Proceedings of Machine Learning Research 277:1–25, 2025 Machine Learning Meets Differential Equations: From Theory to Applications

# Learning non-Markovian Dynamical Systems with Signature-based Encoders

**Eliott Pradeleix** ELIOTT.PRADELEIX@POLYTECHNIQUE.EDU
*Ecole Polytechnique, 91120, Palaiseau France*
*CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), Université Paris-Saclay, 91405, Orsay*

**Rémy Hosseinkhan-Boucher** REMY.HOSSEINKHAN@UNIVERSITE-PARIS-SACLAY.FR
*CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), Université Paris-Saclay, 91405, Orsay, France*

**Alena Shilova** ALENA.SHILOVA@INRIA.FR
*Inria TAU, Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), Université Paris-Saclay, 91405, Orsay, France*

**Onofrio Semeraro** ONOFRIO.SEMERARO@CNRS.FR
**Lionel Mathelin** LIONEL.MATHELIN@CNRS.FR
*CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique (LISN), Université Paris-Saclay, 91405, Orsay*

**Editors:** Cecília Coelho, Bernd Zimmering, M. Fernanda P. Costa, Luís L. Ferrás, Oliver Niggemann

## Abstract

Neural ordinary differential equations offer an effective framework for modeling dynamical systems by learning a continuous-time vector field. However, they rely on the Markovian assumption—that future states depend only on the current state—which is often untrue in real-world scenarios where the dynamics may depend on the history of past states. This limitation becomes especially evident in settings involving the continuous control of complex systems with delays and memory effects. To capture historical dependencies, existing approaches often rely on recurrent neural network (RNN) based encoders, which are inherently discrete and struggle with continuous modeling. In addition, they may exhibit poor training behavior. In this work, we investigate the use of the signature transform as an encoder for learning non-Markovian dynamics in a continuous-time setting. The signature transform offers a continuous-time alternative with strong theoretical foundations and proven efficiency in summarizing multidimensional information in time. We integrate a signature-based encoding scheme into encoder-decoder dynamics models and demonstrate that it outperforms RNN-based alternatives in test performance on synthetic benchmarks. Code is available at : `https://github.com/eliottprdlx/Signature-Encoders-For-Dynamics-Learning.git`.

**Keywords:** Delay Differential Equations, Learning Dynamical Systems, Signature Transform, Neural Differential Equations, Continuous-time Modeling, Latent Space, Encoding

## 1. Introduction

Neural Ordinary Differential Equation (NODE), first introduced by Chen et al. (2018), allows one to model physical systems through an ODE of the form

$$\dot{\mathbf{x}}(t) = f_\theta(\mathbf{x}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0$$

with a learnable vector field $f_\theta$, initial condition $\mathbf{x}_0 \in \mathbb{R}^d$ and state $\mathbf{x}(t) = (x_1(t), \ldots, x_d(t)) \in \mathbb{R}^d$. Although effective in many situations, this framework fails when trying to model non-Markovian systems, *i.e.*, systems whose future is influenced by their history, not just the present state, for instance delayed differential equations (Kuang, 1993) of the form $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{x}(t-\tau), t)$. These non-Markovian differential equations naturally arise in biology (Gopalsamy, 1992; Erneux, 2009; Bressloff, 2013), reinforcement learning and (stochastic) control theory (Holt et al., 2023; Hoglund et al., 2023), especially under partial observability. Numerous models have been developed specifically to tackle these differential equations. (Zhu et al., 2021; Monsel et al., 2024a,b).

To allow for more expressiveness than the standard NODE, an augmented version was proposed by Dupont et al. (2019), the so-called Augmented Neural ODE (ANODE). Letting $\mathbf{a}(t) \in \mathbb{R}^p$ denote a point in an augmented space, the ODE problem is formulated as

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{a}}(t) \end{bmatrix} = f_\theta\left(\begin{bmatrix} \mathbf{x}(t) \\ \mathbf{a}(t) \end{bmatrix}, t\right), \quad \begin{bmatrix} \mathbf{x}(0) \\ \mathbf{a}(0) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{0} \end{bmatrix}.$$

