Explicit and Effectively Symmetric Schemes for Neural SDEs on Lie Groups

Anonymous Author(s)
Affiliation
Address
email

Abstract

Backpropagation through (neural) SDE solvers is traditionally approached in two ways: discretise-then-optimise, which offers accurate gradients but incurs prohibitive memory costs; and optimise-then-discretise, which achieves constant memory cost by solving an auxiliary backward SDE, but suffers from slower evaluation and gradient approximation errors. Algebraically reversible solvers promise both memory efficiency and gradient accuracy, yet existing methods such as Reversible Heun are often unstable under complex models and large step sizes, and their non-standard auxiliary-state structure obstructs extension to manifold-valued SDEs. Building on the recently introduced *Explicit and Effectively Symmetric (EES)* schemes – a class of stable, near-reversible explicit Runge–Kutta methods – we address both limitations of existing schemes. We extend EES schemes from ODEs to SDEs and show that they admit an efficient Williamson $2N$-storage realisation. Bazavov's commutator-free construction then lifts these schemes to arbitrary Lie groups and homogeneous spaces. To our knowledge, this is the first explicit (near-)reversible integrator in this setting, unlocking the reversible adjoint approach for manifold-valued problems. On Euclidean neural SDE benchmarks, our schemes improve stability under stiff drift and large steps compared with other reversible solvers, while the commutator-free lift reduces memory by up to an order of magnitude on manifold-valued problems versus other baselines. These results establish effectively symmetric integration as a unified, geometry-aware foundation for memory-efficient and stable training of neural SDEs.

1 Introduction

Neural stochastic differential equations (NSDEs) have recently emerged as a flexible tool for modelling stochastic dynamics, with training typically cast as a distribution-matching problem between generated and observed trajectories. Several approaches have been proposed in the literature, differing mainly in the choice of discriminating divergence. SDE-GANs [49] use the 1-Wasserstein distance, while Latent SDEs [56] optimise with respect to the KL divergence via variational inference. Another alternative proposed by [41] trains neural SDEs non-adversarially using maximum mean discrepancies (MMD) with *signature kernels* [50, 80, 55], a recently introduced family of efficient kernels on path space [81, 54, 72, 64]. Across these formulations, training requires differentiating through an SDE solver, making the numerical method a central part of the learning algorithm.

A second pressure on the design of an SDE integrator comes from the geometry of the state space. Many stochastic dynamics of practical interest evolve in non-Euclidean spaces. In biology, torus-valued processes underpin peptide design and torsion-angle prediction [57, 97, 24]. In motion capture, articulated poses evolve on products of rotations [90, 94, 2]. In finance, asset-return covariances evolve on the SPD manifold [68, 43]. These geometries share a common structure: they are Lie groups, products of Lie groups, or are generated by Lie-group actions. In such settings, leaving the

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

manifold breaks the required geometric constraints, and training therefore requires differentiating through a geometry-preserving solver.

Adjoint methods provide a family of approaches to perform this backpropagation through the solver. A first approach, known as *discretise-then-optimise*, directly backpropagates through the solver's internal operations. This yields the exact gradient of the discretised computation and is computationally efficient, but requires storing all intermediate states, making it memory-intensive ($\mathcal{O}(n)$); we refer to this method as the **Full** adjoint. A second approach, *optimise-then-discretise*, instead derives a backward-in-time adjoint equation and solves it numerically using another call to the solver. This avoids storing the full forward trajectory, resulting in constant memory with respect to the number of solver steps. However, it generally does not return the exact gradient of the discretised solver and is often slower due to the need to recompute forward trajectories during the backward pass. Practical implementations use recursive checkpointing [88] to balance memory and recomputation, yielding an intermediate $\mathcal{O}(\sqrt{n})$ memory regime; we refer to this hybrid as the **Recursive** adjoint.

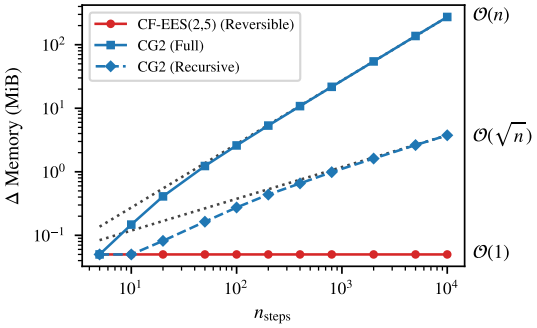


Figure 1: *Growth in memory requirement for one forward and backward solve of a batch of 1024 SDEs on the 7-torus $\mathbb{T}^7$. Our method is in red.*

A third option leverages *algebraically reversible solvers*, which enable exact reconstruction of the solution trajectory from the terminal state, and hence permit accurate and memory-efficient backpropagation in $\mathcal{O}(1)$ memory; we refer to this as the **Reversible** adjoint. In the setting of an autonomous ODE

$$\mathrm{d}y_t = f(y_t)\,\mathrm{d}t, \tag{1}$$

a one step method $y_{n+1} = y_n + \Phi_h(y_n)$ is said to be *reversible* or *symmetric* if a step of the method starting from y_1 with a negative step size exactly recovers the initial condition y_0, that is, $\Phi_{-h} = \Phi_h^{-1}$. While reversible schemes offer an efficient approach to backpropagation through differential equations, such schemes are difficult to construct. It is well known that Runge–Kutta schemes are reversible only if they are implicit, making them unsuitable for applications to Neural ODEs. More generally, symmetric parasitism-free general linear methods cannot be explicit [15].

To overcome this problem, existing reversible methods proposed in literature, such as the Reversible Heun method [48] and the McCallum-Foster methods [60], track auxiliary states as part of the integration. While this allows explicit reversible schemes, it comes at the cost of stability, with both methods known to be unstable and prone to failure when integrating complex equations [95]. In addition to these stability concerns, the non-standard constructions of these schemes make it unclear how to generalise them to non-Euclidean spaces. To our knowledge, there are no efficient reversible numerical schemes on Lie groups, making such $\mathcal{O}(1)$ memory efficiency unattainable for manifold-valued NSDEs.

A potential solution to the difficulties of reversible solvers comes with the class of *Explicit and Effectively Symmetric (EES)* Runge–Kutta schemes [87]. EES schemes relax exact reversibility to reversibility within a controlled tolerance, giving stable explicit methods that are empirically indistinguishable from truly symmetric schemes while retaining stability comparable to classical schemes such as RK3 and RK4. Importantly, unlike auxiliary-state reversible solvers, EES schemes retain the structure of ordinary Runge–Kutta methods making them more suited to geometric integration. However, the existing formulation in Shmelev et al. [87] is limited to Euclidean ODEs: it does not treat stochastic dynamics, establish low-storage $2N$ realisations, or provide a construction capable of geometric integration over Lie groups.

Building on this prior work, we extend EES schemes to SDEs and derive Williamson $2N$ realisations, halving the constant-factor memory footprint of the resulting reversible solvers. We then use this $2N$ structure to construct a commutator-free lift to Lie groups. The resulting schemes, denoted $\mathrm{EES}_{\mathcal{R}}$ in Euclidean space and CF-EES on Lie groups, improve the stability of reversible Euclidean NSDE training and, to our knowledge, provide the first explicit near-reversible route to constant-memory training of manifold-valued NSDEs.

Concretely, we make three main contributions:

1. **Stable, reversible SDE integration.** We extend EES schemes to SDEs, obtaining explicit effectively symmetric integrators applicable to neural SDE training, with substantially better stability than existing explicit reversible solvers.

2. **Geometric reversible integration.** We show that EES schemes admit a low-memory $2N$ formulation which naturally lifts to a commutator-free Lie group integrator family, CF-EES. We then rigorously prove the convergence and order of these new SDE schemes.

3. **Practical training benefit.** Practically, the improved stability of EES schemes enhances the accuracy of Euclidean NSDEs; while CF-EES cuts the memory requirements of Lie-group-valued NSDEs by an order of magnitude, or more.

We release open-source implementations of the standard, low-memory, and Lie group EES schemes in JAX, and Julia to facilitate adoption by the broader research community.

The paper is organized as follows. Section 2 reviews reversible ODE solvers and their stability. Section 3 develops EES schemes for SDEs, including low-storage, adaptive, and manifold-valued variants, with convergence, stability, and backpropagation results. We compare our EES schemes against other reversible methods in Section 4, before concluding in Section 5.

2 Preliminaries

Existing reversible ODE and SDE solvers. The major drawback of classical reversible schemes is their low efficiency. It is well known that Runge–Kutta schemes are reversible only if they are implicit. More generally, symmetric parasitism-free general linear methods cannot be explicit [15]. A limited number of efficient reversible solvers have been proposed in the literature. The asynchronous leapfrog integrator (ALF) [98] for Neural ODEs overcomes the barrier of implicit schemes by tracking an additional state v as part of the integration. Reversible Heun takes a similar approach for SDEs of the form $dy_t = g(t, y_t)dt + f(t, y_t)dW_t$. Although efficient, requiring only one evaluation of the drift g and the diffusion f per step, Reversible Heun is known to be inherently unstable.

Theorem 2.1. [48, Theorem D.19] Suppose that the Reversible Heun method is used to obtain a solution $\{y_n, v_n\}_{n\geq 0}$ to the linear test ODE $dy = \lambda y\, dt$, where $\lambda \in \mathbb{C}$ and $y_0 \neq 0$. Then $\{y_n, v_n\}_{n\geq 0}$ is bounded if and only if $\lambda h \in [-i, i]$.

As remarked in [48], this domain is also the absolute stability region for the reversible asynchronous leapfrog integrator [98]. This instability has proven to be a significant bottleneck in certain practical applications [95, 60]. McCallum and Foster [60] proposed a method to transform any ODE integration method $y_{n+1} = y_n + \Psi_h(t, y_n)$ into one which is reversible, by coupling the action of the integrator with both positive and negative step sizes. The method offers a way of constructing reversible schemes with larger stability domains than those of the ALF and Reversible Heun integrators [60, Theorem 2.3]. However, the resulting stability domain of the transformed method is typically much smaller than that of the underlying method Ψ, and depends additionally on the coupling parameter. The McCallum-Foster methods were subsequently extended by [7] to *REX solvers* – a class of algebraically reversible exponential RK/SRK solvers specifically designed for diffusion-model inversion.

EES schemes for ODEs. EES schemes [87] are a class of explicit Runge–Kutta methods which offer an efficient approach to reversible integration without compromising on stability. Given positive integers $m \geq n$, an explicit Runge–Kutta scheme Φ_h is said to be an $\mathrm{EES}(n, m)$ scheme if Φ_h is of order n and $\Phi_{-h} \circ \Phi_h$ recovers the initial condition of the ODE up to order m. When m is large, such schemes exhibit near-reversible behaviour, which is often sufficient in practice. In [87], Butcher tableaux are derived for 3-stage $\mathrm{EES}(2, 5)$ and 4-stage $\mathrm{EES}(2, 7)$ schemes. Although the emphasis there is mostly on $\mathrm{EES}(2, 7)$, for our Neural SDE applications, we restrict attention to $\mathrm{EES}(2, 5)$ as the additional accuracy of $\mathrm{EES}(2, 7)$ does not justify the extra stage in the NSDE setting, as shown in Figure 9. Proposition 2.1 gives the general one-parameter family of $\mathrm{EES}(2, 5; x)$ tableaux.

Proposition 2.1 ([87, Proposition 8.4]). *For* $x \in \mathbb{R} \setminus \{1, \pm\frac{1}{2}\}$*, the 3-stage* $\mathrm{EES}(2,5;x)$ *Runge–Kutta scheme has Butcher tableau*

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1+2x}{4(1-x)} & \frac{1+2x}{4(1-x)} & & \\ \frac{3}{4(1-x)} & \frac{(4x-1)^2}{4(x-1)(1-4x^2)} & \frac{1-x}{1-4x^2} & \\ \hline & x & \frac{1}{2} & \frac{1}{2}-x \end{array}$$

The stability region for $\mathrm{EES}(2,5)$ is comparable to that of classical methods such as Kutta's RK4, but significantly larger than those of Reversible Heun and the MCF methods. Theorem 2.2 gives the exact form of this region for $\mathrm{EES}(2,5;x)$.

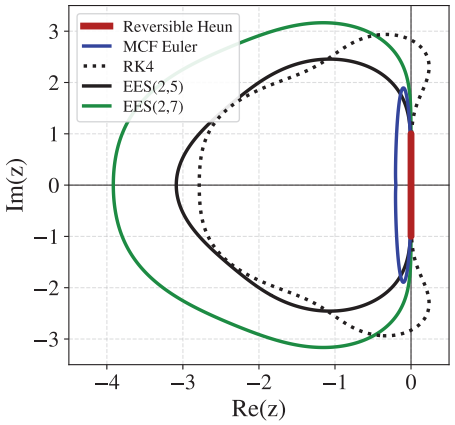


Figure 2: *Stability domains for* $\mathrm{EES}(2,5)$ *and* $\mathrm{EES}(2,7)$ *compared to RK4, MCF Euler, and Reversible Heun.*

Theorem 2.2. Suppose that, for $x \neq 1, \pm\frac{1}{2}$, $\mathrm{EES}(2,5;x)$ is used to obtain a solution $\{y_n\}_{n\geq 0}$ to the linear test equation $dy = \lambda y\, dt$, where $\lambda \in \mathbb{C}$ and $y_0 \neq 0$. Then $y_n \to 0$ as $n \to \infty$ if and only if

$$\left|1 + \rho + \frac{1}{2}\rho^2 + \frac{1}{8}\rho^3\right| < 1, \qquad \rho = \lambda h.$$

This follows by a direct computation of the stability function $R(\rho) = 1 + \rho + \frac{1}{2}\rho^2 + \frac{1}{8}\rho^3$, which is independent of x. Following Shmelev et al. [87, Section 8.1], we fix $x = 1/10$ to minimise leading error, and refer to $\mathrm{EES}(2,5;1/10)$ as *the* $\mathrm{EES}(2,5)$ scheme.

3 Explicit and Effectively Symmetric Schemes for SDEs on Lie Groups

This section develops the main technical contributions of the paper. We extend the Euclidean EES schemes of Shmelev et al. [87] from the ODE setting to SDEs, establish their mean-square stability, show that all $\mathrm{EES}(2,5;x)$ and $\mathrm{EES}(2,7;x)$ schemes admit an efficient Williamson $2N$-storage form, and lift them via Bazavov's commutator-free construction to a new family of effectively symmetric CF-EES integrators on arbitrary homogeneous spaces. To our knowledge, CF-EES is the first explicit (near-)reversible integrator in this setting, unlocking the reversible-adjoint backpropagation pipeline for manifold-valued neural SDEs.

EES schemes for SDEs. We apply EES schemes to SDEs in the canonical way. Given an SDE $\mathrm{d}y_t = f(y_t)dt + g(y_t) \circ \mathrm{d}W_t$, we rewrite the equation as being driven by $X_t := (t, W_t)$ with vector field (f, g) and apply the Runge–Kutta scheme with increments $\mathrm{d}X_t$ in place of h. For Brownian drivers, this gives the usual strong order $1/2$, and weak order 1. More generally, the same construction applies to differential equations driven by rough paths (RDEs) following [74], with the convergence rate governed by the driver regularity. For a detailed description of EES schemes applied to RDEs, we refer the reader to Appendix B. Convergence rates for EES schemes applied to RDEs (of which SDEs are a special case) are given in Appendix B.3, followed by convergence experiments in Appendix G.

Stability. As discussed in the introduction, EES schemes offer a stable alternative to reversible integration. While the stability of EES in the case of ODEs has been studied in Shmelev et al. [87] and Section 2, we are interested in the stability of EES when applied to stochastic drivers for our applications to Neural SDEs. To evaluate the stability in the context of SDEs, we consider the *mean-square stability*, which is widely used for analysis of stochastic integration methods in the literature [38, 25, 36, 53, 52, 73, 78, 79, 84]. Given the test equations $\mathrm{d}y_t = \lambda y_t \mathrm{d}t + \mu y_t \mathrm{d}W_t$, where $\lambda, \mu \in \mathbb{C}$ and $y_0 \neq 0$ almost surely, a solution $\{y_n\}_{n\geq 0}$ derived from a numerical integrator is said to be *mean-square stable* if $\lim_{n\to\infty} \mathbb{E}(|y_n|^2) = 0$. It follows in a similar fashion to Theorem 2.2 that $\mathrm{EES}(2,5;x)$ applied to this test equation is mean-square stable if and only if

$$\mathbb{E}\left[\left|1 + \rho + \frac{1}{2}\rho^2 + \frac{1}{8}\rho^3\right|^2\right] < 1,$$

where $\rho = \lambda \mathrm{d}t + \mu \mathrm{d}W_t \sim N(\lambda \mathrm{d}t, \mu^2 \mathrm{d}t)$. Figure 3 shows 4 cross-sections of the stability domain, compared to those of RK3 and RK4. Along most cross-sections, $\mathrm{EES}(2,5)$ achieves similar or greater stability than RK3 and RK4.

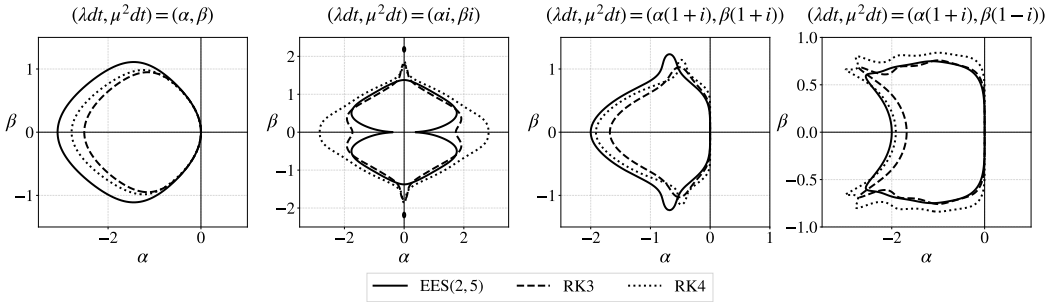


Figure 3: *Cross sections of the mean-square stability domains of* $\mathrm{EES}(2,5)$*, RK3 and RK4.*

A $2N$ realization of EES Schemes. The memory footprint of a classical Runge–Kutta step is proportional to the number of stages and thus becomes costly when the problem dimensionality is large. In [91], Williamson proposed a memory-efficient restructuring of the operations of a Runge–Kutta scheme into the following form

$$\begin{aligned} \Delta Y_i &= A_i \Delta Y_{i-1} + h\, f(Y_{i-1}), \\ Y_i &= Y_{i-1} + B_i \Delta Y_i, \end{aligned} \qquad i = 1, \dots, s. \tag{2}$$

Thus, for a state of size N, an s-stage implementation requires only $2N$ registers, compared with $(s+1)N$ for a standard explicit Runge–Kutta method. The conditions under which a Runge–Kutta scheme admits a Williamson 2N representation were formulated by Bazavov [4].

Theorem 3.1 ([4, Theorem 2]). An explicit Runge–Kutta method admits a Williamson $2N$ representation if and only if

$$a_{ij}(b_{j-1} - a_{j,j-1}) = (a_{i,j-1} - a_{j,j-1})b_j, \qquad i = 3, \dots, s, \quad j = 2, \dots, i-1. \tag{3}$$

In particular, a direct check of the conditions (3) for $\mathrm{EES}(2,5;x)$ and $\mathrm{EES}(2,7;x)$ shows that both of these classes admit the Williamson 2N form.

