = 3\left|\mathcal{B}^{\mathrm{prim}}_{\mathrm{GL}}(N-1)\right| C_\theta,$$

$$\mathcal{C}^{(N)}_{\sqcup\!\sqcup,\nabla} \approx 6\left|\mathcal{B}^{\mathrm{prim}}_{\sqcup\!\sqcup}(N-1)\right| C_\theta, \quad \mathcal{C}^{(N)}_{\mathrm{MKW},\nabla} \approx 3\sum_{n=1}^{N}(n-1)b_{\mathrm{MKW}}(n)C_\theta.$$

Why homogeneous-space shuffle costs more

A non-letter Lyndon primitive is evaluated by a Lie bracket. With a homogeneous-space frame, the shuffle lift still represents ordinary Lie brackets of vector fields. In frame coordinates, the bracket is evaluated as

$$[\![X, Y]\!] = \nabla_X Y - \nabla_Y X + [X, Y]_{\mathrm{frame}},$$

where $[X, Y]_{\mathrm{frame}}$ is the Lie-algebra bracket in the Lie-group case, and the induced frame bracket in the homogeneous-space case. Equivalently, it is the correction from the moving frame. It accounts for torsion, or failure of the frame to commute.

Complexity: what changes when we change the algebra?

Instantiated at depth $N = 2$

C_θ = one primal evaluation of f_θ, J = solver steps per window, $C_{\mathcal{M}}$ = one group exponential/action.

(R, Q) = field evaluations and exponentials per solver step.

Space / algebra	Primitive dimension	Derivative multiplier	(R, Q)	Cost per window
$\mathbb{R}$, $\mathcal{H}_{\sqcup\!\sqcup}$	$\frac{d(d+1)}{2}$	d	$(2, 0)$	$6JdC_\theta$
$\mathbb{R}$, $\mathcal{H}_{\mathrm{GL}}$	$d + d^2$	d	$(2, 0)$	$6JdC_\theta$
$\mathcal{M}$, $\mathcal{H}_{\sqcup\!\sqcup}$	$\frac{d(d+1)}{2}$	$2d$	$(3, 3)$	$18JdC_\theta + 3JC_{\mathcal{M}}$
$\mathcal{M}$, $\mathcal{H}_{\mathrm{MKW}}$	$d + d^2$	d^2	$(3, 3)$	$9Jd^2C_\theta + 3JC_{\mathcal{M}}$

- Euclidean models use Heun: two field evaluations per step.
- Homogeneous-space models use CF-EES$(2, 5)$: three field evaluations and three exponentials per step.

PART THREE

III

Experimental Results.

Branched Signature Kernel Objective.

Why include this?

Signature kernels compare laws of paths through their signature coordinates.

For a semimartingale observed on a grid, the piecewise linear lift has zero bracket - no QV.

This is not the main contribution of the paper; it tests whether the same branched structure that improves the log-ODE step is also useful as a training objective.

Branched Signature Kernel Objective.

Why include this?

Signature kernels compare laws of paths through their signature coordinates.

For a semimartingale observed on a grid, the piecewise linear lift has zero bracket - no QV.

This is not the main contribution of the paper; it tests whether the same branched structure that improves the log-ODE step is also useful as a training objective.

Branched signature-kernel objective

Fix a Hopf algebra $\mathcal{H}$, truncation depth N, and the coordinate basis used by $\mathrm{Sig}^N_{\mathcal{H}}$. For enhanced drivers $\mathbf{X}, \mathbf{Y}$, define

$$k^N_{\mathrm{br}}(\mathbf{X}, \mathbf{Y}) = \left\langle \mathrm{Sig}^N_{\mathcal{H}}(\mathbf{X}), \mathrm{Sig}^N_{\mathcal{H}}(\mathbf{Y}) \right\rangle_{\mathcal{H}_{\leq N}}.$$

The corresponding MMD objective is

$$\mathcal{L}_{\mathrm{br}}(\theta) = \mathbb{E}_{\mathbf{X},\mathbf{X}' \sim P_\theta}\left[k^N_{\mathrm{br}}(\mathbf{X}, \mathbf{X}')\right] - 2\mathbb{E}_{\mathbf{X}\sim P_\theta, \mathbf{Y}\sim P_{\mathrm{data}}}\left[k^N_{\mathrm{br}}(\mathbf{X}, \mathbf{Y})\right].$$

The omitted data–data term is constant in θ.