While the additional dimensions introduced by ANODE may alleviate the bottleneck faced by NODE, *i*) it comes at the cost of lifting to a higher dimensional space, *ii*) the initial condition still depends solely on the current state, whereas explicit history-dependent information is required when modeling non-Markovian systems. In order to provide such information, several latent dynamics models have been proposed. These models aim to learn the vector field in a latent space, either by explicitly defining it in advance (*e.g.*, using a Laplace-domain representation (Holt et al., 2022)) or by learning it implicitly from data. To do so, an encoder compresses history dependencies into a latent initial condition, and a decoder unrolls the trajectory to both reconstruct and extrapolate in the original space. In that spirit, Rubanova et al. (2019) proposed Latent ODE to model the vector field in a latent space, and Yıldız et al. (2019) proposed to add higher-order terms. Nonetheless, these two models still rely on the numerical solver of Neural ODE, which might fail to accurately solve the initial value problem for some type of differential equations (*e.g.* stiff ones) (Holt et al., 2022).

To model a broader class of differential equations, other models have been proposed (Holt et al., 2022; Biloš et al., 2021), the main idea being to learn directly the latent vector *flow* to avoid relying on numerical solver for ODEs. These methods have proven to be very successful, although the choice of the encoder has been mostly overlooked. Indeed, the encoder is systematically chosen to be a recurrent neural network (RNN)-based model (Cho et al., 2014; Rubanova et al., 2019; De Brouwer et al., 2019; Biloš et al., 2021). However, RNN are inadequate to deal with continuous time series because of their discrete nature. Moreover, they are computationally inefficient because they operate sequentially and are prone to vanishing gradient issues (Pascanu et al., 2013). While ODE-GRU approaches

(Rubanova et al., 2019; De Brouwer et al., 2019) and GRU-Flow (Biloš et al., 2021) are better suited for continuous-time modeling, they still rely on the RNN architecture.

As an alternative, the signature transform introduced by Chen (1954, 1957, 1958) has recently been used in several realms including rough path theory (Lyons, 1998; Lyons et al., 2007; Lyons, 2014), finance, stochastic control (Morrill et al., 2021; Sabate-Vidales et al., 2020; Hoglund et al., 2023; Arribas, 2018), and machine learning (Bonnier et al., 2019; Fermanian, 2021; Liao et al., 2021; Chevyrev and Kormilitzin, 2016; Moreno-Pino et al., 2024). The signature transform can be viewed as a collection of statistics that summarize the sequential structure of a trajectory, and it has proven to be a very effective tool to summarize the information of paths and dependencies across different dimensions, with high computational efficiency (Bonnier et al., 2019; Reizenstein and Graham, 2020).

Moreover, numerous models leveraging the signature transform (Bonnier et al., 2019; Kidger et al., 2020; Morrill et al., 2021; Liao et al., 2021; Moreno-Pino et al., 2024) have shown increased performance in tasks involving continuous-time modeling in comparison to standard discrete RNNs and attention-based architectures. In addition, theoretical work (Fermanian, 2021) shows that there is a deep connection between RNN and signatures, the latter being in some sense a continuous-time counterpart of the former.

In this work, we present a unified approach that integrates various models for learning dynamical systems within a general encoder-decoder framework. Then, we propose a signature-based encoder designed to learn from non-Markovian systems. Unlike RNN-based architectures that rely on discrete-time transitions, our encoding scheme is inherently suited for continuous-time modeling and may mitigate issues such as vanishing gradients, thereby facilitating more effective training. To support these claims, we empirically show that signatured-based architectures exhibit better predicting performance and lead to faster training in comparison to their RNN-based counterparts.

## 2. Signature transform

The aim of this section is to provide the reader with a brief introduction to the signature transform and to give some intuition behind its use in the machine learning realm.

**Definition 1 ($C^1$ Path)** *Let $a < b$ be two real numbers. A $C^1$* path *is a continuously differentiable mapping $\mathbf{x} = (x_1, \dots, x_d) : t \in [a, b] \to \mathbb{R}^d$. We denote by $C^1([a, b]; \mathbb{R}^d)$ the set of such paths.*

We consider time series of the form $(\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)) \in \mathbb{R}^{d\times n}$, sampled from trajectories $(\mathbf{x}(t))_{a\leq t\leq b}$ of typical dynamical system mentioned in the Introduction. Thus, we may see the vector $(\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)) \in \mathbb{R}^{d\times n}$ as being the time discretized version of a $C^1$ path $\mathbf{x}$.