Proposition 3.1. $\mathrm{EES}(2,5;x)$ and $\mathrm{EES}(2,7;x)$ are Williamson 2N for any admissible parameter x.

This formulation reduces the memory requirements of $\mathrm{EES}(2,5)$ and $\mathrm{EES}(2,7)$ from $4N$ and $5N$, respectively, to $2N$. As we will see in the following sections, the Williamson 2N formulation not only significantly reduces the memory footprint of the scheme but also allows lifting the scheme to commutator-free methods over Lie groups, or, more generally, homogeneous spaces.

Homogeneous spaces. A *homogeneous space* is a smooth manifold $\mathcal{M}$ together with a transitive action $\Lambda\colon G \times \mathcal{M} \to \mathcal{M}$ of a Lie group G. Equivalently $\mathcal{M} \cong G/H$ for the isotropy subgroup $H \leq G$ at any chosen base point. We write $\mathfrak{g} = T_e G$ for the Lie algebra of G. A vector field F on $\mathcal{M}$ is represented through a state-dependent generator $\xi\colon \mathcal{M} \to \mathfrak{g}$ via the *fundamental vector field*

$$\xi(y)_{\mathcal{M}}(y) := \frac{\mathrm{d}}{\mathrm{d}t}\bigg|_{t=0} \Lambda\big(\exp(t\,\xi(y)),\, y\big), \qquad F(y) = \xi(y)_{\mathcal{M}}(y),$$

the infinitesimal version of the group action (see Appendix C.1).

Commutator-free methods. A natural strategy to integrate F on $\mathcal{M}$ is the Munthe-Kaas (RKMK) approach [67], which uses the exponential map to pull each step back to an equation on the Lie algebra and integrates it there with a classical Runge–Kutta scheme. However, achieving an order higher than two requires evaluating nested commutators of the stage generators, which is computationally expensive. *Crouch–Grossman (CG) methods* [21] avoid this by composing single-slope exponentials of the form $\exp(h\,\alpha_{ij} K_j)$, with $K_j = \xi(Y_{j-1}) \in \mathfrak{g}$, so that no Lie brackets appear. *Commutator-free (CF) methods* [19] generalise this by allowing each exponential's argument to be a real linear combination $\sum_j a_{l;ij} K_j$ of stage generators. More efficient CF variants further minimise the number of exponentials and storage required per step by reusing exponentials across stages [3]. Full descriptions of RKMK, CG, and CF methods are given in Appendices C.2–C.4.

Bazavov's $2N$ commutator-free lift. Bazavov [3] showed that any explicit Williamson $2N$ Runge–Kutta scheme can be lifted to a commutator-free method on a homogeneous space. The Euclidean update $Y_l = Y_{l-1} + B_l\delta_l$ is replaced by the action $Y_l = \Lambda(\exp(B_l\delta_l), Y_{l-1})$, while the increment recurrence $\delta_l = A_l\delta_{l-1} + hK_l$ on the Lie algebra is unchanged. Concretely, given a Williamson $2N$ scheme with coefficients $(A_l, B_l)_{l=1}^s$ with $A_1 = 0$, the update rule from $y_n \in \mathcal{M}$ to y_{n+1} is

$$\begin{aligned} Y_0 &:= y_n, \qquad \delta_0 := 0, \\ K_l &= \xi(Y_{l-1}), \\ \delta_l &= A_l\,\delta_{l-1} + h\,K_l, \\ Y_l &= \Lambda\big(\exp(B_l\,\delta_l),\, Y_{l-1}\big), \qquad l = 1, \dots, s, \end{aligned} \tag{4}$$

and $y_{n+1} \coloneqq Y_s$. Here $Y_l \in \mathcal{M}$ are the internal stage values and $\delta_l \in \mathfrak{g}$ is the current Lie-algebra increment; only these two quantities are stored at any time, preserving the two-register low-storage pattern. On a flat manifold ($\Lambda(\exp(v), y) = y + v$, $\xi(y) = f(y)$), the recurrence collapses to (2).

The CF-EES family. Since $\mathrm{EES}(2,5;x)$ and $\mathrm{EES}(2,7;x)$ are Williamson $2N$ for every admissible x (Proposition 3.1), Bazavov's lift (4) applies and produces the CF-$\mathrm{EES}(2,5;x)$ and CF-$\mathrm{EES}(2,7;x)$ schemes on any homogeneous space $\mathcal{M}$. Substituting the $2N$ coefficients of Section 3 into (4) gives the explicit recurrences with $s = 3$ and $s = 4$ stages respectively; the explicit reused-stage form of CF-$\mathrm{EES}(2,5;x)$ is recorded in Appendix E.1 and memory-compute optimality is shown in C.6.

Remark. We note that, among existing explicit reversible solvers, the above lift to a commutator-free method is unique to EES thanks to its Runge–Kutta form and Williamson 2N properties. As Reversible Heun and McCallum-Foster methods are not of Runge–Kutta form, they do not admit an analogous lift. A manifold extension of those schemes would require replacing affine state operations with non-canonical group operations.[1]

Order and reversibility of CF-EES on homogeneous spaces. The order and near-reversibility properties of the Euclidean EES schemes carry over to the manifold lift.

Theorem 3.2. For every admissible x and every homogeneous space $\mathcal{M} = G/H$, CF-$\mathrm{EES}(2,5;x)$ applied to an ODE on $\mathcal{M}$ is of order 2, and $\Phi_{-h} \circ \Phi_h$ recovers the initial condition up to order 5. The analogous statement holds for CF-$\mathrm{EES}(2,7;x)$ with recovery up to order 7.

The proof follows a symbolic computation of the Lie-Butcher series for CF-$\mathrm{EES}(2,5;x)$ and CF-$\mathrm{EES}(2,7;x)$ in terms of the free parameter x, which is compared to the Lie-Butcher series of the true solution to the ODE to extract the local order of the scheme. For the recovery of the initial condition, the LB series of $\Phi_{-h} \circ \Phi_h$ is computed similarly and compared to 0. For further details, the reader is referred to Appendix E.

Remark. The corresponding statement for the local order of the SDE schemes follows naturally via the formalism of rough path theory. The argument follows that of [74] for Euclidean spaces, and is detailed in Appendix F.

Backpropagation through EES and CF-EES. Algorithm 1 of Appendix B.4 describes backpropagation through the Euclidean EES integrator for general RDEs. The commutator-free analogue (Algorithm 2, Appendix F.3) differs mainly in that the adjoint evolves on the cotangent bundle.

4 Experiments

We evaluate the performance of CF-$\mathrm{EES}(2,5)$ on a range of problems covering Euclidean spaces, Lie groups and homogeneous spaces. Suitable reversible SDE baselines are limited in these settings. At the time of writing, Reversible Heun [48] is the only widely adopted explicit reversible solver. To broaden this comparison, we adapt the construction of McCallum and Foster [60] for reversible ODE solvers to the SDE versions of the Euler and Explicit Midpoint methods. Algorithm 1, together with the backpropagation procedure of McCallum and Foster [60], then enables efficient differentiation through the resulting schemes. The focus of our Euclidean experiments is the superior stability of EES schemes compared to these schemes.

[1] Affine combinations are not well-defined on a general manifold. On a Lie group, logarithmic coordinates supply a substitute, but only after fixing a base point or trivialisation.

Neither method admits a lift to Lie group integrators, and we are not aware of other explicit reversible schemes for Lie groups. Therefore, for problems on Lie groups or homogeneous spaces, we compare against non-reversible methods using memory-intensive full adjoints and emphasize the memory advantages of CF-EES over these schemes. We report metrics with two standard deviations, as well as runtimes and memory, in the Lie-group case. Appendix H contains additional experiments on unstable/stiff dynamics, finance, and molecular dynamics.

High volatility Ornstein–Uhlenbeck process. Consider learning the Ornstein–Uhlenbeck (OU) dynamics $\mathrm{d}y_t = \nu(\mu - y_t)\mathrm{d}t + \sigma \mathrm{d}W_t, y_0 \in \mathbb{R}$, under a high-volatility regime $\sigma \gg 0$. Specifically, we take $\nu = 0.2, \mu = 0.1$ and $\sigma = 2$. Motivated by Oh et al. [69], we take a Neural Langevin SDE (LSDE) defined by

$$\mathrm{d}z_t = g(z_t; \theta_g)\mathrm{d}t + f(t; \theta_f) \circ \mathrm{d}W_t,$$

with $z_0 = h(\mathbf{x}, \theta_h) \in \mathbb{R}^{d_z}$, where h is a learnable affine function of the input data $\mathbf{x} = \{x_n\}_{n \geq 0}$, $x_n \in \mathbb{R}^2$, sampled from the true OU dynamics, and g, f are neural networks parametrised by θ_g, θ_f respectively. Architecture, training schedule, and loss are detailed in Appendix I.2.

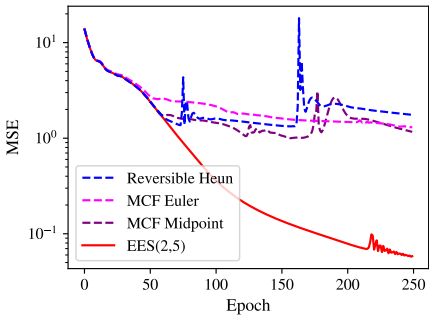


Figure 4: *Training MSE for OU dynamics with a fixed f, g evaluation count.*

Figure 4 shows the training loss using Reversible Heun, McCallum-Foster (MCF) methods, and $\mathrm{EES}(2,5)$, with the step size chosen such that the number of evaluations of f, g is fixed between solvers. Such a choice yields comparable runtimes across all solvers, enabling a fair comparison. Table 1 gives the number of evaluations of f, g per step of the solvers, the chosen step size, the terminal MSE, and the total runtime of each solver. From Figure 4, we see that for the initial ~ 50 epochs, the methods perform similarly. After this, $\mathrm{EES}(2,5)$ significantly outperforms the other methods, suggesting the model has begun to learn high-volatility dynamics which cause instability in the Reversible Heun and McCallum-Foster methods.

Table 1: Metrics for OU dynamics. The step size is chosen such that the total number of evaluations of f, g per integration is fixed.

Method	# Eval. / Step	Step Size	Terminal MSE	Runtime (s)
Reversible Heun	1	$1/12$	1.02	368.2
MCF Euler	2	$1/6$	1.30	307.5
MCF Midpoint	4	$1/3$	1.17	279.5
$\mathrm{EES}(2,5)$	3	$1/4$	0.05	261.3

Stochastic Volatility. Following the empirical observations of Gatheral et al. [29], rough volatility models posit that log-volatility behaves as a rough fractional process, with Hurst parameter $H \ll \frac{1}{2}$. In direct discretisations of the rough driver, reduced regularity sharply increases the timestep budget required to attain a target error ε, from ε^{-2} in the Brownian case to $\varepsilon^{-6.25}$ when $H = 0.33$ [28]. While finite-dimensional Markovian lifts can mitigate this, they do so at the cost of a substantially enlarged state space. This makes rough volatility a natural stress test for reversible solvers: long trajectories are required, and the benefit of $\mathcal{O}(1)$ adjoint memory grows with more timesteps.

We therefore train a neural SDE with the same architecture and hyperparameters as in (23) on rough Bergomi dynamics using a signature kernel score-matching objective [41]. We use a generous fixed total evaluation budget to ensure good path fidelity and to reveal the runtime advantages of the $2N$ recurrence at high evaluation counts. Table 2 shows that, in this regime, all methods attain comparable terminal MSE, while $\mathrm{EES}(2,5)$ is the fastest by a clear margin. Thus, in the long-horizon setting, $\mathrm{EES}(2,5)$ preserves the accuracy of the reversible baselines while achieving the most favourable runtime among the reversible methods considered. We present additional results in Appendix H.2 for Black-Scholes, classical Bergomi, a local stochastic volatility model, the Heston model, rough Heston model, and quadratic rough Heston model.

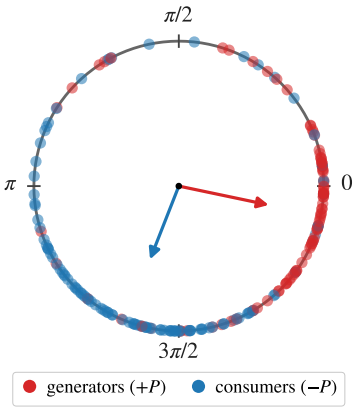
Table 2: Metrics for rough Bergomi dynamics. The step size is chosen such that the total number of evaluations of f, g per integration is fixed.

Method	# Eval. / Step	Step Size	Terminal MSE	Runtime (s)
Reversible Heun	1	$1/504$	$7.81_{\pm 1.06}$	999.3
MCF Euler	2	$1/252$	$7.81_{\pm 1.06}$	928.9
MCF Midpoint	4	$1/126$	$7.81_{\pm 1.06}$	519.4
EES$(2,5)$	3	$1/168$	$7.81_{\pm 1.06}$	405.4

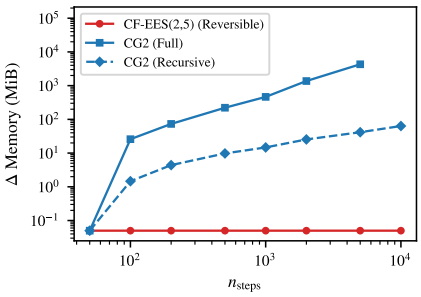
Stochastic Kuramoto network on $T\mathbb{T}^N$. The Kuramoto model [1] describes a network of N coupled oscillators with phases $\theta_i \in S^1$ that interact through their pairwise differences. The model captures synchronisation phenomena across a wide range of physical and biological systems, such as circadian rhythms [58], neuronal oscillations [51, 10], and the rotor-angle dynamics of synchronous machines on a power grid [26]. Stochastic variants [83, 82] add the ability to model noise in the system. We use the second-order stochastic form of Olmi and Torcini [70, eq. (1) with $K_2 = 0$],

$$m\,\ddot{\theta}_i \;=\; -\dot{\theta}_i \;+\; \Omega_i \;+\; \tfrac{K}{N}\textstyle\sum_j \sin(\theta_j - \theta_i) \;+\; \xi_i(t),$$
$$\langle \xi_i(t)\xi_j(s)\rangle = 2D\,\delta_{ij}\delta(t-s). \tag{5}$$

with bimodal natural frequencies $\Omega_i \in \{+P, -P\}$ as in the power-grid generator/consumer split of Filatrella et al. [26]. Equation (5) evolves on the product Lie group $T\mathbb{T}^N \cong \mathbb{T}^N \times \mathbb{R}^N$. We train a neural SDE on $T\mathbb{T}^N$, with MLP drift and diffusion fields whose inputs are the periodic encoding $(\sin\theta, \cos\theta, \omega) \in \mathbb{R}^{3N}$ and whose outputs lie in the Lie algebra $\mathbb{R}^{2N}$. We train against synthetic trajectories of (5) in the partial-synchronisation regime ($K = 2$, $P = 0.5$, $D = 0.05$), using a multi-horizon wrapped energy score [30]. Figure 5b shows the observed memory complexity of the training using CF-EES$(2,5)$ with the reversible adjoint and CG2 with the recursive and full adjoints. Table 3 reports the test energy score of each method. CF-EES$(2,5)$ attains a test energy score within roughly one standard deviation of the CG2 baselines whilst maintaining $\mathcal{O}(1)$ memory complexity. Architecture, training, and adjoint diagnostics are in Appendix I.5.



(a) 200 Kuramoto oscillators on $\mathbb{T}^{200}$ at partial synchronisation. At full synchronisation, the arrows would be superimposed.



(b) Memory complexity of CFEES and CG2 on the stochastic Kuramoto problem on $T\mathbb{T}^{1000}$ using different adjoints.

Table 3: Test energy score on the stochastic Kuramoto problem. Step sizes are chosen so that all methods use the same number of vector-field evaluations per integration.

Method	Adjoint	#Eval. / Step	Step size	Test ES ↓	Runtime (s)
CG2	Full	2	$1/75$	$370.30_{\pm 0.78}$	2,988
CG2	Recursive	2	$1/75$	$370.27_{\pm 0.76}$	3,994
CF-EES$(2,5)$	Reversible	3	$1/50$	$392.77_{\pm 15.55}$	2,892

Latent SDE on the sphere. We reproduce the experiment of Zeng et al. [94] and train a variational latent SDE on the unit sphere S^{n-1}, viewed as the homogeneous space $\mathrm{SO}(n)/\mathrm{SO}(n-1)$, for human-activity classification on the UCI Human Activity benchmark [76]. This dataset comprises 12-dimensional time series of wearable motion sensor readings, each labelled with one of seven activity classes at each time point. An mTAN attention encoder summarises the input sequence into

a fixed-length context vector h that conditions both the initial-state distribution $q(z_0 \mid h)$ and the time-varying drift of a latent SDE on S^{15}. An MLP decoder maps each resulting latent state z_t back to a Gaussian likelihood over the observed sensors, and a per-timepoint head predicts the activity class from z_t. Zeng et al. [94] use the geometric Euler-Maruyama integrator (Geo E-M), with the discretise-then-optimise (full) adjoint. We instead test the performance of CF-EES$(2,5)$ with the reversible adjoint and also examine a stochastic RKMK (SRKMK) [65] form of ShARK [27] with strong order 1 to see whether improved stochastic order aids learning.

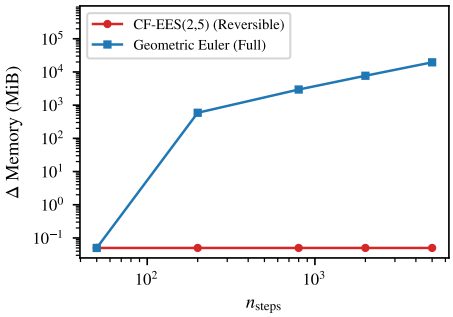


Figure 6: Memory complexity of CFEES and Geo E-M on the UCI Human Activity problem on S^{15} using different adjoints.

Figure 6 shows the observed peak memory requirement of one forward and backward pass of the model as a function of the number of steps taken by the latent SDE. As expected, the memory requirement of Geometric Euler-Maruyama grows linearly in the number of steps, while CF-EES$(2,5)$ with the reversible adjoint retains $\mathcal{O}(1)$ memory complexity. At the same network-evaluation budget, SRKMK ShARK gives only a marginal accuracy improvement over CF-EES$(2,5)$, despite its higher nominal stochastic order. This suggests that the lower strong order of CF-EES$(2,5)$ is not a limiting factor in latent SDE learning.

Table 4: UCI Human Activity classification accuracy. The step size is chosen to keep the total number of NN evaluations per integration fixed.

Method	Adjoint	#Eval. / Step	Step Size	Test accuracy (%)	Runtime (s)
Geo E-M [94]	Full	1	1/30	$86.25_{\pm 0.76}$	366.3
CG2	Full	2	1/15	$88.17_{\pm 1.23}$	734.6
CF-EES$(2,5)$	Reversible	3	1/10	$88.30_{\pm 0.37}$	405.5
SRKMK ShARK	Full	3	1/10	$88.66_{\pm 0.59}$	657.2

5 Conclusions, limitations and future work

In this paper, we have extended Explicit and Effectively Symmetric (EES) schemes [87] to SDEs using the rough RK framework of Redmann and Riedel [74]. We have shown that the resulting schemes are stable via mean-square stability analysis, and that the resulting integrators admit clean generalisation to homogeneous spaces while retaining their $2N$ memory-optimality. These developments enable more stable training of Euclidean neural SDEs and unlock stable, constant-memory training of manifold neural SDEs, with applications in finance, biology, and robotics.