Branched Signature Kernel Objective.

Why include this?

Signature kernels compare laws of paths through their signature coordinates.

For a semimartingale observed on a grid, the piecewise linear lift has zero bracket - no QV.

This is not the main contribution of the paper; it tests whether the same branched structure that improves the log-ODE step is also useful as a training objective.

Branched signature-kernel objective

Fix a Hopf algebra $\mathcal{H}$, truncation depth N, and the coordinate basis used by $\mathrm{Sig}^N_{\mathcal{H}}$. For enhanced drivers $\mathbf{X}, \mathbf{Y}$, define

$$k^N_{\mathrm{br}}(\mathbf{X}, \mathbf{Y}) = \left\langle \mathrm{Sig}^N_{\mathcal{H}}(\mathbf{X}), \mathrm{Sig}^N_{\mathcal{H}}(\mathbf{Y}) \right\rangle_{\mathcal{H}_{\leq N}}.$$

The corresponding MMD objective is

$$\mathcal{L}_{\mathrm{br}}(\theta) = \mathbb{E}_{\mathbf{X},\mathbf{X}' \sim P_\theta}\left[k^N_{\mathrm{br}}(\mathbf{X}, \mathbf{X}')\right] - 2\mathbb{E}_{\mathbf{X}\sim P_\theta, \mathbf{Y}\sim P_{\mathrm{data}}}\left[k^N_{\mathrm{br}}(\mathbf{X}, \mathbf{Y})\right].$$

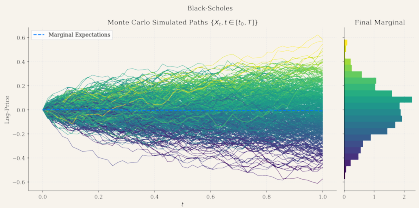
The omitted data–data term is constant in θ.

In Practice

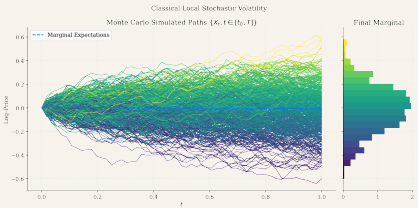
A piecewise linear lift has no QV, so we either

- Insert realised bracket increments $\Delta X^i_k \Delta X^j_k$; for standard continuous semimartingales, fixed-time realised-covariance fluctuations are typically $O(\sqrt{\Delta_n})$.
- Insert analytic/simulator bracket increments when available (i.e., in rough volatility experiments)

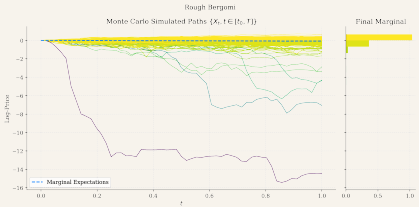
Rough Volatility



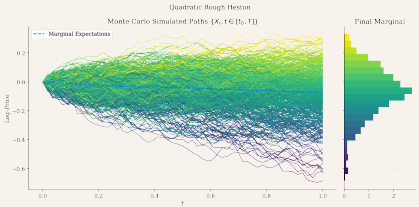
(a) Black–Scholes.



(b) Local Stochastic Volatility.



(c) Rough Bergomi.



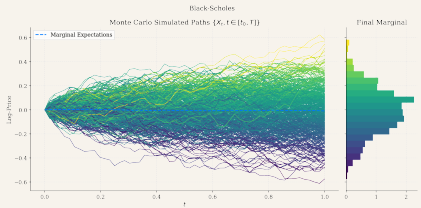
(d) Quadratic Rough Heston.

Figure: Log prices of four stochastic volatility models.

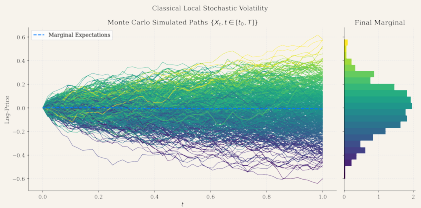
Motivation & Design

- Rough-volatility models posit that log-volatility has fractional regularity.
- Rough Bergomi (rBergomi) is a particularly strong case, with the potential for strong SPX calibration
- Can we speed up simulation with neural samplers?
- Good performance may suggest B-NRDEs could learn stochastic volatility models from data
- Flat Itô setting, so we use $\mathcal{H}_{\mathrm{GL}}$
  - All models train with $\mathcal{L}_{\mathrm{geo}}$; B-NRDE then receives three additional $\mathcal{L}_{\mathrm{br}}$ fine-tuning epochs.
  - Add covariance data to paths: oracle for ground truth, squared increments for model-produced paths as $(\Delta X_t)^2 \approx V_t \Delta t$ (integrated variance)