**Definition 2 (Signature transform of $C^1$ paths)** *Let $\mathbf{x} \in C^1([a, b]; \mathbb{R}^d)$. The signature of $\mathbf{x}$ is then defined as the collection of iterated integrals*

$$\mathrm{Sig}(\mathbf{x}) := \left( \left( \int \cdots \int_{a<t_1<\cdots<t_k<b} \prod_{j=1}^{k} \frac{\mathrm{d}x_{i_j}}{\mathrm{d}t}(t_j)\mathrm{d}t_1 \cdots \mathrm{d}t_k \right)_{1\leq i_1,\dots,i_k\leq d} \right)_{k\geq 0},$$

*where the $k = 0$ term is taken to be $1 \in \mathbb{R}$. The truncated signature of depth $N$ of $\mathbf{x}$ is defined as*

$$\mathrm{Sig}^N(\mathbf{x}) \coloneqq \left( \left( \int_{a<t_1<\cdots<t_k<b} \cdots \int \prod_{j=1}^{k} \frac{\mathrm{d}x_{i_j}}{\mathrm{d}t}(t_j)\mathrm{d}t_1 \cdots \mathrm{d}t_k \right)_{1\leq i_1,\ldots,i_k\leq d} \right)_{0\leq k\leq N}.$$

We introduce the signature transform only for $C^1$ paths for convenience, but it is actually defined on a broader class of functions (see Appendix A), making it applicable to paths that are not strictly $C^1$, such as those arising in stiff differential equations. The signature transform provides a structured way to summarize sequential data by hierarchically extracting a series of statistics from a path, thus offering a principled approach to deal with temporal data. Intuitively, one can think of the signature as an *infinite dictionary* of path features, where simple statistics such as increments or areas under the curve appear at the first level, and higher-order terms capture more nuanced interactions. This layered representation makes this transformation particularly well-suited for machine learning applications, as it allows models to work with data in a compressed form yet rich in terms of information. We refer the reader to Bonnier et al. (2019); Fermanian (2021); Chevyrev and Kormilitzin (2016) for further information about the use of signature methods in machine learning.

Briefly, the signature of a path captures its essential characteristics in a unique, robust to irregular-sampling, interpretable and efficient way, while being expressive enough so that any continuous function of the path can be arbitrarily approximated by a linear function of the signature, effectively acting as a universal nonlinearity. In addition, the magnitude of higher order terms behaves in a way that *reasonable* depths are enough to essentially capture the information of the path. These claims are stated in a more formal way in Appendix A.

## 3. Encoder-decoder architectures for learning dynamical systems

Let $\mathcal{D}$ be a dataset consisting of time series $(\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)) \in \mathbb{R}^{d\times n}$ sampled from trajectories $(\mathbf{x}(t))_{t\geq 0}$ of some of the aforementioned dynamical systems. Let $u_\alpha : \mathbb{R}^{d\times n} \to \mathbb{R}^l$, $l \ll dn$, an encoder and $v_\beta : \mathbb{R}^l \times [0, +\infty[ \to \mathbb{R}^d$ a decoder, respectively parameterized by $\alpha$, $\beta$. The goal of the encoder is to compress the history into a representation in the latent space, and the decoder can be interpreted as the composition of the flow in the latent space and the mapping back to the original space.

Within the context of learning dynamical systems, we define an encoder–decoder model as follows

$$\hat{\mathbf{x}}_{\alpha,\beta}(t) \coloneqq v_\beta(u_\alpha(\mathbf{x}(t_1), ..., \mathbf{x}(t_n)), t).$$

Such a model is trained on the dataset $\mathcal{D}$ using the mean squared error minimization objective

$$\min_{\alpha,\beta} \sum_{(\mathbf{x}(t_1),\ldots,\mathbf{x}(t_n))\in\mathcal{D}} \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}(t_i) - \hat{\mathbf{x}}_{\alpha,\beta}(t_i)||^2.$$

Once trained, this architecture (Figure 1) allows us to extrapolate and make continuous predictions on $\mathbf{x}(t > t_n)$. The most trivial example of encoder-decoder model is NODE,