Limitations. Our constructions are fixed-step, and adaptive step sizing is nontrivial: step rejection requires restoring the previous state, which is incompatible with the two-register reversible implementation and would require a $3S^*$ reformulation [44]. Additionally, on homogeneous spaces, local error estimates must be compared in a common linear space, e.g. via logarithmic maps or tangent-space trivialisations [40, Chapter. 10]. A second structural limitation is that CF-EES relies intrinsically on a homogeneous-space representation with a transitive group action and tractable exponential map, and so does not apply to general Riemannian manifolds lacking such algebraic structure. Recent work [11] suggests that such a generalisation is possible, albeit with different technical machinery.

Future directions. There are several potential extensions to this paper that are left for future research. Applications of EES schemes to more complicated models, including but not limited to Neural Jump SDEs [42, 37], Neural CDEs [47] may be of interest, as would further examination of EES for accelerating Neural RDEs [63]. An extension of EES schemes to include partitioned or adaptive step-size schemes would be valuable for training stiff neural differential equations. Further extension of our rough RK construction to higher levels of the signature suggests the possibility of a log-ODE-free neural RDE, escaping the primary cost of such methods [75].

Reproducibility. All experiments are available at this anonymised GitHub repository, and the three released packages can be found at the links provided in Table 10.

References

[1] Juan A. Acebrón, Luis L. Bonilla, Conrad J. Pérez Vicente, Félix Ritort, and Renato Spigler. The Kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of Modern Physics*, 77(1):137–185, 2005.

[2] Lennart Bastian, Mohammad Rashed, Nassir Navab, and Tolga Birdal. Continuous-Time SO(3) Forecasting with Savitzky–Golay Neural Controlled Differential Equations, June 2025. URL http://arxiv.org/abs/2506.06780. arXiv:2506.06780 [cs].

[3] Alexei Bazavov. Commutator-free Lie group methods with minimum storage requirements and reuse of exponentials. *BIT Numerical Mathematics*, 62(3), 2022. doi: 10.1007/s10543-021-00892-x. arXiv:2007.04225.

[4] Alexei Bazavov. 2N-storage Runge-Kutta methods: Order conditions, general properties and some analytic solutions, June 2025. URL http://arxiv.org/abs/2506.07359. arXiv:2506.07359 [math].

[5] Mikkel Bennedsen, Asger Lunde, and Mikko S. Pakkanen. Hybrid scheme for Brownian semistationary processes. *Finance and Stochastics*, 21(4):931–965, October 2017. ISSN 1432-1122. doi: 10.1007/s00780-017-0335-5. URL http://dx.doi.org/10.1007/s00780-017-0335-5.

[6] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, May 1973. ISSN 1537-534X. doi: 10.1086/260062. URL http://dx.doi.org/10.1086/260062.

[7] Zander W. Blasingame and Chen Liu. Rex: Reversible Solvers for Diffusion Models, 2025.

[8] Ofelia Bonesini, Emilio Ferrucci, Ioannis Gasteratos, and Antoine Jacquier. Rough differential equations for volatility, 2024. URL https://arxiv.org/abs/2412.21192. _eprint: 2412.21192.

[9] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

[10] Michael Breakspear, Stewart Heitmann, and Andreas Daffertshofer. Generative models of cortical oscillations: Neurobiological implications of the kuramoto model. *Frontiers in Human Neuroscience*, 4, 2010. ISSN 1662-5161. doi: 10.3389/fnhum.2010.00190. URL http://dx.doi.org/10.3389/fnhum.2010.00190.

[11] Eugen Bronasco, Adrien Busnot Laurent, and Baptiste Huguet. High order integration of stochastic dynamics on Riemannian manifolds with frozen flow methods, June 2025. URL http://arxiv.org/abs/2503.21855. arXiv:2503.21855 [math].

[12] Kevin Burrage and Pamela M Burrage. Order conditions of stochastic runge–kutta methods by b-series. *SIAM Journal on Numerical Analysis*, 38(5):1626–1646, 2000.

[13] Kevin Burrage and Pamela Marion Burrage. High strong order explicit runge-kutta methods for stochastic ordinary differential equations. *Applied Numerical Mathematics*, 22(1-3):81–101, 1996.

[14] Kevin Burrage and PM Burrage. General order conditions for stochastic runge-kutta methods for both commuting and non-commuting stochastic ordinary differential equation systems. *Applied Numerical Mathematics*, 28(2-4):161–177, 1998.

[15] JC Butcher, AT Hill, and TJT Norton. Symmetric general linear methods. *BIT Numerical Mathematics*, 56:1189–1212, 2016.

[16] John C Butcher. *B-series: algebraic analysis of numerical methods*, volume 55. Springer, 2021.

[17] John C Butcher, Taketomo Mitsui, Yuto Miyatake, and Shun Sato. On the B-series composition theorem. *arXiv preprint arXiv:2409.08533*, 2024.

[18] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[19] Elena Celledoni, Arne Marthinsen, and Brynjulf Owren. Commutator-free Lie group methods. *Future Generation Computer Systems*, 19(3):341–352, April 2003. ISSN 0167-739X. doi: 10.1016/S0167-739X(02)00161-9. URL https://www.sciencedirect.com/science/article/pii/S0167739X02001619.

[20] Alain Connes and Dirk Kreimer. Hopf algebras, renormalization and noncommutative geometry. In *Quantum field theory: perspective and prospective*, pages 59–109. Springer, 1999.

[21] P. E. Crouch and R. Grossman. Numerical integration of ordinary differential equations on manifolds. *Journal of Nonlinear Science*, 3(1):1–33, December 1993. ISSN 1432-1467. doi: 10.1007/BF02429858. URL https://doi.org/10.1007/BF02429858.

[22] Charles Curry, Kurusch Ebrahimi-Fard, Dominique Manchon, and Hans Z. Munthe-Kaas. Planarly branched rough paths and rough differential equations on homogeneous spaces. *Journal of Differential Equations*, 269:9740–9782, 2020. doi: 10.1016/j.jde.2020.06.058. arXiv:1804.08515.

[23] Aurélien Deya, Andreas Neuenkirch, and Samy Tindel. A milstein-type scheme without lévy area terms for sdes driven by fractional brownian motion. In *Annales de l'IHP Probabilités et statistiques*, volume 48, pages 518–550, 2012.

[24] Allan dos Santos Costa, Ilan Mitnikov, Franco Pellegrini, Ameya Daigavane, Mario Geiger, Zhonglin Cao, Karsten Kreis, Tess Smidt, Emine Kucukbenli, and Joseph Jacobson. EquiJump: Protein dynamics simulation via SO(3)-equivariant stochastic interpolants, 2024. URL https://arxiv.org/abs/2410.09667. arXiv: 2410.09667 [cs.LG].

[25] PD Drummond and IK Mortimer. Computer simulations of multiplicative stochastic differential equations. *Journal of computational physics*, 93(1):144–170, 1991.

[26] Giovanni Filatrella, Arne Hejde Nielsen, and Niels Falsig Pedersen. Analysis of a power grid using a Kuramoto-like model. *The European Physical Journal B*, 61(4):485–491, 2008.

[27] James Foster, Goncalo dos Reis, and Calum Strange. High order splitting methods for SDEs satisfying a commutativity condition. *arXiv:2210.17543*, 2023.

[28] Peter Friz and Sebastian Riedel. Convergence rates for the full gaussian rough paths. In *Annales de l'IHP Probabilités et statistiques*, volume 50, pages 154–194, 2014.

[29] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative Finance*, 18(6):933–949, 2018. doi: 10.1080/14697688.2017.1393551. arXiv:1410.3394.

[30] Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[31] Massimiliano Gubinelli. Ramification of rough paths. *Journal of Differential Equations*, 248 (4):693–721, 2010.

[32] E. Hairer and G. Wanner. On the Butcher group and general multi-value methods. *Computing (Arch. Elektron. Rechnen)*, 13(1):1–15, 1974. ISSN 0010-485X,1436-5057. doi: 10.1007/bf02268387. URL https://doi.org/10.1007/bf02268387.

[33] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer, 2 edition, 2006.

[34] Martin Hairer and David Kelly. Geometric versus non-geometric rough paths. In *Annales de l'IHP Probabilités et statistiques*, volume 51, pages 207–251, 2015.

[35] Bin Han and Kuang Yu. Refining potential energy surface through dynamical properties via differentiable molecular simulation. *Nature Communications*, 16(1):816, January 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-56061-z. URL https://www.nature.com/articles/s41467-025-56061-z.

[36] Diego Bricio Hernandez and Renato Spigler. Convergence and stability of implicit runge-kutta methods for systems with multiplicative noise. *BIT Numerical Mathematics*, 33(4):654–669, 1993.

[37] Calypso Herrera, Florian Krach, and Josef Teichmann. Neural jump ordinary differential equations: Consistent continuous-time prediction and filtering. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2006.04727.

[38] Desmond J Higham. Mean-square and asymptotic stability of the stochastic theta method. *SIAM journal on numerical analysis*, 38(3):753–769, 2000.

[39] Michael E. Hoffman. Combinatorics of rooted trees and Hopf algebras. *Trans. Amer. Math. Soc.*, 355(9):3795–3811, 2003. ISSN 0002-9947,1088-6850.

[40] Arieh Iserles, Hans Z. Munthe-Kaas, Syvert P. Nørsett, and Antonella Zanna. Lie-group methods. *Acta Numerica*, 9:215–365, 2000. doi: 10.1017/S0962492900002154.

[41] Zacharia Issa, Blanka Horvath, Maud Lemercier, and Cristopher Salvi. Non-adversarial training of neural sdes with signature kernel scores. *Advances in Neural Information Processing Systems*, 36:11102–11126, 2023.

[42] Junteng Jia and Austin R Benson. Neural jump stochastic differential equations. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] Kasper Johansson, Mehmet G. Ogut, Markus Pelger, Thomas Schmelzer, and Stephen Boyd. A Simple Method for Predicting Covariance Matrices of Financial Returns. *Foundations and Trends in Econometrics*, 12(4):324–407, November 2023. ISSN 1551-3076, 1551-3084. doi: 10.1561/0800000047. URL https://www.emerald.com/fteco/article/12/4/324/1320852/A-Simple-Method-for-Predicting-Covariance-Matrices.

[44] David I. Ketcheson. Runge–Kutta methods with minimum storage implementations. *Journal of Computational Physics*, 229(5):1763–1773, 2010. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2009.11.006. URL https://www.sciencedirect.com/science/article/pii/S0021999109006251.

[45] Patrick Kidger. *On Neural Differential Equations*. PhD Thesis, University of Oxford, 2021.

[46] Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[47] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33: 6696–6707, 2020.

[48] Patrick Kidger, James Foster, Xuechen Li, and Terry Lyons. Efficient and accurate gradients for neural SDEs. *Advances in Neural Information Processing Systems*, 34:18747–18761, 2021.

[49] Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural SDEs as infinite-dimensional GANs. In *International conference on machine learning*, pages 5453–5463. PMLR, 2021.

[50] Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20(31):1–45, 2019.

[51] Manfred G. Kitzbichler, Marie L. Smith, Søren R. Christensen, and Ed Bullmore. Broadband criticality of human brain network synchronization. *PLoS Computational Biology*, 5 (3):e1000314, March 2009. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000314. URL http://dx.doi.org/10.1371/journal.pcbi.1000314.

[52] Y Komori, Y Saito, and T Mitsui. Some issues in discrete approximate solution for stochastic differential equations. *Computers & Mathematics with Applications*, 28(10-12):269–278, 1994.

[53] Yoshio Komori and Taketomo Mitsui. Stable row-type weak scheme for stochastic differential equations. *Monte Carlo Methods and Applications*, 1:279–300, 01 1995.

[54] Maud Lemercier, Cristopher Salvi, Theodoros Damoulas, Edwin Bonilla, and Terry Lyons. Distribution regression for sequential data. In *International Conference on Artificial Intelligence and Statistics*, pages 3754–3762. PMLR, 2021.

[55] Maud Lemercier, Terry Lyons, and Cristopher Salvi. Log-pde methods for rough signature kernels. *arXiv preprint arXiv:2404.02926*, 2024.

[56] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David K Duvenaud. Scalable gradients and variational inference for stochastic differential equations. In *Symposium on Advances in Approximate Bayesian Inference*, pages 1–28. PMLR, 2020.

[57] Haitao Lin, Odin Zhang, Huifeng Zhao, Dejun Jiang, Lirong Wu, Zicheng Liu, Yufei Huang, and Stan Z. Li. PPFlow: Target-aware Peptide Design with Torsional Flow Matching. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2405.06642.

[58] Zhixin Lu, Kevin Klein-Cardeña, Steven Lee, Thomas M. Antonsen, Michelle Girvan, and Edward Ott. Resynchronization of circadian oscillators and the east-west asymmetry of jet-lag. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(9), July 2016. ISSN 1089-7682. doi: 10.1063/1.4954275. URL http://dx.doi.org/10.1063/1.4954275.

[59] Dominique Manchon. Hopf algebras, from basics to applications to renormalization. In *Comptes-rendus des Rencontres mathématiques de Glanon 2001*. 2003. arXiv:math/0408405.

[60] Sam McCallum and James Foster. Efficient, accurate and stable gradients for neural ODEs. *arXiv preprint arXiv:2410.11648*, 2024.

[61] Robert I McLachlan, Klas Modin, Hans Munthe-Kaas, and Olivier Verdier. Butcher series: a story of rooted trees and numerical methods for evolution equations. *Asia Pacific Mathematics Newsletter*, 7(1):1–11, 2017. arXiv:1512.00906.

[62] Steven Morad. Cyreal, January 2026. URL https://github.com/smorad/cyreal. original-date: 2025-12-15T05:39:30Z.

[63] James Morrill, Cristopher Salvi, Patrick Kidger, James Foster, and Terry Lyons. Neural rough differential equations for long time series. In *International Conference on Machine Learning*, pages 7829–7838. PMLR, 2021.

[64] Nicola Muça Cirone and Cristopher Salvi. Rough kernel hedging, 2025.

[65] Michelle Muniz, Matthias Ehrhardt, Michael Günther, and Renate Winkler. Strong stochastic Runge-Kutta–Munthe-Kaas methods for nonlinear Itô SDEs on manifolds. *Applied Numerical Mathematics*, 193:196–203, November 2023. ISSN 0168-9274. doi: 10.1016/j.apnum.2023.07.024. URL http://dx.doi.org/10.1016/j.apnum.2023.07.024.

[66] H. Z. Munthe-Kaas and W. M. Wright. On the Hopf algebraic structure of Lie group integrators. *Foundations of Computational Mathematics*, 8(2):227–257, 2008. doi: 10.1007/s10208-006-0222-5.

[67] Hans Munthe-Kaas. Runge-Kutta methods on Lie groups. *BIT Numerical Mathematics*, 38(1):92–111, March 1998. ISSN 1572-9125. doi: 10.1007/BF02510919. URL https://doi.org/10.1007/BF02510919.

[68] Diaa Noureldin, Neil Shephard, and Kevin Sheppard. Multivariate high-frequency-based volatility (HEAVY) models. *Journal of Applied Econometrics*, 27(6):907–933, September 2012. ISSN 0883-7252, 1099-1255. doi: 10.1002/jae.1260. URL https://onlinelibrary.wiley.com/doi/10.1002/jae.1260.

[69] YongKyung Oh, Dong-Young Lim, and Sungil Kim. Stable neural stochastic differential equations in analyzing irregular time series data. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2402.14989.

[70] Simona Olmi and Alessandro Torcini. Stochastic Kuramoto oscillators with inertia and higher-order interactions. *Physical Review E*, 111:L012202, 2025. doi: 10.1103/PhysRevE.111.L012202. arXiv:2407.14874.

[71] Brynjulf Owren. Order conditions for commutator-free Lie group methods. *Journal of Physics A: Mathematical and General*, 39(19):5585–5599, 2006. doi: 10.1088/0305-4470/39/19/S15.

[72] Alexandre Pannier and Cristopher Salvi. A path-dependent pde solver based on signature kernels, 2024.

[73] WP Petersen. A general implicit splitting for stabilizing numerical simulations of itô stochastic differential equations. *SIAM journal on numerical analysis*, 35(4):1439–1451, 1998.

[74] Martin Redmann and Sebastian Riedel. Runge-kutta methods for rough differential equations. *arXiv preprint arXiv:2003.12626*, 2020. Published in J. Stoch. Anal. 3(4):6, 2022.

[75] Martin Redmann and Justus Werner. State-of-the-Art Numerical Schemes for Solving Rough Differential Equations. In *Fractional S(P)DEs*, pages 1–24. WORLD SCIENTIFIC, October 2024. ISBN 978-981-98-0209-8. doi: 10.1142/9789819802104_0001. URL https://www.worldscientific.com/doi/10.1142/9789819802104_0001.

[76] Jorge Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto, and Xavier Parra. Human activity recognition using smartphones, 2013. tex.howpublished: UCI Machine Learning Repository.

[77] Callum Rough. The Rough Bergomi Model: From Motivation to Implementation. Master's thesis, Imperial College London, London, April 2023. URL https://www.imperial.ac.uk/media/imperial-college/faculty-of-natural-sciences/department-of-mathematics/math-finance/212261203---Callum-Rough---ROUGH_CALLUM_01333836.pdf.

[78] Yoshihiro Saito and Taketomo Mitsui. T-stability of numerical scheme for stochastic differential equations. In *Contributions in numerical mathematics*, pages 333–344. World Scientific, 1993.

[79] Yoshihiro Saito and Taketomo Mitsui. Stability analysis of numerical schemes for stochastic differential equations. *SIAM Journal on Numerical Analysis*, 33(6):2254–2267, 1996.

[80] Cristopher Salvi, Thomas Cass, James Foster, Terry Lyons, and Weixin Yang. The signature kernel is the solution of a goursat pde. *SIAM Journal on Mathematics of Data Science*, 3(3): 873–899, 2021.

[81] Cristopher Salvi, Maud Lemercier, Chong Liu, Blanka Horvath, Theodoros Damoulas, and Terry Lyons. Higher order kernel mean embeddings to capture filtrations of stochastic processes. *Advances in Neural Information Processing Systems*, 34:16635–16647, 2021.

[82] Benjamin Schäfer, Dirk Witthaut, Marc Timme, and Vito Latora. Dynamically induced cascading failures in power grids. *Nature Communications*, 9(1):1975, 2018.

[83] K. Schmietendorf, J. Peinke, R. Friedrich, and O. Kamps. Self-organized synchronization and voltage stability in networks of synchronous machines. *The European Physical Journal Special Topics*, 223(12):2577–2592, 2014.

[84] Henri Schurz. Asymptotical mean square stability of an equilibrium point of some linear numerical solutions with multiplicative noise. *Stochastic Analysis and Applications*, 14(3): 313–353, 1996.

[85] Daniil Shmelev. kauri: symbolic computation with rooted trees, Hopf algebras and Lie group integrators, 2026. Python package, available at https://github.com/daniil-shmelev/kauri.

[86] Daniil Shmelev and Cristopher Salvi. pySigLib – Fast Signature-Based Computations on CPU and GPU, September 2025. URL http://arxiv.org/abs/2509.10613. arXiv:2509.10613 [cs].