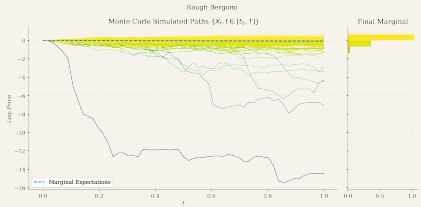
Rough Volatility



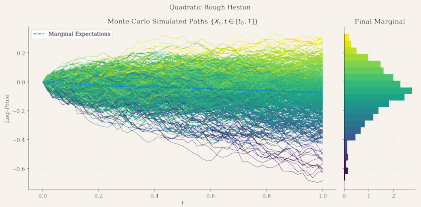
(a) Black–Scholes.



(b) Local Stochastic Volatility.



(c) Rough Bergomi.



(d) Quadratic Rough Heston.

Figure: Log prices of four stochastic volatility models.

Experimental Procedure

- Generate 1024 rBergomi paths
- The model takes in a latent Brownian motion and outputs a log-price path
- Training minimises the signature kernel score between the model and the ground truth log-prices
- Time marginal KS test between model and ground truth paths to measure if we have learned the law.

Rough Volatility

METHOD	TRAINING TIME (s)	KS SCORE ($\times 10^{-2}$)			
		128	256	384	512
xLSTM	38	11.04±1.38	10.31±1.30	10.94±0.89	11.71±0.71
NCDE	1154	9.06±1.94	8.93±1.07	9.75±1.02	9.65±0.81
NCDE++	12021	8.11±0.93	8.82±0.86	9.67±0.34	10.73±0.52
Stacked xLSTM	313	7.69±0.16	9.19±1.09	9.50±1.21	10.95±1.11
log-NCDE	74	7.75±1.39	7.38±0.46	8.72±0.88	8.89±0.24
NRDE	157	8.74±0.96	8.41±0.94	8.12±0.79	**7.47±0.83**
GRU	42	7.73±1.78	7.59±0.58	7.87±0.67	8.02±1.15
B-NRDE (GK)	233	**6.89±1.81**	**6.91±0.98**	8.56±0.38	9.58±0.36
B-NRDE (BK)	47	7.93±1.49	7.70±0.87	**7.77±1.02**	8.11±0.52

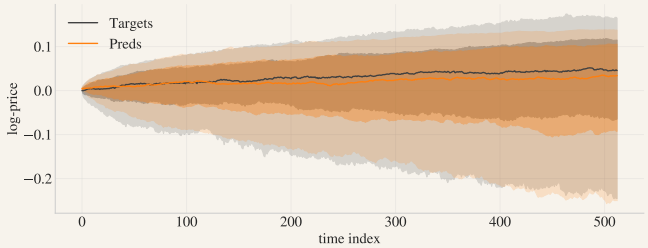


Figure: rBergomi sample paths generated by B-NRDE showing good fit to the underlying law.