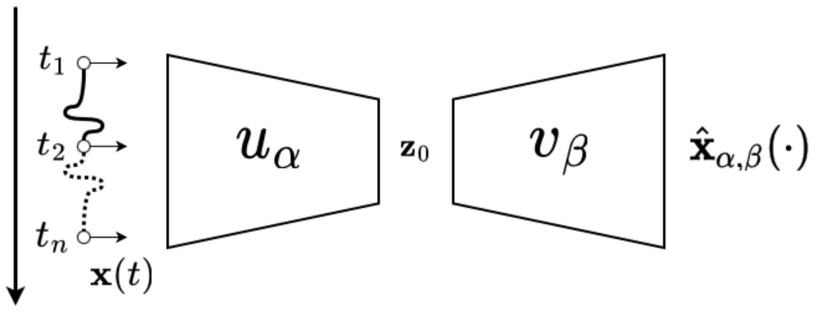

Figure 1: Encoder-decoder architecture for learning dynamical systems.

where the encoder is just a simple projection $u_\alpha : (\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)) \to \mathbf{x}(t_1)$[1], and the decoder is $v_\beta : (\mathbf{z}_0, t) \to \text{ODESolve}(f_\beta, \mathbf{z}_0, t)$ with $f_\beta$ the vector field. For ANODE, the encoder becomes $u_\alpha : (\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)) \to \begin{bmatrix}\mathbf{x}(t_1)^1 & \mathbf{0}\end{bmatrix}$, and the decoder $v_\beta : (\mathbf{z}_0, t) \to \pi(\text{ODESolve}(f_\beta, \mathbf{z}_0, t))$ with $\pi$ being the projection on the first $d$ coordinates.

## 4. Methodology

**Motivation** Architectures such as Neural ODEs (Chen et al., 2018), Augmented Neural ODEs (Dupont et al., 2019), Latent ODEs/flows (Rubanova et al., 2019; Biloš et al., 2021), ODE2VAE (Yıldız et al., 2019) and Neural Laplace (Holt et al., 2022) all fall under the general framework discussed in Section 3, with different designs for $u_\alpha$ and $v_\beta$. In all cases, the encoder is either a simple coordinate projection or a RNN-based model. In the following, we propose a signature-based model for $u_\alpha$.

**Signature-based encoder** We now present a signature-based encoder, compatible with the aforementioned dynamics learning models, and largely inspired from the literature (Bonnier et al., 2019; Liao et al., 2021; Moreno-Pino et al., 2024). The intuition behind such a choice of encoding scheme is multifaceted. Indeed, the goal of the encoder is to perform feature selection on paths. Signature-based models : *i*) have proven to be excellent path descriptors and exhibit universality (Fermanian, 2021; Bonnier et al., 2019), *ii*) are particularly suited for continuous-time modeling, especially to jointly account for multidimensional correlations (Bonnier et al., 2019; Liao et al., 2021; Moreno-Pino et al., 2024), *iii*) are more computationally efficient (Bonnier et al., 2019; Reizenstein and Graham, 2020) and interpretable than standard RNNs.

Let $(\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)) \in \mathbb{R}^{d \times n}$ be a stream of data. Let $\phi_\theta : \mathbb{R}^{d \times m} \to \mathbb{R}^e$, $m < n$, a feed-forward neural network. As proposed by Bonnier et al. (2019), we first apply the mapping

$$\Phi_\theta(\mathbf{x}(t_1), \dots, \mathbf{x}(t_n)) \coloneqq (\phi_\theta(\mathbf{x}(t_1), \dots, \mathbf{x}(t_m)), \dots, \dots, \phi_\theta(\mathbf{x}(t_{n-m+1}), \dots, \mathbf{x}(t_n))),$$

to our stream of data, akin to 1D convolution. The rationale is that, since we have to truncate the signature, and as a relevant choice of the depth is *a priori* unknown, applying

1. or $\mathbf{x}(t_n)$ for extrapolation (see Appendix C for further details).

a learnable layer before taking the signature should help the convergence (Bonnier et al., 2019). A vector $\Phi_\theta(\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)) \in \mathbb{R}^{e\times(n-m+1)}$ is obtained. Then, we apply the $N^{th}$-truncated signature to get a vector $(\mathrm{Sig}^N \circ \Phi_\theta)(\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n)) \in \mathbb{R}^q$ with $q = \frac{e^{N+1}-1}{e-1}$. Finally, we need a projection mapping $g_\xi : \mathbb{R}^q \to \mathbb{R}^l$, with $l$ the dimension of the latent space. In practice, we choose it to be a feed-forward neural network. *In fine*, the encoder is defined as

$$u_\alpha(\mathbf{x}(t_1), ..., \mathbf{x}(t_n)) \coloneqq (g_\xi \circ \mathrm{Sig}^N \circ \Phi_\theta)(\mathbf{x}(t_1), \ldots, \mathbf{x}(t_n))$$

with $\alpha = (\theta, \xi)$.