[87] Daniil Shmelev, Kurusch Ebrahimi-Fard, Nikolas Tapia, and Cristopher Salvi. Explicit and effectively symmetric runge-kutta methods. *arXiv preprint arXiv:2507.21006*, 2025.

[88] Philipp Stumm and Andrea Walther. New algorithms for optimal online checkpointing. *SIAM Journal on Scientific Computing*, 32(2):836–854, 2010. doi: 10.1137/080742439.

[89] Ch Tsitouras. Runge–Kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62(2):770–775, 2011.

[90] Vedrana Vidulin and Mitja Lustrek. Localization data for person activity, 2010. URL https://archive.ics.uci.edu/dataset/196.

[91] J. H Williamson. Low-storage Runge-Kutta schemes. *Journal of Computational Physics*, 35(1):48–56, March 1980. ISSN 0021-9991. doi: 10.1016/0021-9991(80)90033-9. URL https://www.sciencedirect.com/science/article/pii/0021999180900339.

[92] Yannik P. Wotte, Federico Califano, and Stefano Stramigioli. Geometric Neural Ordinary Differential Equations: From Manifolds to Lie Groups. *Entropy*, 27(8), 2025. ISSN 1099-4300. doi: 10.3390/e27080878. URL https://www.mdpi.com/1099-4300/27/8/878.

[93] Antonella Zanna, Kenth Engø, and Hans Z. Munthe-Kaas. Adjoint and selfadjoint lie-group methods. *BIT Numerical Mathematics*, 41(2):395–421, March 2001. ISSN 1572-9125. doi: 10.1023/a:1021950708869. URL http://dx.doi.org/10.1023/A:1021950708869.

[94] Sebastian Zeng, Florian Graf, and Roland Kwitt. Latent SDEs on homogeneous spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2306.16248.

[95] Qinsheng Zhang and Yongxin Chen. Path integral sampler: a stochastic control approach for sampling. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2111.15141.

[96] Yaolong Zhang, Ce Hu, and Bin Jiang. Embedded Atom Neural Network Potentials: Efficient and Accurate Machine Learning with a Physically Inspired Representation. *The Journal of Physical Chemistry Letters*, 10(17):4962–4967, September 2019. doi: 10.1021/acs.jpclett.9b02037. URL https://doi.org/10.1021/acs.jpclett.9b02037.

[97] Yuchen Zhu, Tianrong Chen, Lingkai Kong, Evangelos A. Theodorou, and Molei Tao. Trivialized Momentum Facilitates Diffusion Generative Modeling on Lie Groups. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2405.16381.

[98] Juntang Zhuang, Nicha C Dvornek, Sekhar Tatikonda, and James S Duncan. MALI: A memory efficient and reverse accurate integrator for neural ODEs. *arXiv preprint arXiv:2102.04668*, 2021.

A Algebraic Background

A.1 The Connes-Kreimer Hopf Algebra

We give a brief account of non-planar (labelled) rooted trees and the Connes-Kreimer Hopf algebra. We refer the reader to [39] for a comprehensive presentation. A non-planar labelled rooted tree is defined as a graph $\tau = (V, E, r)$ with vertex set V, edge set E and a root vertex $r \in V$, together with a set of vertex decorations drawn from $\{1, \dots, d\}$. We denote the empty tree by $\emptyset$. Given trees $\tau_1, \dots, \tau_m$, we write $[\tau_1, \dots, \tau_m]_a$ to denote the tree formed by connecting the root vertices of $\tau_1, \dots, \tau_m$ to a new root, which receives the label $a \in \{1, \dots, d\}$. Non-planarity means tree order in $[\tau_1, \dots, \tau_m]_a$ is irrelevant. Repeated trees will be denoted using power notation, for instance

$$[\tau_1, \tau_1, \tau_2, \tau_3, \tau_3, \tau_3]_a = [\tau_1^2, \tau_2, \tau_3^3]_a.$$

We write $|\tau|$ to denote the number of vertices in a tree. Additionally, we define the following combinatorial quantities, defined on unlabelled trees:

$$\emptyset! = 1, \quad \bullet! = 1, \quad [\tau_1, \dots, \tau_m]! = |[\tau_1, \dots, \tau_m]| \prod_{i=1}^{m} \tau_i!,$$

$$\sigma(\emptyset) = 1, \quad \sigma(\bullet) = 1, \quad \sigma([\tau_1^{k_1}, \dots, \tau_m^{k_m}]) = \prod_{i=1}^{m} k_i! \sigma(\tau_i)^{k_i},$$

$$\beta(\emptyset) = 1, \quad \beta(\bullet) = 1, \quad \beta([\tau_1^{k_1}, \dots, \tau_m^{k_m}]) = \binom{[\tau_1^{k_1}, \dots, \tau_m^{k_m}]}{|\tau_1|, \dots, |\tau_m|} \prod_{i=1}^{m} \frac{1}{k_i!} \beta(\tau_i)^{k_i}.$$

We will refer to the commutative juxtaposition of trees as a forest. We write $\mathcal{T}$ to denote the set of all non-planar labelled rooted trees, and $\mathcal{T}_N \subset \mathcal{T}$ to denote the trees τ with $|\tau| \leq N$. The free commutative $\mathbb{R}$-algebra generated by $\mathcal{T}$ will be denoted $\mathcal{H}$. The Connes-Kreimer [20] Hopf algebra on $\mathcal{H}$ is defined as follows. Multiplication $\mu : \mathcal{H} \otimes \mathcal{H} \to \mathcal{H}$ is defined as the commutative juxtaposition of two forests, extended linearly to $\mathcal{H}$. The multiplicative unit is defined to be the empty forest $\emptyset$. The counit map $\varepsilon : \mathcal{H} \to \mathbb{R}$ is defined by $\varepsilon(\emptyset) = 1$ and $\varepsilon(\tau) = 0$ for all non-empty trees $\tau \in \mathcal{H}$. The coproduct map is defined recursively by

$$\Delta(\emptyset) = \emptyset \otimes \emptyset, \quad \Delta[\tau_1, \dots, \tau_m]_a = [\tau_1, \dots, \tau_m]_a \otimes \emptyset + (\mathrm{id} \otimes B_+^a)(\Delta\tau_1 \cdots \Delta\tau_m),$$

where $B_+^a(\tau_1 \cdots \tau_m) := [\tau_1 \cdots \tau_m]_a$ for a forest $\tau_1 \cdots \tau_m$. The definition extends to a linear multiplicative map on $\mathcal{H}$. We will occasionally use Sweedler's coproduct notation $\Delta\tau = \sum_{(\tau)} \tau^{(1)} \otimes \tau^{(2)}$ and omit the definition of the antipode S here, instead referring the reader to [59, 39]. We denote the dual of the Connes-Kreimer Hopf algebra by $\mathcal{H}^*$. For $\varphi_1, \varphi_2 \in \mathcal{H}^*$, the convolution product is defined by $\varphi_1 * \varphi_2 = \mu_{\mathbb{R}} \circ (\varphi_1 \otimes \varphi_2) \circ \Delta$, with $\mu_{\mathbb{R}} \colon \mathbb{R} \otimes \mathbb{R} \to \mathbb{R}$ denoting multiplication in $\mathbb{R}$.

A.2 B-Series Expansions of ODEs

For any tree $\tau \in \mathcal{T}$, the so-called elementary differential $F(\tau)(y)$ [18] is defined recursively by

$$F(\emptyset)(y) = y, \quad F(\bullet_i)(y) = f_i(y),$$
$$F([\tau_1, \tau_2, \dots, \tau_m]_i)(y) = f_i^{(m)}(y)(F(\tau_1)(y), F(\tau_2)(y), \dots, F(\tau_m)(y)).$$

Given a map $\varphi : \mathcal{T} \to \mathbb{R}$, the associated B-series is defined

$$B_h(\varphi, y_0) := \sum_{\tau \in \mathcal{T}} \frac{h^{|\tau|}}{\sigma(\tau)} \varphi(\tau) F(\tau)(y_0).$$

A key property of B-series is closure under composition [32, 17]: for two B-series, $B_h(\varphi_2, B_h(\varphi_1, y_0)) = B_h(\varphi_1 * \varphi_2, y_0)$, where $\varphi_1 * \varphi_2$ denotes the convolution product defined above. The exact solution of (1) has the B-series representation $y(h) = B_h(e, y_0)$, with $e(\tau) = 1/\tau!$. Likewise, a Runge–Kutta method with coefficients $\{a_{ij}\}_{1 \leq i,j \leq s}$ and $\{b_i\}_{1 \leq i \leq s}$ has the B-series representation $B_h(\varphi, y_0)$, where [18, Lemma 312B]

$$\varphi(\tau) := \sum_{i_1, \dots, i_n} b_{i_1} \prod_{(k,\ell) \in E} a_{i_k, i_\ell}$$

for a tree $\tau = (V, E, r)$ with $|\tau| = n$. We refer the reader to [33, 61, 16] for a detailed account of B-series and the Butcher group.

A.3 Branched Rough Paths

Let $\mathcal{H}$ be the Connes-Kreimer Hopf algebra of non-planar labelled rooted trees defined above.

Definition A.1. Let $\alpha \in (0, 1]$. An α-Hölder branched rough path is a map $\mathbf{X} : [0, T]^2 \to \mathcal{H}^*$ such that

1. for all $s, t \in [0, T]$ and $\tau_1, \tau_2 \in \mathcal{H}$,
$$\langle \mathbf{X}_{s,t}, \tau_1 \rangle \langle \mathbf{X}_{s,t}, \tau_2 \rangle = \langle \mathbf{X}_{s,t}, \tau_1 \tau_2 \rangle,$$
2. for all $\tau \in \mathcal{H}$,
$$\langle \mathbf{X}_{s,t}, \tau \rangle = \sum_{(\tau)} \langle \mathbf{X}_{s,u}, \tau^{(1)} \rangle \langle \mathbf{X}_{u,t}, \tau^{(2)} \rangle,$$
where $\Delta\tau = \sum_{(\tau)} \tau^{(1)} \otimes \tau^{(2)}$.
3. for all $\tau \in \mathcal{H}$,
$$\sup_{s \neq t} \frac{|\langle \mathbf{X}_{s,t}, \tau \rangle|}{|t - s|^{\alpha|\tau|}} < \infty.$$

Remark. As remarked in [34, 31], the components $\langle \mathbf{X}_{s,t}, \tau \rangle$ with $|\tau| > N$ are determined by those with $|\tau| \leq N$, where N is the largest integer such that $N\alpha \leq 1$.

The space of α-Hölder branched rough paths is a complete metric space under the metric

$$\varrho_\alpha(\mathbf{X}, \mathbf{Y}) := \sum_{\tau \in \mathcal{T}_N} \sup_{s \neq t} \frac{|\langle \mathbf{X}_{s,t} - \mathbf{Y}_{s,t}, \tau \rangle|}{|t - s|^{\alpha|\tau|}},$$

where $N = \lfloor 1/\alpha \rfloor$.

A.4 The Munthe-Kaas–Wright Hopf algebra

Given a finite alphabet A, let $\mathcal{F}_A^{\mathrm{pl}}$ denote the set of A-decorated ordered planar rooted forests, and write $\mathcal{H}_{\mathrm{MKW}}$ for the free $\mathbb{R}$-vector space on $\mathcal{F}_A^{\mathrm{pl}}$. As before, for ordered trees $\tau_1, \ldots, \tau_m$, $[\tau_1, \ldots, \tau_m]_a$ denotes the planar tree formed by grafting $\tau_1, \ldots, \tau_m$ in the given order onto a new root labelled a. The MKW Hopf algebra structure on $\mathcal{H}_{\mathrm{MKW}}$ [66] is given by:

- **Product:** the shuffle product $\sqcup\!\sqcup : \mathcal{H}_{\mathrm{MKW}} \otimes \mathcal{H}_{\mathrm{MKW}} \to \mathcal{H}_{\mathrm{MKW}}$, summing all interleavings of two ordered forests preserving the relative order within each.
- **Coproduct on trees:** the left-admissible-cuts coproduct
$$\Delta_{\mathrm{MKW}}(\tau) = \tau \otimes \emptyset + \emptyset \otimes \tau + \sum_{c \in \mathcal{C}^{\ll}(\tau)} P^c(\tau) \otimes T^c(\tau),$$
where $\mathcal{C}^{\ll}(\tau)$ is the set of non-trivial cuts whose root-level edges form a *left prefix* of the children at each vertex; $P^c(\tau)$ is the ordered forest of pruned subtrees, and $T^c(\tau)$ is the trunk. The coproduct is extended to forests via $\Delta_{\mathrm{MKW}}(\omega) = (\mathrm{id} \otimes B_-)\big(\Delta_{\mathrm{MKW}}(B_+(\omega)) - B_+(\omega) \otimes \mathbf{1}\big)$, where B_- extracts the children of a tree as an ordered forest.
- **Counit:** $\varepsilon(\emptyset) = 1$ and $\varepsilon(\tau) = 0$ for any non-empty τ.

B RDE Framework and Convergence

We transform EES ODE schemes into RDE schemes using the framework of Redmann and Riedel [74]. Similarly to classical RK schemes for ODEs, the study of these methods is conducted through the formalism of B-series (see Appendix A). A natural consequence of this analysis is that a general

Runge–Kutta method for RDEs is given in terms of tree-iterated integrals of the underlying driving process. In practice, these tree-iterated integrals cannot be simulated directly as their distributions are often intractable. Following [23], [74] replaced these tree-iterated integrals with products of increments of the driving path. This substitution simplifies the derivation of Runge–Kutta coefficients and makes it feasible to establish order conditions up to any desired order.

B.1 Simplified Runge–Kutta Methods for RDEs

We consider rough differential equations (RDEs) of the form

$$dy_t = f(y_t)\, d\mathbf{X}_t, \tag{6}$$

where $\mathbf{X}$ is an α-Hölder branched rough path, $\alpha \in (0, 1]$, and f is smooth and bounded with bounded derivatives; see Appendix A.3. Following Redmann and Riedel [74], we specialise to the geometric setting by assuming that there exist smooth approximations $\{X^h\}_{h>0}$ whose natural lifts $\{\mathbf{X}^h\}_{h>0}$ satisfy $\varrho_\alpha^g(\mathbf{X}^h, \mathbf{X}) = \mathcal{O}(h^{r_0})$ for some $r_0 > 0$, where ϱ_α^g is the inhomogeneous geometric rough path metric. Such rates are known for Gaussian processes; see Friz and Riedel [28]. Letting y^h solve (6) driven by $\mathbf{X}^h$, a simplified Runge–Kutta scheme on an equidistant grid with stepsize h is

$$\begin{aligned} y_{n+1}^h &= y_n^h + \sum_{m=1}^{d} \sum_{i=1}^{s} b_i f_m(k_i) X_{t_n,t_{n+1}}^{(m)}, \\ k_i &= y_n^h + \sum_{m=1}^{d} \sum_{j=1}^{s} a_{ij} f_m(k_j) X_{t_n,t_{n+1}}^{(m)}, \end{aligned} \tag{7}$$

where $X_{t_n,t_{n+1}}^{(m)}$ denotes the m-th component increment of X^h over $[t_n, t_{n+1}]$. Thus an ODE Runge–Kutta tableau induces an RDE scheme by weighting each tableau coefficient by the corresponding driver increment. Convergence rates for schemes of the form (7) from Redmann and Riedel [74] are recalled in Appendix B.3.

B.2 Order Conditions for General RDE Runge–Kutta Methods

The simplified scheme (7) is a special case of the general Runge–Kutta class considered by Burrage and Burrage [13, 14, 12], Redmann and Riedel [74]. Namely, consider methods of the form

$$\begin{aligned} y_{n+1} &= y_n + \sum_{m=1}^{d} \sum_{i=1}^{s} z_i^{(m)} f_m(k_i), \\ k_i &= y_n + \sum_{m=1}^{d} \sum_{j=1}^{s} Z_{ij}^{(m)} f_m(k_j), \end{aligned} \tag{8}$$

where $Z^{(1)}, \dots, Z^{(d)} \in \mathbb{R}^{s\times s}$ and $z^{(1)}, \dots, z^{(d)} \in \mathbb{R}^s$. In particular, (7) is recovered by taking $z_i^{(m)} = b_i X_{t_n,t_{n+1}}^{(m)}$ and $Z_{ij}^{(m)} = a_{ij} X_{t_n,t_{n+1}}^{(m)}$. We recall the local and global error rates for (8), as formulated in Redmann and Riedel [74], based on the adaptation of B-series to RDEs presented above.

Definition B.1. Given $h > 0$, define the maps a, φ recursively over non-planar labelled rooted trees τ by setting $\varphi(\emptyset)(h) := (1, \dots, 1)^T \in \mathbb{R}^s$, where $\emptyset$ denotes the empty tree. For a tree $\tau = [\tau_1 \cdots \tau_n]_i$ formed by joining $\tau_1, \dots, \tau_n$ by a new root labelled i, set

$$\begin{aligned} \varphi(\tau)(h) &:= \prod_{j=1}^{n} (Z^{(i)} \varphi(\tau_j)(h)), \\ a(\tau)(h) &:= \left\langle z^{(i)}, \prod_{j=1}^{n} \varphi(\tau_j)(h) \right\rangle. \end{aligned}$$

Theorem B.1 ([74]). The general Runge–Kutta method given by (8) has a local error of order $(p+1)\alpha$ if and only if

$$\langle \mathbf{X}_{t_0,t_0+h}, \tau \rangle = a(\tau)(h)$$

for all non-planar labelled rooted trees τ with p or fewer nodes, i.e. $|\tau| \le p$.

Proposition B.1 ([74, Proposition 4.1]). Let $y(t, y_0)$ denote the solution to (6) at time t starting at y_0. Suppose the Runge–Kutta method (8) has a local error of order $(p+1)\alpha$, and there exists a constant $C_1 > 0$ such that

$$|y(h, y_0) - y(h, \widetilde{y}_0)| \leq C_1 |y_0 - \widetilde{y}_0|,$$

for h sufficiently small. Then there exists $C > 0$ such that

$$\max_{n=0,\dots,N} |y(t_n) - y_n| \leq C h^{(p+1)\alpha - 1}.$$

B.3 Convergence of simplified Runge–Kutta methods

Given an ODE Runge-Kutta scheme Φ, we will write $\mathcal{R}(\Phi)$ to denote the RDE scheme of the form in (7) with the same coefficients $\{a_{ij}\}_{1\leq i,j\leq s}$ and $\{b_i\}_{1\leq i\leq s}$ as the ODE scheme.

Theorem B.2 ([74, Theorem 3.3]). Let Φ be an ODE Runge–Kutta scheme. The Runge-Kutta method $\mathcal{R}(\Phi)$ approximating y^h has a local error of order $(p+1)\alpha$, i.e.

$$y^h(t_0 + h) - y_1^h = \mathcal{O}(h^{(p+1)\alpha}),$$

if and only if the ODE Runge–Kutta method Φ is of order p.