SO(3) Motion Forecasting

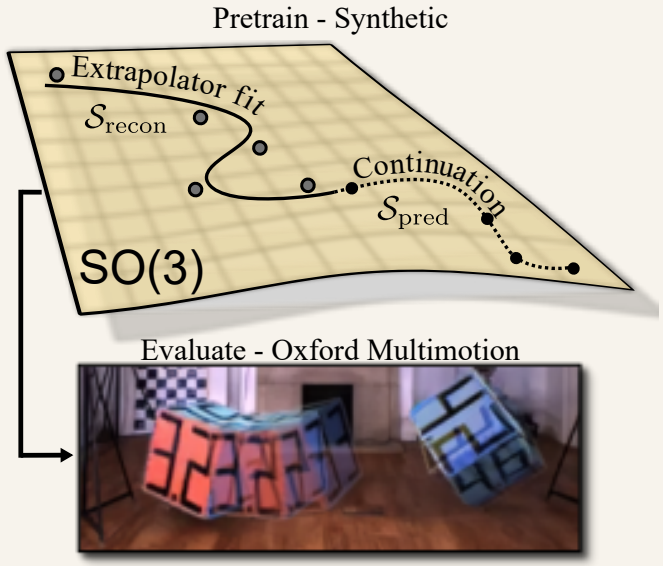


Figure: Schematic Diagram of the Forecasting Task

Motivation & Design

- Forecasting of rotational dynamics is a headline task in robotics, motion capture, and pose estimation
- Non-probabilistic formulation, so we use $\mathcal{H}^{\mathrm{g}}_{\sqcup\!\sqcup}$
- Labelled pose-estimation data is rare, so we are concerned with sim-to-real generalisation
  - Train on simulated data - Synthetic $\mathrm{SO}(3)$ trajectories with varying dampening, timescales, etc
  - Test on limited labelled real-world data - Oxford Multimotion

SO(3) Motion Forecasting

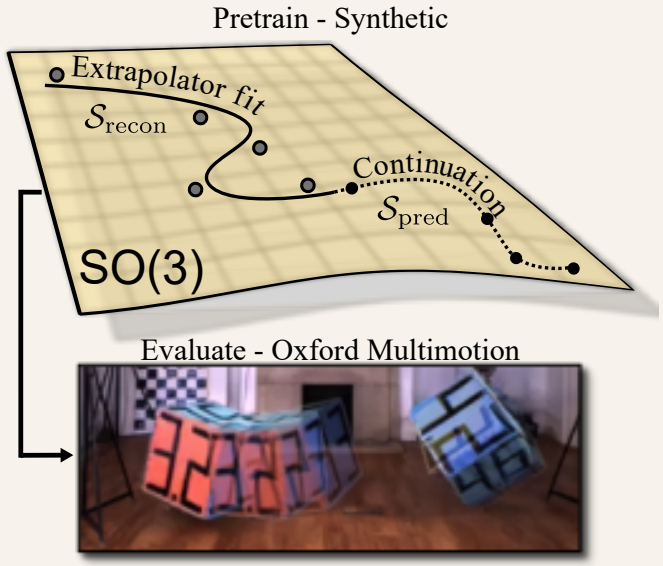


Figure: Schematic Diagram of the Forecasting Task

Experimental Procedure

- Split each window into a reconstruction prefix $\mathcal{S}_{\text{recon}}$ and prediction suffix $\mathcal{S}_{\text{pred}}$.
- Fit an extrapolator E on $\mathcal{S}_{\text{recon}}$, then evaluate it on $\mathcal{S}_{\text{recon}} \cup \mathcal{S}_{\text{pred}}$ to obtain a dense input path $\widetilde{R}(t)$.
- Feed $\widetilde{R}(t)$ to the model to produce rotations $\hat{R}_\theta(t)$. Train against the ground-truth rotations $R(t_j)$ over the full window using RGE:

$$\mathrm{RGE}\Big(\widehat{R}_\theta(t_j), R(t_j)\Big) = 2\arcsin\left(\frac{\left\|\widehat{R}_\theta(t_j) - R(t_j)\right\|_{\mathrm{F}}}{2\sqrt{2}}\right), \qquad t_j \in \mathcal{S}_{\text{recon}} \cup \mathcal{S}_{\text{pred}}.$$