## 5. Numerical experiments

We evaluate our signature-based encoder on a diverse set of dynamical systems, encompassing a variety of dimensions, delays, stiffness, and chaotic behaviors, with applications in biology, healthcare, and engineering. For each system, our encoder is tested within two distinct model architectures-Neural Laplace (Holt et al., 2022) and Neural Flow ResNet (Biloš et al., 2021) (see Appendix B for detailed model descriptions) and compared with its RNN-based counterpart used in the original paper. In what follows, Sig Neural Laplace (resp. Sig Neural Flow ResNet) denotes the model where the original encoder from Appendix B is replaced with the signature-based encoder from Section 4.

To test the performance of the models, we use the exact same method as Holt et al. (2022). We evaluate model performance in an extrapolation setting by splitting each sampled trajectory into two parts $[0, T/2]$ and $[T/2, T]$. The first half is used for encoding, and the second half for prediction. For each dynamical system, we sample 1,000 trajectories, each initialized from a distinct initial condition and consisting of 200 time points. To ensure a fair comparison focused on the choice of encoder, we keep the decoder architecture constant in terms of parameter count across all configurations, while ensuring that the signature-based encoder consistently uses fewer parameters than its RNN-based counterpart (see Table 3). For further details, see Appendix C and D.

### 5.1. Description of test datasets

**Delayed Lotka-Volterra** We consider a delayed variant of the classical Lotka–Volterra system (Gopalsamy, 1992; Kuang, 1993) which incorporates a fixed delay capturing realistic biological lags. The model writes

$$\begin{aligned}\dot{x}_1(t) &= x_1(t)\left(1 - x_2(t-\tau)\right),\\ \dot{x}_2(t) &= \frac{1}{2}x_2(t)\left(1 - x_1(t-\tau)\right)\end{aligned}$$

with $\tau > 0$ the delay.

**Spiral DDE** Spiral delay differential equations (Zhu et al., 2021) arise in biological and healthcare systems, capturing nonlinear delayed feedback where the dynamics with saturation effect introduced by the hyperbolic tangent. The model writes

$$\dot{\mathbf{x}}(t) = A\tanh(\mathbf{x}(t) + \mathbf{x}(t-\tau))$$

with $\mathbf{x}(t) \in \mathbb{R}^2$, $\tau > 0$ the delay and $A$ is a $2 \times 2$ matrix with real coefficients.

**Delayed Fitzhugh-Nagumo** The delayed FitzHugh-Nagumo dynamical system (Erneux, 2009; Bressloff, 2013) models the interaction between a fast activator $x_1$, and a slow recovery variable $x_2$. The system writes

$$\begin{aligned}\dot{x}_1(t) &= x_1(t) - \frac{x_1(t)^3}{3} - x_2(t-\tau) + I,\\ \dot{x}_2(t) &= \varepsilon\left(x_1(t) + a - bx_2(t)\right)\end{aligned}$$

where $a, b, I > 0$ and $\epsilon > 0$ are system parameters, and $\tau > 0$ the delay. The system is known to exhibit stiffness when $\epsilon \ll 1$.

**Delayed Rössler** The delayed Rössler dynamical system extends the classical Rössler model (Rössler, 1976) of chaotic dynamics by incorporating time-delayed feedback, which enables the investigation of broader temporal behaviors. The model writes

$$\begin{aligned}\dot{x}_1(t) &= -x_2(t) - x_3(t),\\ \dot{x}_2(t) &= x_1(t) + ax_2(t),\\ \dot{x}_3(t) &= b + x_3(t)\left[x_1(t-\tau) - c\right],\end{aligned}$$

where $a, b, c > 0$ are system parameters and $\tau > 0$ the delay. The classical system is known to be chaotic for some well-chosen parameters.