Theorem B.3 ([74, Theorem 4.2]). Let Φ be an ODE Runge–Kutta method of order p. Suppose that f is Lip_b^γ for some $\gamma > 1/\alpha$. Then $\mathcal{R}(\Phi)$ has a global error rate of $\eta = \min\{r_0, (p+1)\alpha - 1\}$, where r_0 is the convergence rate of the Wong-Zakai approximation. That is,

$$\max_{n=0,\dots N} |y(t_n) - y_n^h| = \mathcal{O}(h^\eta).$$

B.4 Backpropagation through explicit Runge–Kutta methods

The algorithm for backpropagation through an explicit Runge–Kutta scheme Φ of the form in (7) is given in Algorithm 1. We assume the solver is applied to a (neural) RDE of the form

$$dy_t^h = f(y_t^h; \theta) d\mathbf{X}_t^h, \tag{9}$$

where θ are learnable parameters requiring backpropagation, trained with respect to a loss $L(\{y_n^h\}_{n=0}^N)$. As with all reversible schemes, a reverse step Φ^{rev} is used to recover y_n from y_{n+1}, followed by a backpropagation through the internal operations of the solver Φ. The latter step is achieved by defining $z_i = f(k_i; \theta)$ and computing the derivatives $\partial L/\partial z_i$ and $\partial L/\partial k_i$ in reverse through the stages $i = s, s-1, \dots, 1$. At each stage, a backpropagation algorithm is called to backpropagate the derivative $\partial L/\partial z_i$ through f, resulting in the derivative $\partial L/\partial k_i$ and a local derivative with respect to θ, d_θ.

Algorithm 1 Backpropagation through Explicit Runge–Kutta Schemes

Input: y_{n+1}, $\partial_{y_{n+1}} L$
Input: Running derivative with respect to θ, $\partial_\theta L$
Input: Explicit RK method Φ of the form in (7) with coefficients $\{a_{ij}\}_{1\leq i,j\leq s}$ and $\{b_i\}_{1\leq i\leq s}$.

$y_n = \Phi^{\mathrm{rev}}(y_{n+1}, dX)$
for $i = s, \dots 1$ **do**
  $\partial_{z_i} L = b_i dX \cdot \partial_{y_{n+1}} L + \sum_{j=i+1}^s a_{ji} dX \cdot \partial_{k_j} L$
  $d_\theta, \partial_{k_i} L = \mathrm{backprop}_f(\partial_{z_i} L)$
  $\partial_\theta L \mathrel{+}= d_\theta$
end for
$\partial_{y_n} L = \partial_{y_{n+1}} L + \sum_{i=1}^s \partial_{k_i} L$
return $y_n, \partial_{y_n} L, \partial_\theta L$

C Integrators on Homogeneous Spaces

This section introduces geometric numerical integration over homogeneous spaces as an extension of the well-known Lie group case and describes an example based on the manifold of symmetric positive definite (SPD) matrices. We then provide some background on commutator-free methods, of which our CF-EES$(2, 5; x)$ belongs, and show its memory-compute optimality.

C.1 Symmetric Integration over Homogeneous Spaces

For chart-based homogeneous space integrators, self-adjointness depends not only on the Butcher tableau of the underlying method, but also requires that the chosen local coordinates be compatible with time reversal. There are two natural ways to enforce this:

(i) use symmetric coordinates, centered at a geodesic or flow midpoint, in the style of self-adjoint Lie-group methods [93], or

(ii) use an embedded frozen-flow model whose frozen dynamics already integrate to an exactly reversible map on M.

The first route is intrinsic, but generally leads to midpoint-centered constructions and hence away from the present explicit frozen-flow framework. The second route is the one used here, which works for arbitrary homogeneous spaces.

Consider a homogeneous space M endowed with a transitive action $\Lambda\colon G \times M \to M$ of a Lie group G. Rather than working with vector fields on M directly, the key idea is to represent them through elements of the Lie algebra $\mathfrak{g} = T_eG$: each $v \in \mathfrak{g}$ acts on M via the **fundamental vector field**

$$v_M(y) \coloneqq \frac{\mathrm{d}}{\mathrm{d}t}\bigg|_{t=0} \Lambda(\exp(tv),\, y), \tag{10}$$

the infinitesimal version of the group action. A vector field F on M is then represented through a state-dependent generator $\xi(y) \in \mathfrak{g}$ via $F(y) = \xi(y)_M(y)$, replacing the role that Lie-algebra multiplication vy plays on the group itself:

$$gh \rightsquigarrow \Lambda(g, y), \qquad vy \rightsquigarrow v_M(y), \tag{11}$$

for $g \in G$, $v \in \mathfrak{h}_y/\mathfrak{g}$, $y \in M$. This substitution lifts standard Lie-group integration schemes from G to M. In general, this construction is only unique up to the isotropy algebra $\mathfrak{h}_y = \{A \in \mathfrak{g}\colon A_m(y) = 0\}$, the set of generators whose induced vector field at y is zero and thus produce no tangent motion. An example is given in Example C.1. To avoid this one fixes a representative of each tangent direction, typically by choosing a complement to the isotropy algebra and lifting only through that subspace.

Example C.1 (Example of degenerate choices due to the isotropy algebra)**.** Consider the sphere $S^2 \simeq \mathrm{SO}(3)/\mathrm{SO}(2)$ with the standard transitive action of $\mathrm{SO}(3)$ on $S^2 \subset \mathbb{R}^3$. At the north pole $p = e_3$, the isotropy subgroup is

$$H_p = \{R \in \mathrm{SO}(3) : Rp = p\} \simeq \mathrm{SO}(2),$$

namely the rotations about the vertical axis. Its Lie algebra consists of the infinitesimal rotations that leave p fixed to first order.
Equivalently, if $K \in \mathfrak{so}(3)$ generates a rotation about the e_3-axis, then its fundamental vector field vanishes at p:

$$K_{S^2}(p) = \frac{\mathrm{d}}{\mathrm{d}t}\bigg|_{t=0} \exp(tK)p = 0.$$

Hence, if $A \in \mathfrak{so}(3)$ represents some tangent vector at p, then so does $A + K$ for any such K, since

$$(A + K)_{S^2}(p) = A_{S^2}(p) + K_{S^2}(p) = A_{S^2}(p).$$

Thus the lift of a tangent vector from T_pS^2 to $\mathfrak{so}(3)$ is not unique: one may add any element of the isotropy algebra without changing the induced tangent motion at p.

The reverse of a frozen homogeneous-space flow is obtained by changing the sign of t. Indeed, for a fixed generator $v \in \mathfrak{g}$, by the group action identity Λ gives

$$\Lambda(\exp(-tv), \Lambda(\exp(tv), y_0)) = \Lambda(\exp(-tv)\exp(tv), y_0) = \Lambda(e, y_0) = y_0. \tag{12}$$

Hence the frozen flow is exactly reversible: its adjoint is obtained by time reversal, that is, by replacing t with $-t$. This is a property of the frozen flow itself, before any recomputation of the generator along the trajectory.

C.2 Runge–Kutta–Munthe-Kaas (RKMK) methods

Runge–Kutta–Munthe-Kaas (RKMK) methods [67] extend classical explicit Runge–Kutta schemes to manifolds by transforming the original equation $\dot{y} = F(y)$ on M into an equivalent equation on the linear space $\mathfrak{g}$. Writing the local solution as $y(t) = \Lambda(\exp(\sigma(t)), y_n)$ with $\sigma(0) = 0$, the curve $\sigma : I \to \mathfrak{g}$ satisfies

$$\dot{\sigma} = \operatorname{dexp}_{\sigma}^{-1}\big(f(\Lambda(\exp(\sigma), y_n))\big), \qquad \text{(pulled-back equation)}$$

where $\operatorname{dexp}_{\sigma} : \mathfrak{g} \to \mathfrak{g}$ is the differential of the exponential map and admits the Bernoulli expansion

$$\operatorname{dexp}_{\sigma}^{-1}(v) = \sum_{k \geq 0} \frac{B_k}{k!} \operatorname{ad}_{\sigma}^{k} v, \qquad B_{2k+1} = 0 \text{ for } k \geq 1. \qquad \text{(Bernoulli expansion)}$$

A classical Runge–Kutta tableau (a_{ij}, b_i) is then applied directly to the pulled-back equation. The stage values $u_i = h \sum_j a_{ij} k_j \in \mathfrak{g}$ and slopes

$$k_i = \operatorname{dexp}_{u_i}^{-1}\big(f(\Lambda(\exp(u_i), y_n))\big) \in \mathfrak{g}, \qquad i = 1, \ldots, s,$$

combine to give the manifold update $y_{n+1} = \Lambda\big(\exp(h \sum_i b_i k_i),\, y_n\big)$.

Achieving order p requires truncating the Bernoulli expansion through $\operatorname{ad}_{\sigma}^{k}$ for $k \leq p - 2$ [67], since each $\operatorname{ad}_{\sigma}^{k} v$ is a k-fold nested Lie bracket of stage generators. As a result, an order-p RKMK method incurs a per-stage cost that grows with p through nested commutator evaluations. Eliminating these nested commutators is the central motivation for the commutator-free constructions that follow.

C.3 Crouch–Grossman (CG) methods

Crouch–Grossman (CG) methods [21] were the first explicit Runge–Kutta-style integrators on Lie groups to dispense with commutators entirely. Each stage and the final update are realized as compositions of exponentials, but each exponential's argument involves only a single stage slope. With left-trivialized stage slopes

$$K_i = f(Y_i) \in \mathfrak{g}, \qquad i = 1, \ldots, s,$$

the stage values and update take the form

$$Y_i = \Lambda\left(\mathcal{T}\left\{\prod_{j=1}^{i-1} \exp\big(h\, \alpha_{ij}\, K_j\big)\right\},\, y_n\right), \qquad i = 1, \ldots, s,$$

$$y_{n+1} = \Lambda\left(\mathcal{T}\left\{\prod_{i=1}^{s} \exp\big(h\, \beta_i\, K_i\big)\right\},\, y_n\right),$$

with $\mathcal{T}$ denoting ordered multiplication in G. Because every exponential argument is a scalar multiple of a single stage slope, no Lie brackets appear, but the cost saving over RKMK comes at the price of additional exponential evaluations per step. CG methods are precisely the special case of the commutator-free framework of Section C.4 in which each linear combination $\sum_j a_{l;ij} K_j$ collapses to a single term.

C.4 Commutator-Free Integrators

Commutator free (CF) [19] generalize Crouch-Grossman (CG) [21] methods by allowing multiple exponentials per stage, while still avoiding the explicit evaluation of commutators as in Runge–Kutta Munthe–Kaas methods [67]. For simplicity, we present the autonomous case; the non-autonomous case is handled by the standard augmentation in time. Given internal stage values $Y_i \in M$, we evaluate the left-trivialized stage slopes

$$K_i = f(Y_i) \in \mathfrak{g}, \qquad i = 1, \ldots, s. \qquad \text{(left-trivialized slope)}$$

Each stage is then formed by acting on the current state y_n with an ordered product of exponentials of linear combinations of previously computed slopes

$$Y_i = \Lambda\left(\mathcal{T}\left\{\prod_{l=1}^{L_i} \exp\left(h \sum_{j=1}^{i-1} a_{l;ij}\, K_j\right)\right\}, y_n\right), \qquad i = 1, \ldots, s.$$

where Y_i is the stage value and y_{n+1} is the next solution point. The step update then takes the form,

$$y_{n+1} = \Lambda\left(\mathcal{T}\left\{\prod_{l=1}^{L}\exp\left(h\sum_{i=1}^{s}\beta_{l;i}\,K_i\right)\right\}, y_n\right),$$

where L_i and L denote the numbers of exponentials used in stage i and in the final update, respectively, and $\mathcal{T}$ denotes ordered multiplication in G, with smaller indices appearing to the right.

Order conditions for the coefficients $\alpha_{l;ij}$ and $\beta_{l;i}$ are derived in [71] via a Lie–Butcher / planar tree calculus, the non-commutativity of group composition forcing ordered rather than unordered trees. The canonical order-4 CF construction of Celledoni et al. [19] uses 5 exponentials per step, with one stage value reused across two stage exponentials.

C.5 Adjoint Sensitivities for Commutator-Free Neural ODEs

The continuous adjoint system underlying CF-EES is obtained by applying Wotte et al. [92, Theorem 4.5] through the homogeneous-space lift introduced in Section C.1.

C.6 Theoretical Compute and Memory Requirements of Lie Group Integrators

The per-step cost of an s-stage Lie group integrator can be decomposed as

$$C_{\text{step}} = s\,C_{\text{eval}} + N_{\exp}\,C_{\exp},$$

where C_{eval} is the cost of one vector-field evaluation and $C_{\exp}$ is the cost of one group exponential. Since all methods considered below use s stage evaluations, the main distinction in compute is the number of exponentials per step. On the memory side, generic Runge–Kutta-style Lie group methods typically retain several intermediate stage quantities, whereas low-storage constructions aim to keep this requirement fixed. The key comparison is therefore not only the exponential count, but whether this count can be achieved without increasing the number of stage registers with s.

For an s-stage Crouch–Grossman (CG) method, stage i uses $i-1$ exponentials and the final update uses s more, giving

$$N_{\exp}^{\text{CG}}(s) = \sum_{i=1}^{s}(i-1) + s = \frac{s(s+1)}{2}.$$

Thus, the exponential count of CG grows quadratically with the number of stages.

A general commutator-free (CF) method instead exponentiates linear combinations of previously computed stage slopes [19]. Writing L_i for the number of exponentials used in stage i and L for the number used in the final update, its total exponential count is

$$N_{\exp}^{\text{CF}}(s) = \sum_{i=1}^{s} L_i + L.$$

CG is recovered as the special case $L_i = i-1$ and $L = s$. In practice, CF methods are designed so that L_i and L remain small, reducing the exponential count from quadratic to linear in s. For example, the 3-stage and 4-stage commutator-free schemes of Celledoni et al. [19] require 3 and 5 exponentials per step, respectively.

A particularly efficient instance arises for 2N-storage methods. In the commutator-free formulation of Bazavov [3], exponentials are reused across stages and only two quantities are stored: the current updated state and the current stage increment. In this representation, each stage uses exactly one exponential applied to an accumulated linear combination of previously computed slopes. Consequently, whenever a classical Runge–Kutta scheme admits 2N form, the induced commutator-free implementation uses

$$N_{\exp}^{\text{2N-CF}}(s) = s$$

exponentials per step while retaining the two-register storage pattern. In this sense, 2N-CF simultaneously achieves linear exponential complexity and minimal storage within this Runge–Kutta-style commutator-free design space.

Table 5: Per-step group-exponential count and forward stage-storage for s-stage Lie group integrators. All methods use s vector-field evaluations so the comparison isolates exponential costs $N_{\exp}C_{\exp}$. We denote the methods of Celledoni et al. [19] by CMO CF and show their best-case scaling.

Method	$N_{\exp}$ per step	Scaling of $N_{\exp}$	Forward stage storage
CG	$\frac{s(s+1)}{2}$	$\mathcal{O}(s^2)$	$\mathcal{O}(s)$
CMO CF	$\sum_{i=1}^{s} L_i + L$	$\mathcal{O}(s)$	$\mathcal{O}(s)$
2N-CF	s	$\mathcal{O}(s)$	2 registers

D New EES Solvers

We now record the general $2N$ representations of the schemes $\mathrm{EES}(2,5;x)$ and $\mathrm{EES}(2,7;x)$ identified by Shmelev et al. [87], valid for every admissible parameter x.

For $x \in \mathbb{R} \setminus \{1, \pm\frac{1}{2}\}$, the Williamson $2N$ coefficients of 2N-$\mathrm{EES}(2,5;x)$ are

$$B_1 = \frac{2x+1}{4(1-x)}, \qquad B_2 = \frac{1-x}{1-4x^2}, \qquad B_3 = \frac{1-2x}{2},$$
$$A_1 = 0, \qquad A_2 = \frac{4x^2-2x+1}{2(x-1)}, \qquad A_3 = -\frac{4x^2-2x+1}{(2x-1)^2(2x+1)}.$$

At $x = \frac{1}{10}$ these evaluate to $(B_1, B_2, B_3) = (\frac{1}{3}, \frac{15}{16}, \frac{2}{5})$ and $(A_2, A_3) = (-\frac{7}{15}, -\frac{35}{32})$.

For $\mathrm{EES}(2,7;x)$ the explicit closed form of the $2N$ coefficients involves $\sqrt{2}$ and quartic polynomials in x, and is most compactly expressed in terms of the Butcher entries of the 4-stage tableau itself [87]: for either sign choice $\pm\sqrt{2}$,

$$B_1 = a_{21}, \qquad B_2 = a_{32}, \qquad B_3 = a_{43}, \qquad B_4 = b_4,$$
$$A_2 = \frac{a_{31}-a_{21}}{a_{32}}, \qquad A_3 = \frac{a_{42}-a_{32}}{a_{43}}, \qquad A_4 = \frac{b_3-a_{43}}{b_4},$$

where (a_{ij}, b_i) are the entries of the $\mathrm{EES}(2,7;x)$ Butcher tableau. At $x = \frac{1}{14}(5-3\sqrt{2})$ with the $+\sqrt{2}$ branch these evaluate to $(B_1, B_2, B_3, B_4) = \left(\frac{2-\sqrt{2}}{3}, \frac{4+\sqrt{2}}{8}, \frac{3(3-\sqrt{2})}{7}, \frac{9-4\sqrt{2}}{14}\right)$ and $(A_2, A_3, A_4) = \left(\frac{-7+4\sqrt{2}}{3}, -\frac{4+5\sqrt{2}}{12}, \frac{3(-31+8\sqrt{2})}{49}\right)$.

Unrolling the Williamson recurrence as in Bazavov [3] turns the $2N$ coefficients (A_i, B_i) above into explicit weight vectors $\beta_l \in \mathbb{R}^s$ that specify the arguments of each exponential in the commutator-free lift. This makes the induced homogeneous-space integrator CF-$\mathrm{EES}(2,5;\frac{1}{10})$ fully explicit.

Proposition D.1 (CF-$\mathrm{EES}(2,5;\frac{1}{10})$ tableau). Let $\mathcal{M} = G/H$ be a homogeneous space with transitive action $\Lambda : G \times \mathcal{M} \to \mathcal{M}$ and generator map $\xi : \mathcal{M} \to \mathfrak{g}$ (as in Section C.1). One step of CF-$\mathrm{EES}(2,5;\frac{1}{10})$ from $y_n \in \mathcal{M}$ to y_{n+1}, with $Y_0 := y_n$, is

$$K_l = \xi(Y_{l-1}), \qquad V_l = h\sum_{i=1}^{l} \beta_{l,i} K_i, \qquad Y_l = \Lambda\big(\exp(V_l), Y_{l-1}\big), \qquad l = 1,2,3, \tag{13}$$

with $y_{n+1} := Y_3$ and weight vectors given by

	$i=1$	$i=2$	$i=3$
$\beta_{1,i}$	$\frac{1}{3}$	0	0
$\beta_{2,i}$	$-\frac{7}{16}$	$\frac{15}{16}$	0
$\beta_{3,i}$	$\frac{49}{240}$	$-\frac{7}{16}$	$\frac{2}{5}$
$\sum_l \beta_{l,i} = b_i$	$\frac{1}{10}$	$\frac{1}{2}$	$\frac{2}{5}$

Row l gives the coefficients of $K_1, \dots, K_s$ inside the lth exponential $\exp(V_l)$; the final row is the Euclidean consistency check $\sum_l \beta_{l,i} = b_i$, so that on a linear manifold (§C.1) the three exponentials collapse to the classical update $y_{n+1} = y_n + h\sum_i b_i K_i$ of $\mathrm{EES}(2,5;\frac{1}{10})$.