SO(3) Motion Forecasting

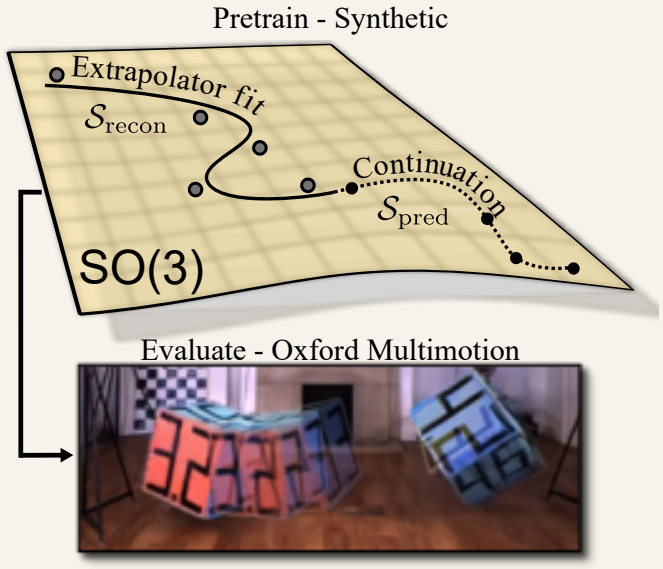


Figure: Schematic Diagram of the Forecasting Task

METHOD	PRETRAINING FROBENIUS	STATIC MOTION	TRANSLATION MOTION	UNCONSTRAINED MOTION
M-NODE	2.461	132.40	118.25	121.08
SO(3)-NCDE	0.294	22.21	21.83	22.45
SO(3)-GRU	0.271	20.25	20.39	21.13
xLSTM	0.233	16.67	16.77	17.36
NRDE	0.097	7.45	7.34	7.32
Stacked xLSTM	0.110	7.11	7.50	7.03
SG-NCDE	0.050	**2.93**	**3.31**	**2.80**
B-NRDE (GK)	**0.049**	3.23	3.70	3.33

B-NRDE is not uniformly better, but requires no feature engineering.

Learning Generative Dynamics on $\mathcal{S}_{++}^d$

Motivation & Design

- Covariance dynamics arise in many fields
  - Multivariate finance
  - Medical imaging
  - Riemannian diffusion
- These dynamics evolve on the manifold of symmetric positive definite matrices

$$\mathcal{S}_{++}^d = \{P \in \mathbb{R}^{d\times d} : P = P^\top,\ P \succ 0\}.$$

- This manifold is the homogeneous space

$$\mathcal{S}_{++}^d \simeq \mathrm{GL}(d)/\mathrm{O}(d)$$

under the congruence action

$$A \cdot P = APA^\top, \qquad A \in \mathrm{GL}(d),\ P \in \mathcal{S}_{++}^d.$$

Learning Generative Dynamics on $\mathcal{S}^d_{++}$

Experimental Procedure

- Learn to generate sample paths of the SDE

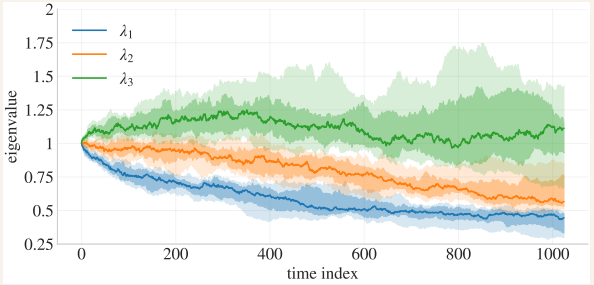
$$\mathrm{d}X_t = \eta \log_{X_t}(M)\,\mathrm{d}t + \sigma\, X_t^{1/2}\,\mathrm{d}B_t\, X_t^{1/2},$$

  with mean reversion strength $\eta > 0$, noise scale $\sigma > 0$, and symmetric matrix Brownian B_t
- Minimise the signature kernel score between ground truth and predicted paths

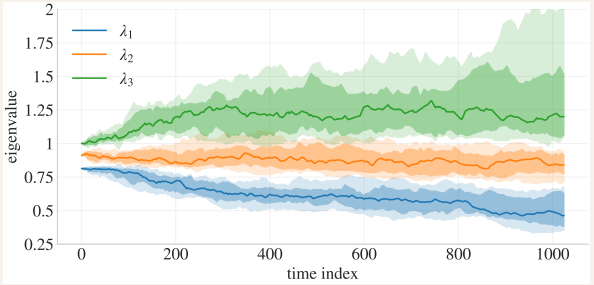
Learning Generative Dynamics on $\mathcal{S}^d_{++}$

METHOD	TRAINING TIME (s)	1-WASSERSTEIN ($\times 10^{-2}$)			
		128	256	384	512
NCDE++	1605	64.95±0.98	58.27±1.36	50.56±1.73	42.15±2.22
NCDE	1241	65.56±1.74	58.72±2.38	50.32±3.02	42.38±3.11
NRDE	232	11.80±0.15	15.50±0.16	17.36±0.13	17.70±0.28
xLSTM	12	8.36±0.63	10.49±0.41	13.16±0.39	14.09±0.53
log-NCDE	481	12.10±0.30	14.76±0.22	16.17±0.32	16.81±0.40
Stacked xLSTM	18	**5.50±0.10**	6.85±0.35	11.47±0.63	14.30±0.46
B-NRDE (GK)	495	5.73±0.20	**5.81±0.19**	**6.28±0.86**	8.35±0.94
B-NRDE (BK)	502	5.81±0.20	5.87±0.21	6.30±0.85	**8.32±0.94**

Improves over Euclidean log-NCDE, no meaningful GK/BK difference. Weak fidelity near initial condition.