## 5.2. Results

Table 1: Test RMSE (mean ± std) averaged across 5 runs (random seed initialization). Best results per group bolded. NODE and ANODE are used as baselines.

| Method | Delayed Lotka-Volterra | Spiral DDE | Delayed Fitzhugh-Nagumo | Delayed Rössler |
|---|---|---|---|---|
| ANODE | .4472 ± .0642 | .0435 ± .0070 | .1001 ± .0531 | 1.3104 ± .2578 |
| NODE | .6061 ± .2521 | .0586 ± .0260 | .0653 ± .0226 | 4.0168 ± 3.1656 |
| Neural Laplace | .1063 ± .0184 | .0426 ± .0088 | .0250 ± .0296 | .2764 ± .1182 |
| Sig Neural Laplace | **.0540 ± .0214** | **.0264 ± .0035** | **.0076 ± .0016** | **.2153 ± .0787** |
| Neural Flow ResNet | .3360 ± .1039 | .1675 ± .0508 | **.0513 ± .0185** | 1.3076 ± .2799 |
| Sig Neural Flow ResNet | **.2693 ± .0343** | **.1343 ± .0207** | **.0401 ± .0235** | **.4501 ± .1229** |

**Test set performance** Table 1 shows that our signature-based encoder consistently outperforms its RNN-based counterpart in terms of extrapolation performance, measured by the test RMSE on test sets. Moreover, we witness a global reduction of the standard deviation of the test RMSE, suggesting a more stable training w.r.t. the weight initialization.

**Sensitivity analysis and ablation study** Table 2 (and Table 4 in Appendix G) demonstrates that increasing the depth substantially improves performance, highlighting the importance of the signature transform. Moreover, the learnt mapping $\Phi_\theta$ appears to be highly effective, supporting the intuition proposed by Bonnier et al. (2019) that such a mapping can mitigate issues arising when path information depends on higher-order signature terms.

Table 2: Sensitivity analysis and ablation study for Sig Neural Laplace. Test RMSE (mean ± std) over 5 runs. Best results bolded.

| Study | Config | Delayed Lotka-Volterra | Spiral DDE | Delayed FitzHugh-Nagumo | Delayed Rössler |
|---|---|---|---|---|---|
| Depth $N$ | 1 | $.1592 \pm .0798$ | $.0316 \pm .0054$ | $.0148 \pm .0094$ | $.5130 \pm .2012$ |
| | 2 | $\mathbf{.0590 \pm .0146}$ | $.0348 \pm .0113$ | $.0178 \pm .0108$ | $.2394 \pm .1015$ |
| | 3 | $\mathbf{.0496 \pm .0243}$ | $\mathbf{.0261 \pm .0038}$ | $\mathbf{.0116 \pm .0048}$ | $\mathbf{.1682 \pm .0373}$ |
| $\Phi_\theta$ | × | $.1435 \pm .0889$ | $.0319 \pm .0081$ | $\mathbf{.0236 \pm .0115}$ | $.7351 \pm .1415$ |
| | ✓ | $\mathbf{.0605 \pm .0195}$ | $\mathbf{.0250 \pm .0015}$ | $\mathbf{.0177 \pm .0136}$ | $\mathbf{.2143 \pm .0840}$ |

**Training speed** Figure 2 (and Appendix F) shows that the signature-based models achieve faster training, suggesting a more favorable loss landscape or informative gradient updates. It supports the claims made in the introduction that using the signature as an encoder may alleviate gradient issues encountered by RNNs (Pascanu et al., 2013).

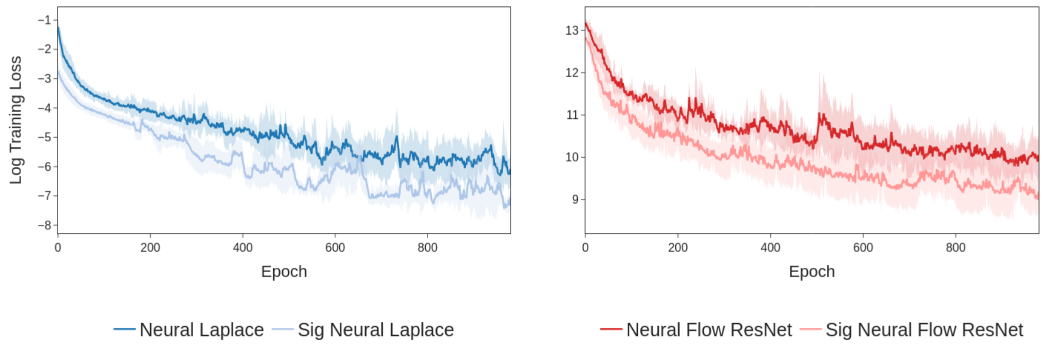

Figure 2: Training loss (log) vs. epochs on Spiral DDE averaged over 5 runs.