Proof. From the $2N$ recurrence $\delta_l = A_l\delta_{l-1} + hK_l$ with $\delta_0 = 0$ and $A_1 = 0$, iteration gives $\delta_l = h\sum_{i\le l} A_lA_{l-1}\cdots A_{i+1}K_i$ (empty product $= 1$). The frozen step $Y_l = \Lambda(\exp(B_l\delta_l), Y_{l-1})$ is therefore the exponential of $V_l = h\sum_{i\le l}\beta_{l,i}K_i$ with $\beta_{l,i} = B_lA_lA_{l-1}\cdots A_{i+1}$ for $i < l$, $\beta_{l,l} = B_l$, and $\beta_{l,i} = 0$ for $i > l$. Substituting $(A_i) = (0, -\frac{7}{15}, -\frac{35}{32})$ and $(B_i) = (\frac{1}{3}, \frac{15}{16}, \frac{2}{5})$ from the preceding proposition gives the tabulated values. The identity $\sum_l \beta_{l,i} = b_i$ follows by telescoping the unrolling and matches the classical weights $b = (\frac{1}{10}, \frac{1}{2}, \frac{2}{5})$ of $\mathrm{EES}(2,5;\frac{1}{10})$. □

Both recurrences additionally admit a three-register low-storage implementation with a first-order embedded estimator, obtained by storing the final internal stage and advancing it over the remaining fraction of the step by a single Euler update; specifically, one stores the stage at $c_3 = \frac{5}{6}$ for $\mathrm{EES}(2,5;1/10)$ and the stage at $c_4 = \frac{4+\sqrt{2}}{6}$ for $\mathrm{EES}(2,7;\frac{1}{14}(5-3\sqrt{2}))$ [91, 44]. In common with other Diffrax solvers, adaptive stepping requires a fourth auxiliary register holding y_n to permit restart on step rejection.

E Order conditions for CF-EES

This section gives a brief account of how the Lie–Butcher (LB) series character of $\mathrm{CF\text{-}EES}(2,5;x)$ is computed on the Munthe-Kaas–Wright (MKW) Hopf algebra of planar rooted forests, lists the closed-form Williamson $2N$ coefficients of the family, and tabulates the symbolic character $\phi(\tau)$ for every planar tree of order at most 5. From this table the planar order conditions and the antisymmetric-order conditions of $\mathrm{CF\text{-}EES}(2,5;x)$ can be read off as identities in x. All expressions in this section are produced symbolically by the `kauri` package [85].

E.1 General form of $\mathrm{CF\text{-}EES}(2,5;x)$

For completeness, we record the explicit reused-stage form of $\mathrm{CF\text{-}EES}(2,5;x)$ for arbitrary admissible x. The underlying $\mathrm{EES}(2,5;x)$ Butcher tableau (Proposition 2.1) admits the Williamson $2N$ reduction

$$B_1 = \frac{2x+1}{4(1-x)}, \qquad B_2 = \frac{1-x}{1-4x^2}, \qquad B_3 = \frac{1-2x}{2}, \tag{14}$$

$$A_2 = \frac{4x^2-2x+1}{2(x-1)}, \qquad A_3 = -\frac{4x^2-2x+1}{(2x-1)^2(2x+1)}, \tag{15}$$

valid for $x \in \mathbb{R}\setminus\{1, \pm\frac{1}{2}\}$. Following the Bazavov $2N$ commutator-free lift [3], one step of $\mathrm{CF\text{-}EES}(2,5;x)$ from $y_n \in \mathcal{M}$ to y_{n+1} on a homogeneous space $\mathcal{M} = G/H$ with action Λ and generator $\xi : \mathcal{M} \to \mathfrak{g}$ is the two-register recurrence

$$\begin{aligned} Y_0 &:= y_n, \qquad \delta_0 := 0, \\ K_l &= \xi(Y_{l-1}), \\ \delta_l &= A_l\,\delta_{l-1} + h\,K_l, \\ Y_l &= \Lambda\big(\exp(B_l\,\delta_l),\, Y_{l-1}\big), \qquad l = 1,2,3, \end{aligned} \tag{16}$$

with $A_1 := 0$, and $y_{n+1} := Y_3$. Only the current state $Y_l \in \mathcal{M}$ and the current stage increment $\delta_l \in \mathfrak{g}$ are stored at any time, hence the "$2N$" descriptor.

Unrolling (16) expresses the exponential argument $B_l\delta_l$ as a linear combination of the slopes $K_1, \dots, K_l$,

$$B_l\delta_l = h\sum_{i=1}^{l}\beta_{l,i}K_i, \qquad \beta_{l,i} = B_lA_lA_{l-1}\cdots A_{i+1} \quad (i<l), \qquad \beta_{l,l} = B_l,$$

yielding the generic weight matrix

	$i=1$	$i=2$	$i=3$
$\beta_{1,i}$	B_1	0	0
$\beta_{2,i}$	B_2A_2	B_2	0
$\beta_{3,i}$	$B_3A_3A_2$	B_3A_3	B_3
$b_i = \sum_l \beta_{l,i}$	x	$\frac{1}{2}$	$\frac{1}{2} - x$

(17)

The bottom row recovers the Butcher weights of $\mathrm{EES}(2,5;x)$, so that on a flat manifold ($\Lambda(\exp(v),y) = y+v$, $\xi(y) = f(y)$) the three exponentials collapse to the classical update $y_{n+1} = y_n + h\sum_i b_i K_i$. At $x = \frac{1}{10}$, (14)–(17) reduce to the numerical coefficients of Proposition D.1.

It is the reused-stage feature of (16), in which the same accumulated increment δ_l supplies the argument of every exponential and the slope $K_l = \xi(Y_{l-1})$ is evaluated at the previously *computed* stage value Y_{l-1} rather than at y_n, that introduces the substitution-and-pseudo-stage subtlety in the LB-series analysis below.

E.2 LB character of CF-EES via Owren's pseudo-stage construction

Each of the three exponentials $\exp(V_l)$ ($l = 1,2,3$) in one step of $\text{CF-EES}(2,5;x)$ has an associated single-exponential character $\phi_{l,\exp} : \mathcal{H}_{\mathrm{MKW}} \to \mathbb{R}$, defined on a planar tree by the shuffle-symmetric formula $\phi_{l,\exp}([\tau_1,\dots,\tau_k]) = \frac{1}{k!}\prod_i \mathcal{V}_l(\tau_i)$, with $\mathcal{V}_l(\tau) = \sum_{i=1}^{s} \beta_{l,i}\, g_i(\tau)$ encoding the fact that the argument of the l-th exponential is a linear combination of the slopes K_i, each of which has its own row character g_i described below.

Because $\text{CF-EES}(2,5;x)$ is a reused-stage method, the slope $K_l = \xi(Y_{l-1})$ is evaluated at the previously computed stage value Y_{l-1}, so the LB character of K_l is itself a Lie–Butcher series that must be propagated through the construction. Following Owren [71, Theorem 2.5], the row character g_l at stage l satisfies the substitution recursion

$$g_l\big([\tau_1,\dots,\tau_k]\big) = \sum_{j=0}^{k} g_l\big(B_+(\tau_1,\dots,\tau_j)\big)\cdot \phi_{l,\exp}\big(B_+(\tau_{j+1},\dots,\tau_k)\big), \tag{18}$$

with base case $g_l(\bullet) = 0$ for $l = 1$ and $g_l(\bullet) = 1$ for $l \geq 2$ (so that the first stage is the identity and each later stage contributes one additional exponential). The output of the method is the result of applying the final exponential to the previous stage value, and its LB character is obtained by treating this final update as a *pseudo-stage*: on a planar tree τ, one grafts τ to a new root via B_+ and evaluates the final row character g_3 at the result,

$$\phi(\tau) := g_3\big(B_+(\tau)\big). \tag{19}$$

E.3 The LB character of $\text{CF-EES}(2,5;x)$ on planar trees of order ≤ 5

Substituting (14)–(15) into the pseudo-stage construction (19) gives, for every planar rooted tree τ of order at most 5, an explicit rational function $\phi(\tau) \in \mathbb{Q}[x]$. Table 6 lists every value, organised by tree order. The first three rows ($|\tau| \leq 2$) match $1/\tau!$ identically in x, confirming the classical order condition; deeper rows give the x-dependent values that encode the antisymmetric-order cancellations proven below.

E.4 Order theorem

Theorem E.1. For every $x \in \mathbb{R} \setminus \{1, \pm\frac{1}{2}\}$, the LB character ϕ of $\text{CF-EES}(2,5;x)$ defined by (19) satisfies, on $\mathcal{H}_{\mathrm{MKW}}$:

1. **Planar order 2:** $\phi(\tau) = 1/\tau!$ for every planar τ with $|\tau| \leq 2$.
2. **Antisymmetric order 5:** the symmetric defect $D := (\mathrm{sign}\cdot\phi) \star_{\mathrm{MKW}} \phi$, where $\mathrm{sign}(\tau) = (-1)^{|\tau|}$, satisfies $D(\tau) = \varepsilon(\tau)$ for every planar τ with $|\tau| \leq 5$.

The same construction applies to $\text{CF-EES}(2,7;x)$. The explicit symbolic table for that family contains $1+1+2+5+14+42+132 = 197$ entries with $\sqrt{2}$ throughout the rational expressions and is too large to reproduce here, but the analogous statement holds:

Theorem E.2. For every x in the admissible domain of $\mathrm{EES}(2,7;x)$ and either sign choice $\pm\sqrt{2}$ in its tableau, $\text{CF-EES}(2,7;x)$ has planar order 2 and antisymmetric order 7 on $\mathcal{H}_{\mathrm{MKW}}$.

Table 6: LB character $\phi(\tau)$ of CF-EES$(2,5;x)$ on every planar rooted tree τ with $|\tau| \le 5$, computed from the pseudo-stage construction (19) with the symbolic Williamson coefficients (14)–(15).

τ	$\phi(\tau)$	τ	$\phi(\tau)$	τ	$\phi(\tau)$
Orders 0–2 (matching $1/\tau!$ identically in x):					
$\emptyset$	1	$\bullet$	1	$\bullet^2$	$\frac{1}{2}$
Order 3:					
$\vee$	$\frac{2x-5}{32(x-1)}$	$\bullet^3$	$\frac{1}{8}$		
Order 4:					
$\vee\bullet$	$-\frac{2x+7}{192(x-1)}$	$\bullet\vee$	$-\frac{x+2}{32(x-1)}$	$\bullet^2\vee$	$\frac{1}{32}$
$\vee\vee$	$-\frac{2x+1}{64(x-1)}$	$\bullet^4$	0		
Order 5:					
$\vee\bullet\bullet$	$\frac{8x^3+24x^2+36x-41}{6144(x-1)^3}$	$\bullet\vee\bullet$	$\frac{4x^2+10x+13}{768(x-1)^2}$	$\bullet^2\vee\bullet$	$-\frac{4x+5}{384(x-1)}$
$\vee\vee\bullet$	$\frac{(2x+1)(x+2)}{256(x-1)^2}$	$\bullet\vee\vee$	0	$\bullet^2\vee\vee$	$\frac{1}{192}$
$\vee\bullet\vee$	$-\frac{1}{64(2x-1)}$	$\bullet\vee\bullet\vee$	$-\frac{2x+1}{256(x-1)}$	$\bullet^2\vee\bullet\vee$	$\frac{(2x+1)^2}{768(x-1)^2}$
$\vee\bullet\bullet\bullet$	0	$\bullet\vee\bullet\bullet$	0	$\bullet^2\vee\bullet\bullet$	0
$\vee\vee\vee$	0	$\bullet^5$	0		

F RDEs on Homogeneous spaces

The order conditions verified in Section E were stated as algebraic identities on the LB character of CF-EES$(2,5;x)$. We sketch here why these identities yield the corresponding local error rates when CF-EES$(2,5;x)$ is applied to a rough differential equation

$$dy_t = \sum_{i=1}^{d} \xi_i(y_t)_{\mathcal{M}}\, dX_t^i, \qquad y_s = y \in \mathcal{M} = G/H, \tag{20}$$

driven by an α-Hölder planarly branched rough path $\mathbb{X}$ above a control $X \in C^\alpha([0,T];\mathbb{R}^d)$ in the sense of Curry et al. [22]. The argument splits cleanly into two parts: the forward (classical) order condition requires a comparison with the exact rough-path expansion of Curry et al. [22], while the antisymmetric-order condition is purely algebraic and inherits unchanged from Section E.

F.1 Forward order

By Curry et al. [22, Theorem 8.1], the exact solution of (20) admits the LB-series representation

$$\mathbb{Y}_{st} = \sum_{\tau \in \mathcal{F}_A^{\mathrm{pl}}} \langle\, \mathbb{X}_{st} \circ \mathfrak{a}^{\ll},\, \tau \,\rangle\, \tau, \tag{21}$$

where the sum is over A-decorated planar rooted forests and $\mathfrak{a}^{\ll} : \mathcal{H}_{\mathrm{MKW}} \to \mathcal{H}_{\text{ш}}$ is the planar arborification, the Hopf-algebra morphism that maps each planar tree to the formal sum of its linear extensions. The exact LB character of the RDE on planar trees is therefore

$$e_{\mathbb{X}}(\tau) := \langle\, \mathbb{X}_{st} \circ \mathfrak{a}^{\ll},\, \tau \,\rangle,$$

which is multiplicative in the shuffle sense and depends on the driving rough path $\mathbb{X}$.

Applying CF-EES$(2,5;x)$ to (20) via the simplified Redmann–Riedel-style embedding (each hK_i replaced by $\sum_l X^l_{0,h}\, K^l_i$) gives a one-step LB character of the form

$$\phi_{\mathbb{X}}(\tau) = \phi(\tau) \cdot \langle X_{0,h},\, \tau \,\rangle, \qquad \langle X_{0,h},\, \tau \,\rangle := \prod_{v \in V(\tau)} X^{\ell(v)}_{0,h},$$

where $\phi(\tau)$ is the path-independent character of Section E.3 and $\ell(v)$ is the label of vertex v. For a *geometric* planarly branched rough path, the planar arborification factorises as $\langle\, \mathbb{X}_{st} \circ \mathfrak{a}^{\ll},\, \tau \,\rangle = \langle X_{0,h}, \tau\rangle/\tau!$, so the local order condition $\phi_{\mathbb{X}}(\tau) = e_{\mathbb{X}}(\tau)$ on every planar τ with $|\tau| \le p$ is equivalent to the algebraic identity

$$\phi(\tau) = 1/\tau!, \qquad |\tau| \le p.$$

By Theorem E.1 (i) this holds for $|\tau| \le 2$, identically in x, so CF-EES$(2,5;x)$ has local error of order $(2+1)\alpha = 3\alpha$ when applied to (20); the same argument with Theorem E.2 gives local order 3α for CF-EES$(2,7;x)$.

F.2 Backward order

The symmetric defect $D := (\mathrm{sign} \cdot \phi) \star_{\mathrm{MKW}} \phi$ is the LB character of the composition $\Phi^{\mathbb{X}^{\mathrm{rev}}} \circ \Phi^{\mathbb{X}}$, that is, one step of CF-EES$(2,5;x)$ under $\mathbb{X}$ followed by one step under the time-reversed driver $\mathbb{X}^{\mathrm{rev}}$. $D(\tau)$ can be computed entirely from the Butcher tableau of EES$(2,5;x)$ and the MKW coproduct, without depending on the driving rough path. The path-applied character of the composition factorises in the same way as the forward character,

$$D_{\mathbb{X}}(\tau) = D(\tau) \cdot \langle X_{0,h},\, \tau \,\rangle,$$

so that vanishing of $D(\tau)$ at the algebra level implies vanishing of $D_{\mathbb{X}}(\tau)$ for any rough path $\mathbb{X}$ and any tree of the same order. Theorem E.1 (ii) gives $D(\tau) = \varepsilon(\tau)$ on every planar τ with $|\tau| \le 5$, identically in x. Hence $\Phi^{\mathbb{X}^{\mathrm{rev}}} \circ \Phi^{\mathbb{X}}$ recovers the identity to local order $(5+1)\alpha = 6\alpha$, for any geometric planarly branched rough path. The same argument with Theorem E.2 gives local order 8α for CF-EES$(2,7;x)$.

F.3 Backpropagation through homogeneous-space 2N commutator-free methods

The algorithm for backpropagation through a homogeneous-space commutator-free scheme is given in Algorithm 2. We assume that the solver is at least approximately reversible, so the backward pass recovers the preceding stage states and the low-storage Lie-algebra register by applying the reverse recurrence. Unlike the Euclidean case, the adjoints with respect to stage states are covectors $\lambda_{Y_l} \in T^*_{Y_l} M$, and each reverse stage applies the pullback of the homogeneous-space action. Thus the backward sweep may be viewed as a discrete evolution on the cotangent bundle T^*M.

For

$$\Psi_l(Y, \delta) = \Lambda(\exp(B_l \delta), Y)\,,$$

we write $D_Y \Psi_l^*$ and $D_\delta \Psi_l^*$ for the adjoints of the differentials with respect to the state and Lie-algebra arguments.

Algorithm 2 Backpropagation through Homogeneous-Space $2N$ Commutator-Free Schemes

Input: y_{n+1}, $\lambda_{Y_s} = \partial_{y_{n+1}} L$
Input: Running derivative with respect to θ, $\partial_\theta L$
Input: Homogeneous-space commutator-free method Φ with $\{A_l\}_{l=1}^{s-1}$ and $\{B_l\}_{l=1}^{s}$.
  $Y_s = y_{n+1}$, $\lambda_\delta = 0$
  for $l = s, \dots, 1$ **do**
    Recover Y_{l-1}, δ_l, and K_l by the algebraic reverse recurrence Φ^{rev}
    $\lambda_{Y_{l-1}} = D_Y \Psi_l(Y_{l-1}, \delta_l)^* \lambda_{Y_l}$
    $\lambda_{\delta_l} \mathrel{+}= D_\delta \Psi_l(Y_{l-1}, \delta_l)^* \lambda_{Y_l}$
    $\lambda_{K_l} = \lambda_{\delta_l}$
    $d_\theta, \eta_l = \mathrm{backprop}_\xi(dX, \lambda_{K_l})$
    $\lambda_{Y_{l-1}} \mathrel{+}= \eta_l$
    $\partial_\theta L \mathrel{+}= d_\theta$
    $\lambda_{\delta_{l-1}} = A_l \lambda_{\delta_l}$
  end for
  $\partial_{y_n} L = \lambda_{Y_0}$
  return $y_n, \partial_{y_n} L, \partial_\theta L$

G Convergence experiments

We verify the global error rates given in Theorem B.3 experimentally by reproducing the example given in [74, 23] for $\mathrm{EES}(2,5;1/10)$ and $\mathrm{EES}(2,7;(5-3\sqrt{2})/14)$. We take the RDE

$$dy_t = \cos(y_t) d\mathbf{X}_t^{(1)} + \sin(y_t) d\mathbf{X}_t^{(2)}, \quad y_0 = 1$$

for $t \in [0,1]$, where $\mathbf{X}$ is the geometric lift of a 2-dimensional fractional Brownian motion (fBm) with Hurst index H. We compute the average of the maximal discretisation error over $M = 10$ realisations of the RDE,

$$\mathcal{E}(h) := \frac{1}{M} \sum_{i=1}^{M} \max_{n=0,\dots,N} |y_i(t_n) - y_{i,n}|,$$

where $y_i(t)$ denotes the solution to the i^{th} realisation of the RDE and $y_{i,n}$ denotes the discretisation of the i^{th} solution using an EES scheme. Additionally, we evaluate the average error when recovering the initial condition,

$$\tilde{\mathcal{E}}(h) := \frac{1}{M} \sum_{i=1}^{M} |y_0 - \tilde{y}_{i,n}|.$$

From [28], the rate r_0 can be chosen arbitrarily close to $2H - 1/2$ for a fractional Brownian motion with Hurst parameter H. It follows that we expect $\eta_1 = 2H - 1/2$ in Theorem B.3 for both $\mathrm{EES}(2,5)$ and $\mathrm{EES}(2,7)$, and $\eta_2 = 6H - 1$ for $\mathrm{EES}(2,5)$ and $\eta_2 = 8H - 1$ for $\mathrm{EES}(2,7)$. We show the rates for $H \in (0.4, 0.5, 0.6)$ in Figure 7.