(a) Ground truth eigenvalue trajectories.



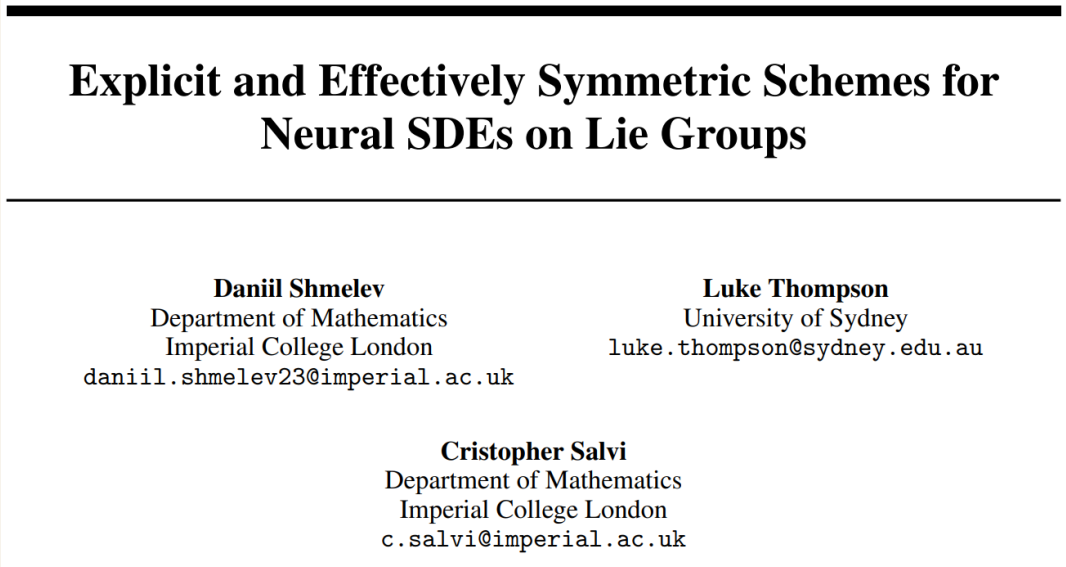
(b) Predicted eigenvalue trajectories.

Future Directions

When first approaching this paper idea, there were no numerical integrators for solving manifold neural ODEs (i.e., our geometric log-ODE).

Future Directions

When first approaching this paper idea, there were no numerical integrators for solving manifold neural ODEs (i.e., our geometric log-ODE).



Explicit and Effectively Symmetric Schemes for Neural SDEs on Lie Groups

Daniil Shmelev
Department of Mathematics
Imperial College London
daniil.shmelev23@imperial.ac.uk

Luke Thompson
University of Sydney
luke.thompson@sydney.edu.au

Cristopher Salvi
Department of Mathematics
Imperial College London
c.salvi@imperial.ac.uk

Figure: Effectively reversible geometric ODE solver: CF-EES$(2,5)$.

Future Directions

For the Signatures

1. The log signature compresses the signature by quotienting out the shuffle relations, but no analogous projection is known for branched rough paths. Recent work of Bellingeri–Ferrucci–Tapia (2024) suggests such a branched projection exists.

2. For scalar RDEs like rough volatility, multi-indices rough paths of $\mathcal{H}_{\mathrm{LOT}}$ exploit symmetries in the vector field, offering maximal compression

Stochastic PDEs

The rough path philosophy extends to SPDEs via regularity structures.

Planarly branched rough paths extend rough paths to manifolds; Rahm's planar regularity structures do the same for regularity structures.

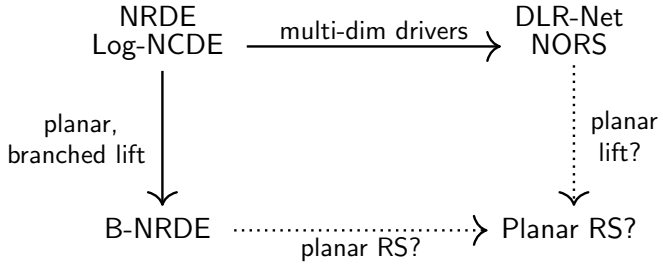


Figure: Positioning of our method and a hypothetical planar-regularity-structure extension.

Thanks

Thank you.

References I