**Multidimensional correlations** One reason signature-based encoders may perform better is their ability to capture multidimensional correlations within trajectories. To highlight this, we introduce a coupling factor $\gamma > 0$ in the Delayed Fitzhugh-Nagumo model and evaluate performance as $\gamma$ increases (see Appendix E for further details). Figure 3 shows that signature-based model performance remains more stable than RNN-based, showcasing this advantage.

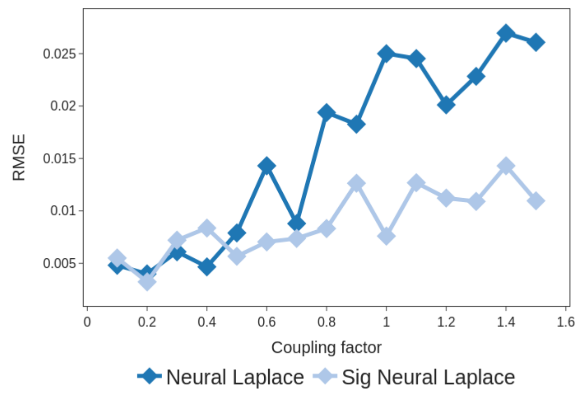

Figure 3: Test RMSE vs. coupling factor $\gamma$ averaged over 5 runs.

**Additional benchmarks** Appendix G shows that using the signature is more computationally efficient. It further examines the models' performance when corrupting input trajectory with noise or sub sampling it.

## 6. Conclusion and future work

In this paper, we proposed a unified approach that brings together diverse models for learning dynamical systems under a global encoder-decoder framework, and introduced a signature-based encoder for learning non-Markovian dynamics. We implemented this novel encoding scheme in two state-of-the-art models—Neural Laplace and Neural Flow ResNet—both capable of handling a broad class of differential equations, including delayed and stiff systems. To the best of our knowledge, previous work has predominantly relied on RNN-based encoders.

We have presented a signature-based architecture that is fully compatible with the general encoder-decoder class of models introduced in this paper. We conducted comparative numerical experiments on synthetic benchmarks. Test datasets included a range of dynamical systems, covering multiple dimensions, delays and stiffness. Our results show that signature-based models yield significant improvements in both accuracy and training stability, lead to faster training and are more computationally efficient. To assess the impact of each encoder component, we performed an ablation study and tailored experiments to explain the observed performance gains.

The signature transform is drawing attention in the machine learning community, and we believe the performance improvement demonstrated in this study provides a valuable contribution to fields such as continuous time reinforcement learning (Yildiz et al., 2021), control (Holt et al., 2023) and time series forecasting. Furthermore, by showcasing the advantages of the signature transform over RNN-based encoders, this work highlights its potential to serve as a standard encoding mechanism for developing future models for learning dynamical systems.

## References

Imanol Perez Arribas. Derivatives pricing using signature payoffs, 2018.

Marin Biloš, Johanna Sommer, Syama Sundar Rangapuram, Tim Januschowski, and Stephan Günnemann. Neural flows: Efficient alternative to neural ODEs. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, pages 21325–21337, 2021.

Patric Bonnier, Patrick Kidger, Imanol Perez Arribas, Cristopher Salvi, and Terry J. Lyons. Deep Signature Transforms. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 3099–3109, 2019.

Paul C. Bressloff. *Waves in Neural Media: From Single Neurons to Neural Fields*. Lecture Notes on Mathematical Modelling in the Life Sciences. Springer, New York, NY, 2013. ISBN 978-1-4614-8865-1. doi: 10.1007/978-1-4614-8866-8.

Kuo-Tsai Chen. Iterated integrals and exponential homomorphisms. *Proceedings of the London Mathematical Society*, s3-4(4):502–512, 1954.

Kuo-Tsai Chen. Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula. *Annals of Mathematics*, 65:163–178, 1957.

Kuo-Tsai Chen. Integration of paths - a faithful representation of paths by non-commutative formal power series. *Transactions of the American Mathematical Society*, 89:395–407, 1958.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 6571–6583, 2018.

Ilya Chevyrev and Andrey Kormilitzin. A primer on the Signature method in machine learning, 2016.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. GRU-ODE-bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 3134–3144, 2019.