The same convergence orders are observed for CF-$\mathrm{EES}(2,5)$ and CF-$\mathrm{EES}(2,7)$; see Figure 8. We consider the $\mathrm{SO}(3)$ rough differential equation

$$dX_t = \sum_{a=1}^{2} X_t \, \xi_a(X_t) \, d\mathbf{X}_t^a, \qquad X_0 = I,$$

driven by a two-dimensional fractional Brownian motion $\mathbf{X}$ with Hurst parameter $H \in \{0.4, 0.5, 0.6\}$. Writing $X = (X_{ij})_{i,j=1}^3$, the coefficient maps $\xi_a : \mathrm{SO}(3) \to \mathfrak{so}(3)$ are affine in the entries of X:

$$\xi_1(X) = \begin{pmatrix} 0 & -0.1 - 0.3X_{31} & 0.25 + 0.2X_{23} \\ 0.1 + 0.3X_{31} & 0 & -0.9 - 0.2X_{11} \\ -0.25 - 0.2X_{23} & 0.9 + 0.2X_{11} & 0 \end{pmatrix},$$

$$\xi_2(X) = \begin{pmatrix} 0 & -0.8 - 0.15X_{33} & -0.35 + 0.2X_{22} \\ 0.8 + 0.15X_{33} & 0 & -0.15 - 0.25X_{12} \\ 0.35 - 0.2X_{22} & 0.15 + 0.25X_{12} & 0 \end{pmatrix}.$$

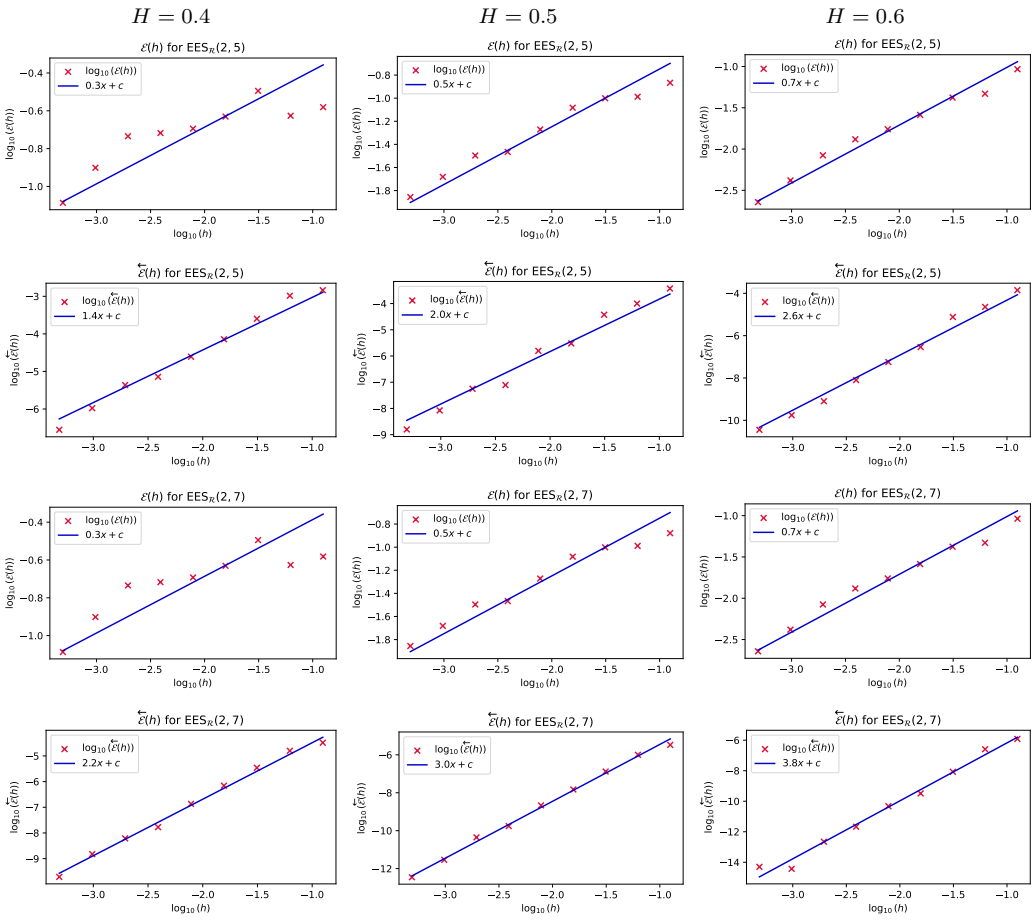


Figure 7: *Convergence rates for* EES *with* $H \in (0.4, 0.5, 0.6)$

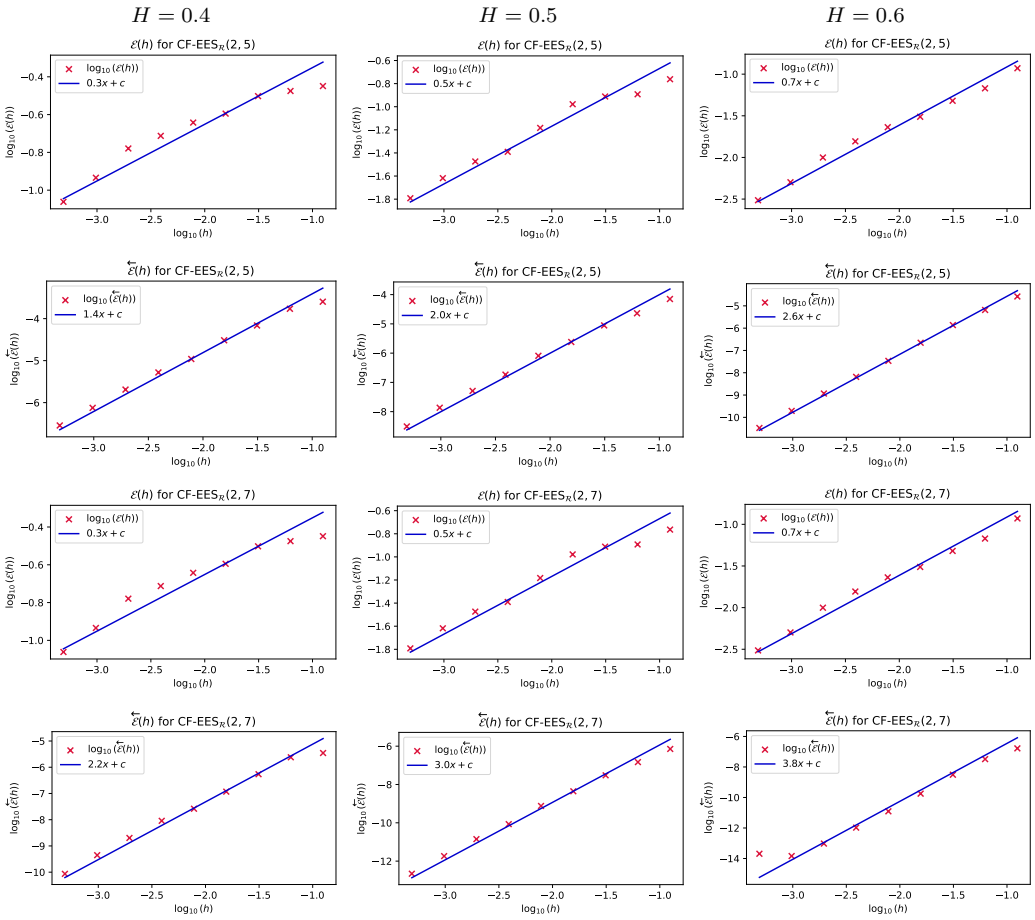


Figure 8: *Convergence rates for* CF-EES *with* $H \in (0.4, 0.5, 0.6)$

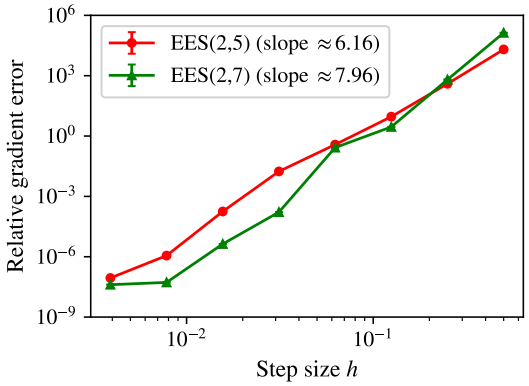


Figure 9: *The higher order of* EES$(2,7)$ *is nullified by instability due to non-smooth NSDE vector fields at practical step sizes.*

H Additional Experiments

H.1 High-dimensional GBM with stiff drift

Consider learning the dynamics of a high-dimensional geometric Brownian motion (GBM)

$$dy_t = Ay_t dt + \sigma y_t dW_t, \quad y_0 \in \mathbb{R}^d,$$

where $A \in \mathbb{R}^{d\times d}$ and $\sigma \in \mathbb{R}$. We introduce a stiff drift component by choosing $A = QDQ^T$, where $D = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{d-1})$, $\lambda_i = -20(1 + \frac{i}{d})$, and Q is a randomly generated orthogonal matrix, and take $d = 25$ and $\sigma = 0.1$. We choose to learn the dynamics using a Neural SDE of the form

$$dz_t = g(z_t; \theta_g)dt + f(z_t; \theta_f) \circ dW_t, \quad z_0 = h(\mathbf{x}, \theta_h) \in \mathbb{R}^{d_z},$$

where f, g are neural networks with the same architecture as in the OU experiment of Section 4. We integrate the SDEs over $t \in [0, 1]$ for 1,000 epochs, sampling 10,000 realisations of the dynamics at every epoch. The Adam optimiser is used with a fixed learning rate of 2×10^{-1}.

As in the previous example, Table 7 and Figure 10 show the results of training using various reversible methods, with the step size chosen such that the number of evaluations of f, g is fixed. We see that the instability caused by the stiff drift results in diverging MSE for all solvers except $\text{EES}(2, 5)$, which manages to retain moderate stability for the entire 1,000 epochs of training.

Table 7: Metrics for stiff GBM dynamics. The step size is chosen such that the total number of evaluations of f, g per integration is fixed.

Method	# Eval. / Step	Step Size	Terminal MSE	Runtime (s)
Reversible Heun	1	$1/60$	–	1283.6
MCF Euler	2	$1/30$	–	1119.9
MCF Midpoint	4	$1/15$	–	1270.1
$\text{EES}(2, 5)$	3	$1/20$	1.1803×10^{-4}	1050.0

Figure 11 shows the MSE of the gradient of the loss during training, where the true gradient is computed by autodifferentiation through a discretise-then-optimise solution, using the same solver and step size. Despite its near-reversibility, $\text{EES}(2, 5)$ achieves a lower gradient MSE compared to other solvers. This effect is likely the result of the superior stability of $\text{EES}(2, 5)$, in combination with the linear nature of the target SDE.

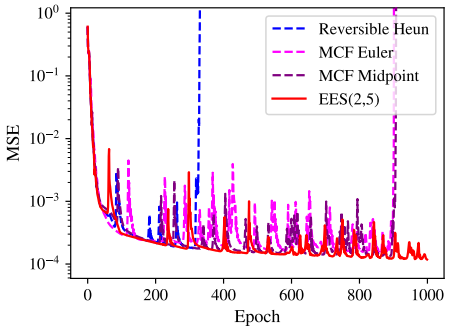


Figure 10: *Training MSE for GBM dynamics with a fixed number of evaluations of f, g.*

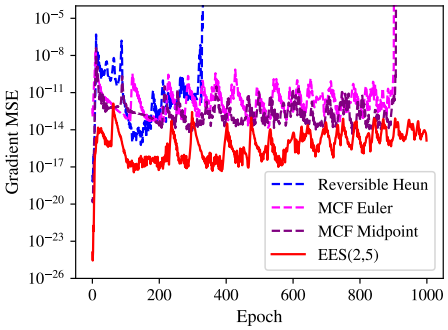


Figure 11: *Gradient MSE for GBM dynamics with a fixed number of evaluations of f, g.*

H.2 Stochastic Volatility

We present results for the remaining stochastic volatility models in Table 8. $\mathrm{EES}(2,5)$ remains the lowest runtime integrator, while maintaining identical terminal MSE performance.

Table 8: Metrics for benchmark stochastic volatility models.

Model	Method	#Eval. / Step	Step Size	Terminal MSE ($\pm 2\sigma$)	Runtime (s)
Black–Scholes	Reversible Heun	1	$1/504$	$6.46_{\pm 1.01}$	999.8
	MCF Euler	2	$1/252$	$6.46_{\pm 1.01}$	937.1
	MCF Midpoint	4	$1/126$	$6.44_{\pm 1.01}$	514.9
	$\mathrm{EES}(2,5)$	3	$1/168$	$6.44_{\pm 1.01}$	405.6
Classical Bergomi	Reversible Heun	1	$1/504$	$19.97_{\pm 1.39}$	1001.2
	MCF Euler	2	$1/252$	$19.97_{\pm 1.39}$	922.7
	MCF Midpoint	4	$1/126$	$19.98_{\pm 1.39}$	522.3
	$\mathrm{EES}(2,5)$	3	$1/168$	$19.97_{\pm 1.39}$	401.8
Local stoch vol	Reversible Heun	1	$1/504$	$6.66_{\pm 1.02}$	1010.5
	MCF Euler	2	$1/252$	$6.66_{\pm 1.02}$	924.7
	MCF Midpoint	4	$1/126$	$6.64_{\pm 1.02}$	521.1
	$\mathrm{EES}(2,5)$	3	$1/168$	$6.64_{\pm 1.02}$	414.0
Heston	Reversible Heun	1	$1/504$	$5.68_{\pm 0.82}$	986.1
	MCF Euler	2	$1/252$	$5.69_{\pm 0.82}$	928.0
	MCF Midpoint	4	$1/126$	$5.69_{\pm 0.82}$	521.7
	$\mathrm{EES}(2,5)$	3	$1/168$	$5.69_{\pm 0.82}$	409.1
Rough Heston	Reversible Heun	1	$1/504$	$6.54_{\pm 2.16}$	983.4
	MCF Euler	2	$1/252$	$6.54_{\pm 2.16}$	914.8
	MCF Midpoint	4	$1/126$	$6.55_{\pm 2.16}$	511.8
	$\mathrm{EES}(2,5)$	3	$1/168$	$6.55_{\pm 2.16}$	402.8
Quadratic rough Heston	Reversible Heun	1	$1/504$	$5.38_{\pm 1.21}$	977.2
	MCF Euler	2	$1/252$	$5.39_{\pm 1.21}$	866.2
	MCF Midpoint	4	$1/126$	$5.39_{\pm 1.21}$	478.8
	$\mathrm{EES}(2,5)$	3	$1/168$	$5.39_{\pm 1.21}$	410.7

H.3 Molecular Dynamics

We investigate whether $\mathrm{EES}(2,5)$ can train a neural molecular dynamics force field from long Langevin rollouts under a fixed force-evaluation budget while retaining low-memory adjoint differentiation. This setting is motivated by force-field objectives defined on trajectory-level ensemble observables, such as vibrational spectra, rather than only on static energy-surface quantities. The central difficulty is therefore to obtain useful gradients through long stochastic trajectories without storing the full rollout.

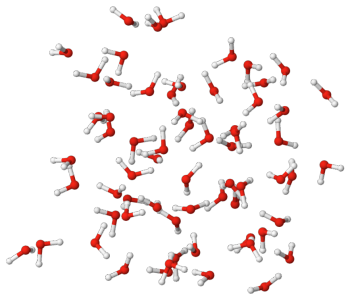


Figure 12: *Initial configuration of the 64-molecule water system*

We benchmark on a 64-molecule (192-atom) water system with a pre-trained embedded atom neural network (EANN) force field [96]. The Langevin state is $y_t = (r_t, v_t) \in \mathbb{R}^{6N_{\mathrm{atom}}} = \mathbb{R}^{1152}$, where $r_t \in \mathbb{R}^{3N_{\mathrm{atom}}}$ and $v_t \in \mathbb{R}^{3N_{\mathrm{atom}}}$ denote the atomic positions and velocities at time t. All solvers are compared under matched total evaluations of the drift and diffusion fields. In contrast to the full IR-spectrum fitting pipeline of Han and Yu [35], our benchmark optimizes a differentiable proxy for the spectral objective: the normalized squared dipole-velocity signal accumulated along finite-temperature Langevin trajectories.

For each batch b, we simulate paired trajectories from velocities $\pm v_0$ and minimize the empirical mean

$$\mathcal{L}_{\mathrm{MD}}^{(B)}(\theta) = \frac{1}{B}\sum_{b=1}^{B} \frac{1}{T\,N_{\mathrm{mol}}} \int_0^T \left\| \dot{\mu}_\theta^{(b)}(t) \right\|_2^2 \,\mathrm{d}t. \tag{22}$$

Here $\dot{\mu}_\theta$ is a charge-weighted molecular dipole-velocity proxy computed along the simulated trajectory, so the experiment retains the main computational bottleneck of spectral fitting: differentiating a neural force field through long stochastic rollouts of large state vectors.

Table 9: Metrics for molecular dynamics. The step size is chosen so that each solver uses 252 evaluations per integration over the same rollout time. MCF midpoint became unstable.

Method	# Eval. / Step	Step Size	Terminal MSE ($\times 10^{-2}$)	Runtime (s)
Reversible Heun	1	$1/2520$	$44.85_{\pm 0.90}$	1083.5
MCF Euler	2	$1/1260$	$44.66_{\pm 0.42}$	984.1
MCF Midpoint	4	$1/630$	–	757.6
EES$(2,5)$	3	$1/840$	$45.04_{\pm 1.09}$	577.7

Under this fixed-budget comparison, EES$(2,5)$ attains a terminal proxy error statistically indistinguishable from the stable baselines while reducing wall-clock time by roughly 47% relative to Reversible Heun and 41% relative to MCF Euler. MCF Midpoint diverged at the prescribed step size and is therefore omitted from the accuracy comparison. We note one limitation: the present subsection evaluates a differentiable Langevin proxy rather than the full experimental IR spectral loss, and full-spectrum fitting together with memory-scaling measurements should be reported separately before any stronger end-to-end IR-fitting claim is made.

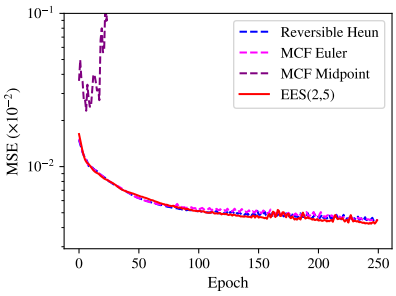


Figure 13: *Training proxy MSE for Langevin MD.*

I Experimental and Implementation Details

I.1 Hardware and Software Details

Software details. Experiments use Python 3.13 with CUDA 13.1. Models are implemented in JAX 0.10.0 [9] and Equinox 0.13.7 [46]. Differential equation solves use Diffrax 0.7.1 [45], dataloading uses Cyreal 0.2.1 [62], and path-signature losses use Stochastax 0.5.0 and Pysiglib 3.0.0 [86] in the manifold and Euclidean cases, respectively. The implementation of the Reversible adjoint used in our experiments is sourced from Sam McCallum's Diffrax fork (`ReversibleAdjoint`).

The writing of this work required the creation of three software packages to support EES, $2N$, and geometric numerical integration in Diffrax and Julia. We present these packages here and the license under which they are released.

Table 10: New packages introduced to support the present work

Package	**Ecosystem**	**Functionality**	**License**
Diffrax-lowstorage	Jax	$2N$ integrators, incl. EES	`Apache-2.0`
Georax	Jax	Geometric integrators, incl. CF-EES	`Apache-2.0`
EffectivelySymmetric.jl	SciML	Standard, $2N$, and CF-EES methods	`Apache-2.0`

Hardware details. All training and evaluation runs were conducted on a single workstation equipped with one NVIDIA RTX 5080 GPU with 16GB of GPU memory, an AMD Ryzen 9 9950X3D CPU, and 64GB of system memory, running Ubuntu 24.04 LTS.

I.2 Additional Details for OU Experiments

Dynamics. We learn the Ornstein–Uhlenbeck (OU) dynamics

$$dy_t = \nu(\mu - y_t)\,dt + \sigma\,dW_t, \quad y_0 \in \mathbb{R},$$

under the high-volatility regime $\nu = 0.2$, $\mu = 0.1$, $\sigma = 2$.

Model. We use a Neural Langevin SDE (LSDE) [69],

$$dz_t = g(z_t;\theta_g)\,dt + f(t;\theta_f) \circ dW_t, \quad z_0 = h(\mathbf{x},\theta_h) \in \mathbb{R}^{d_z},$$

where h is a learnable affine function of the input data $\mathbf{x} = \{x_n\}_{n\geq 0}$, $x_n \in \mathbb{R}^2$, sampled from the true OU dynamics, and g, f are neural networks parametrised by θ_g, θ_f respectively. We take latent dimensionality $d_z = 32$, and parametrise f, g as 2-layer neural networks of width 32 with LipSwish activations.

Training. The SDEs are integrated over $t \in [0, 10]$ and the LSDE is trained for 250 epochs using the Adam optimiser with a fixed learning rate of 10^{-3}. At each epoch, 50,000 realisations of the trained dynamics are sampled and the MSE loss is computed against the true OU dynamics.

I.3 Additional Details for GBM Experiments

Dynamics. We learn the dynamics of a high-dimensional geometric Brownian motion (GBM)

$$dy_t = Ay_t\,dt + \sigma y_t\,dW_t, \quad y_0 \in \mathbb{R}^d,$$

where $A \in \mathbb{R}^{d\times d}$ and $\sigma \in \mathbb{R}$. We introduce a stiff drift component by choosing $A = QDQ^T$, where $D = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{d-1})$ with $\lambda_i = -20\,(1 + \frac{i}{d})$ and Q a randomly generated orthogonal matrix; we take $d = 25$ and $\sigma = 0.1$.

Model. We use a Neural Langevin SDE (LSDE) [69],

$$dz_t = g(z_t;\theta_g)\,dt + f(z_t;\theta_f) \circ dW_t, \quad z_0 = h(\mathbf{x},\theta_h) \in \mathbb{R}^{d_z}, \tag{23}$$

where h is a learnable affine function of the input data $\mathbf{x} = \{x_n\}_{n\geq 0}$, $x_n \in \mathbb{R}^2$, sampled from the true dynamics, and g, f are neural networks parametrised by θ_g, θ_f respectively.

Training. We integrate the SDEs over $t \in [0, 1]$ for 1,000 epochs, sampling 10,000 realisations of the dynamics at every epoch. The Adam optimiser is used with a fixed learning rate of 2×10^{-1}.

I.4 Additional Details for Stochastic Volatility Experiments

Model. We use an unconditional Euclidean neural SDE $\mathrm{d}x_\tau = f(x_\tau,\tau)\mathrm{d}\tau + g(x_\tau,\tau)\mathrm{d}W_\tau$, $x_0 = \mathbf{1}$, for the stochastic-volatility benchmarks. The drift is a 4-layer MLP (width 16, LipSwish), while the diffusion is a 3-layer MLP (width 16, LipSwish, softplus output scaled by 0.2). Both take $(\tau, x_\tau) \in \mathbb{R}^{1+d}$ as input, and g is interpreted as the diagonal matrix $\mathrm{diag}(\sigma(x_\tau,\tau))$, giving learned state- and time-dependent coordinatewise volatility.

Dataset. We embed all stochastic volatility models in the RDE framework of Bonesini et al. [8], which spans classical Black–Scholes [6] to contemporary rough volatility models [29]. Numerically, we form the lead–lag path on a uniform noise grid of $N_{\text{noise}} = 128$ intervals and integrate the Wong–Zakai ODE on a finer uniform grid of $N_{\text{RDE}} = 768$ fixed steps with linear interpolation using Tsitouras' 5(4) Runge–Kutta method [89], recording states at the noise times. The underlying Riemann-Liouville volatility process is simulated with the hybrid method of [5]. Parameter selections are shown in Table 11. We adopt the rBergomi parameter values of Rough [77], namely $v_0 = 0.04$, $\nu = 1.991$, $H = 0.25$, and $\rho = -0.848$. For SPX options on 30/05/2022, this calibration gives a model-implied volatility surface with mean-squared error 3.73×10^{-5} relative to market implied volatilities. The remaining coefficients are fixed to standard benchmark values in plausible financial ranges. Since we use the model in its standard form, we refer to [8, Equation 1.2] for the full specification.

Table 11: Parameters for the rough volatility model configurations. Missing entries indicate parameters not used by the model.

Model	S_0	v_0	ρ	ν	H	λ	$\bar{v}$
Black–Scholes	1.0	0.04	–	–	–	–	–
Classical Bergomi	1.0	0.04	-0.7	–	–	–	–
Local stoch volatility	1.0	0.04	-0.3	–	–	1.0	0.04
Heston	1.0	0.04	-0.7	0.5	–	1.5	0.04
Rough Heston	1.0	0.04	-0.7	0.5	0.1	1.5	0.04
Quadratic rough Heston	1.0	0.04	–	–	0.1	1.0	–
Rough Bergomi	1.0	0.04	-0.848	1.991	0.1	–	–

Training. The model is trained with truncated (time-augmented) path-signature MMD^2 loss, vanilla SGD at learning rate 10^{-3}, batch size $4{,}096$, 100 epochs, no schedule or gradient clipping at a fixed NFE budget of 504. Test-time paths are single-rollout trajectories from $x_0 = \mathbf{1}$, evaluated via two-sample KS statistic at step $t = 55$.

I.5 Additional Details for the Stochastic Kuramoto Experiments

Data-generating dynamics. Trajectories are simulated from equation (5) with bimodal natural frequencies $\Omega_i \in \{+P, -P\}$ [26], default parameters $m = 1$, $K = 2.0$, $P = 0.5$, $D = 0.05$. The deterministic $N = 2$ subsystem admits a unique stable phase-locked equilibrium at $\Delta\theta_\infty = \arcsin(2P/K)$ for $K > 2P$ [1], used as a verification anchor for the simulator (Appendix below). The integration horizon is $T = 5\,\text{s}$, sub-sampled to $n_{\text{obs}} = 200$ uniform observation timepoints from a fine integration grid of $n_{\text{fine}} = 16{,}384$ steps using diffrax's `Heun` solver. Splits are 5,000 / 1,000 / 1,000 trajectories (train / val / test) per network size $N \in \{2, 4, 8\}$, each with an independent initial condition and Brownian path.

Simulator verification. At $\sigma = 0$, $N = 2$, the deterministic 2-oscillator phase difference at $T = 5\,\text{s}$ is independent of n_{fine} for $n_{\text{fine}} \geq 1024$ (relative variation $< 10^{-4}$ across $n_{\text{fine}} \in \{1024, 2048, 4096, 8192, 16384\}$), confirming Heun convergence at the chosen production grid. Residual $\sim 7.7\%$ deviation from $\arcsin(2P/K) = \pi/6$ is consistent with the analytical decay constant of the linearised dynamics around the phase-locked equilibrium ($\sim 0.5\,\text{s}^{-1}$) at finite T rather than integrator error. At $\sigma > 0$, the per-trajectory order parameter $r(t) = \left|N^{-1}\sum_j e^{i\theta_j}\right|$ saturates to a stationary mean $r_\infty \approx 0.70 \pm 0.24$ over an ensemble of 128 trajectories, placing the operating point in the partial-synchronisation regime as designed.

Model. The neural SDE on $T\mathbb{T}^N = \mathbb{T}^N \times \mathbb{R}^N$ uses MLP drift and diffusion fields of width 128 over the feature embedding $(\sin\theta, \cos\theta, \omega) \in \mathbb{R}^{3N}$, with depths 3 (drift) and 2 (diffusion), SiLU activation, and a softplus-output diffusion scaled by 0.1. The diffusion is decoupled (additive noise on ω only). Integration uses CFEES(2,5) on the product Lie group $T\mathbb{T}^N$, lifted via Bazavov's commutator-free construction [3].

Loss. Multi-horizon energy score evaluated at horizons $h \in \{T/8, T/4, T/2, T\}$ with $m = 4$ Monte Carlo rollouts per horizon, using the wrapped-on-θ, plain-on-ω distance $d((\theta_a, \omega_a), (\theta_b, \omega_b)) = \sum_i |\text{wrap}(\theta_a^i - \theta_b^i)| + \sum_i |\omega_a^i - \omega_b^i|$. Optimisation uses AdamW at peak learning rate 10^{-3}, gradient clipping at global norm 1, 30 epochs, batch size 64.

Adjoint pilot (Table 12). Before any training sweep, we verify that the three adjoints under test compute mathematically equivalent gradients. Using a fixed (model, data, Brownian-path) cell at $N = 2$, $n_{\text{steps}} \in \{200, 1000, 5000\}$, batch 32, we compute the relative ℓ_2 distance between each adjoint's gradient and a fine-dt reference at $n_{\text{steps,ref}} = 10{,}000$ under CFEES(2,5) with the Reversible adjoint. Differences across adjoints at the same n_{steps} are at the float32 round-off floor; the residual gap to the fine-dt reference reflects the discretisation difference between the test cell and reference grid (and uses an independent Brownian path), not adjoint error.

Table 12: M3.1 + M3.3 gradient fidelity. Values are relative ℓ_2 distance to a fine-dt CFEES(2,5) with the Reversible adjoint reference at $n_{\text{steps,ref}} = 10{,}000$. The three adjoints agree to 4 significant figures at every n_{steps} (the reported gap is shared discretisation error vs the fine grid).

n_{steps}	Reversible	Full	Recursive
200	9.251×10^{-2}	9.251×10^{-2}	9.251×10^{-2}
1000	9.290×10^{-2}	9.290×10^{-2}	9.290×10^{-2}
5000	9.378×10^{-2}	9.378×10^{-2}	9.378×10^{-2}

Memory scaling (Figure 5b). Table 13 gives the absolute peak GPU memory measurements that underlie Figure 5b: one forward+backward solve through the Kuramoto neural SDE at $N = 1000$, batch 64, hidden width 128, on a single GPU, with each cell run in an isolated subprocess.

Table 13: Peak GPU memory (MiB) for one forward+backward solve of the Kuramoto NSDE, $N = 1000$, batch 64.

n_{steps}	CF-EES(2,5) (Reversible)	CG2 (Full)	CG2 (Recursive)
50	356	670	648
100	356	696	650
200	352	743	653
500	355	891	658
1,000	355	1,134	663
2,000	355	2,040	674
5,000	355	4,970	690
10,000	352	OOM	712

I.6 Additional Details for Sphere Latent SDE Experiments

Setting. Zeng et al. [94] parametrise the drift of the latent SDE as a Chebyshev polynomial in t of an MLP applied to the encoder output, and integrate the resulting Stratonovich SDE with one-step *geometric Euler–Maruyama*: at every step a Lie-algebra increment $\omega = K(t)\,\mathrm{d}t + \sigma\,\mathrm{d}W \in \mathfrak{so}(n)$ is formed and lifted to $\mathrm{SO}(n)$ via $\exp(\omega)$, which then acts on the current state $z_t \in S^{n-1}$ by left multiplication. Their ELBO combines reconstruction at observed times with a closed-form Girsanov path-KL on the sphere; both terms reparameterise through every integrator step, so backpropagation must traverse the entire trajectory—the regime in which a reversible adjoint pays off.

Backbone. The standard backbone setup for the UCI Human Activity benchmark of Zeng et al. [94] uses $z \in S^{15}$, batch 64, and $N = 228$ integration steps per trajectory. We replace the geometric Euler–Maruyama step with CF-EES$(2, 5)$ and route backpropagation through the Reversible adjoint; the encoder, Chebyshev drift, decoder, and classification head are left unchanged.

Memory. Figure 6 reports peak GPU memory for one forward+backward through the integrator alone. The matrix exponential on $\mathfrak{so}(16)$ is heavier per step than the elementwise lift on $\mathbb{T}^d$, so the memory ratio at a given N is larger than for the Kuramoto benchmark (Figure 5b). Table 14 gives the underlying measurements.

Table 14: Peak GPU memory (MiB) for one forward+backward solve of the latent SDE on S^{15}, batch 64. The reversible measurement at $n_{\text{steps}} = 5{,}000$ was not collected; the figure extrapolates from the constant trend at lower step counts.

n_{steps}	CF-EES(2,5) (Reversible)	Geometric Euler (Full)
50	22	212
200	23	802
800	28	3,166
2,000	40	7,894
5,000	–	19,711

Compute parity. Table 4 reports test accuracy after 30 training epochs at a fixed total NN-evaluation budget per trajectory $B = 30$ (geometric Euler–Maruyama uses $N = B$ steps; CF-EES$(2, 5)$ uses $N = B/3$), averaged over two seeds. The published 990-epoch accuracy of Zeng et al. [94] at the full $N = 228$ grid is $90.56 \pm 0.45\%$; both solvers in our 30-epoch parity sweep land within the expected range for the truncated training and reduced step count.

I.7 Additional Details for Molecular Dynamics Experiment

Model. The EANN potential uses $n_{\mathrm{GTO}} = 12$ radial Gaussian-type orbitals per species pair with a 6 Å cutoff, followed by a per-element MLP of two hidden layers of width 64 with SiLU activations and LayerNorm [96]; pre-trained weights are loaded from `params_eann4.pickle`. Forces are obtained by automatic differentiation of the total energy. The Langevin SDE is integrated with friction $\gamma = 1.0\ \mathrm{ps}^{-1}$ and fluctuation–dissipation noise $\sigma = \sqrt{2\gamma k_{\mathrm{B}} T/\mathbf{m}}$ at $T = 298.15$ K, with all quantities expressed in GROMACS units (nm, ps, g/mol). The dipole-velocity proxy uses charge weights $w_{\mathrm{O}} = 1$, $w_{\mathrm{H}} = -1/2$ and is accumulated as an augmented coordinate of the integrator state so its gradient flows through the full rollout.

Dataset. Initial configurations are drawn from the bundled `water64.pdb` structure (cubic periodic box, edge 1.86 nm). Per training step we sample a batch of 6 initial conditions: positions are the reference geometry perturbed by $\mathcal{N}(\mathbf{0}, 10^{-3}\ \mathrm{nm})$ and velocities are drawn from the Maxwell–Boltzmann distribution at T. Each batch element is duplicated under time reversal $\mathbf{v}_0 \mapsto -\mathbf{v}_0$, giving 12 effective trajectories. Pair lists are rebuilt at every force evaluation with a 0.6 nm neighbour-list cutoff.

Training. We rollout for $t_{\mathrm{end}} = 0.1$ ps and train for 250 optimization steps with Adam at learning rate 5×10^{-4}, gradients clipped to global norm 1.0. All non-reversible base solvers (Euler, Midpoint) are lifted to the reversible setting via the algebraically reversible wrapper of McCallum and Foster [60] with coupling parameter $\lambda = 0.999$, and every solver uses the Reversible adjoint. The reported terminal proxy MSE is the loss of (22) averaged over the batch at the final training step; runtime is wall-clock time for the full 250-step run on a single GPU with fixed random seed across solvers.

I.8 Memory Benchmark for Figure 1

Figure 1 reports peak XLA scratch memory (`temp_bytes`) for one forward+backward solve of an SDE on the 7-torus $\mathbb{T}^7$, batch 1,024, hidden width 128. Each cell is run in an isolated subprocess. Table 15 gives the underlying measurements in MiB.

Table 15: Peak XLA scratch memory (MiB) for one forward+backward solve of an SDE on $\mathbb{T}^7$, batch 1,024.

n_{steps}	CF-EES (Reversible)	CG2 (Full)	CG2 (Recursive)	CG4 (Full)	CG4 (Recursive)
5	390	384	673	402	708
10	391	385	673	403	708
20	391	385	673	404	708
50	390	386	673	405	708
100	390	387	673	406	708
200	390	390	673	408	708
400	390	395	673	413	708
800	391	406	674	425	709
2,000	390	439	674	458	709
5,000	391	522	675	540	710
10,000	390	658	676	676	712

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We verify our claims with both theoretical proofs regarding solver order, integration spaces, and convergence. We then numerically validate our accuracy/runtime claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: There is a dedicated section in the discussion entitled limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We prove all theoretical claims: (1) that $\mathrm{EES}(2, 5; x)$ and $\mathrm{EES}(2, 7; x)$ are 2N schemes, (2) that $\mathrm{EES}(2, 5; x)$ retains its forward and backward orders after lifting to CF, (3) that both orders are retained in the RDE cases.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide a algorithm blocks for backpropagation, the Butcher tableaux of each method, and a complete description of the geometry for manifold experiments. Model architectures are described in the main text, and further detailed with hyperparameters and training descriptions in the appendices.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code for our reversible integrations, baseline models, and experiments is provided in the anonymous GitHub repositories.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The body sections describe the model, dataset, and training regimes employed. We then provide a complete description of the model, dataset and training scheme used in each experiment in Appendix I.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report 2 standard deviations on our accuracy metrics throughout our experiments. We were unable to add them for the molecular dynamics and Kuramoto experiments as each run was computationally expensive.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report memory requirements and wall-clock times for each experiment. Hardware used is reported in Appendix I.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We do not involve human subjects, or datasets with any identifying components or non-free licenses. We disclose experimental procedures for reproducibility, and have anonymised code submissions.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: As our work concerns fundamental methods for improving the efficiency of neural SDE training, we do not expect any adverse societal impacts or harms.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: As our contribution regards foundational methods, and we do not release any pretrained models or datasets as part of our work, we do not believe there is substantial risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We provide citations to the baseline models used and abide by the license terms when one is provided.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All new packages are presented in Table 10, which also document the licenses under which they are released. Documentation and a brief usage guide is provided in the ReadMe files available on the anonymised repositories.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our experiments do not employ crowdsourced data or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our experiments do not employ crowdsourced data or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLM usage was limited to grammar checking, stylistic advice, plotting improvements, and assistance in implementing standard methods from the literature.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.