Thomas Erneux. *Applied Delay Differential Equations*, volume 3 of *Surveys and Tutorials in the Applied Mathematical Sciences*. Springer, New York, NY, 2009. ISBN 978-0-387-74371-4. doi: 10.1007/978-0-387-74372-1.

Adeline Fermanian. *Learning Time-Dependent Data with the Signature Transform*. Phd thesis, Sorbonne Université, 2021. URL https://theses.hal.science/tel-03507274.

Peter K. Friz and Nicolas B. Victoir. *Multidimensional Stochastic Processes as Rough Paths: Theory and Applications*. Cambridge University Press, 2010.

K. Gopalsamy. *Stability and Oscillations in Delay Differential Equations of Population Dynamics*, volume 74 of *Mathematics and Its Applications*. Kluwer Academic Publishers, Dordrecht, 1992. ISBN 978-0-7923-1594-0. doi: 10.1007/978-94-015-7920-9.

Ben Hambly and Terry J. Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, 171(1):109–167, 2010. doi: 10.4007/annals.2010.171.109.

Melker Hoglund, Emilio Ferrucci, Camilo Hernandez, Aitor Muguruza Gonzalez, Cristopher Salvi, Leandro Sanchez-Betancourt, and Yufei Zhang. A neural RDE approach for continuous-time non-Markovian stochastic control problems. Presented at the Workshop on New Frontiers in Learning, Control, and Dynamical Systems, ICML, 2023. Peer-reviewed workshop contribution.

Samuel Holt, Alihan Hüyük, Zhaozhi Qian, Hao Sun, and Mihaela van der Schaar. Neural Laplace control for continuous-time delayed systems. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 1747–1778. PMLR, 25–27 Apr 2023.

Samuel I. Holt, Zhaozhi Qian, and Mihaela van der Schaar. Neural Laplace: Learning diverse classes of differential equations in the laplace domain. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 8811–8832. PMLR, 2022.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, 2020.

Patrick Kidger, Ricky T. Q. Chen, and Terry J. Lyons. "hey, that's not an ODE": Faster ODE Adjoints via Seminorms. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5443–5452. PMLR, July 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. San Diego, CA, USA.

Yang Kuang. *Delay Differential Equations: With Applications in Population Dynamics*, volume 191 of *Mathematics in Science and Engineering*. Academic Press, Boston, 1993. ISBN 978-0124276109.

Shujian Liao, Terry J. Lyons, Weixin Yang, Kevin Schlegel, and Hao Ni. Logsig-RNN: a novel network for robust and efficient skeleton-based action recognition. In *Proceedings of the 32nd British Machine Vision Conference (BMVC)*, page 173, 2021.

Terry J. Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2):215–310, 1998.

Terry J. Lyons. Rough paths, signatures and the modelling of functions on streams. In *Proceedings of the International Congress of Mathematicians Seoul 2014*. Kyung Moon SA, 2014. ISBN 9788961058070. © 2014 by SEOUL ICM 2014 Organizing Committee.

Terry J. Lyons, Michael Caruana, and Thierry Lévy. *Differential Equations Driven by Rough Paths*, volume 1908 of *Lecture Notes in Mathematics*. Springer, 2007.

Thibault Monsel, Emmanuel Menier, Onofrio Semeraro, Lionel Mathelin, and Guillaume Charpiat. Neural ddes with learnable delays for partially observed dynamical systems. *arXiv preprint*, arXiv:2410.02843, Oct 2024a. Preprint.

Thibault Monsel, Onofrio Semeraro, Lionel Mathelin, and Guillaume Charpiat. Time and state dependent neural delay differential equations. In *Proceedings of the 1st ECAI Workshop on "Machine Learning Meets Differential Equations: From Theory to Applications"*, volume 255 of *Proceedings of Machine Learning Research*, pages 1–20. PMLR, Oct 20 2024b.

Fernando Moreno-Pino, Álvaro Arroyo, Harrison Waldon, Xiaowen Dong, and Álvaro Cartea. Rough transformers: Lightweight and continuous time series modelling through signature patching. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

James Morrill, Cristopher Salvi, Patrick Kidger, James Foster, and Terry J. Lyons. Neural rough differential equations for long time series. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7829–7838. PMLR, 2021.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International conference on machine learning (ICML)*, pages 1310–1318, 2013.

Jeremy F. Reizenstein and Benjamin Graham. Algorithm 1004: The iisignature library: Efficient calculation of iterated-integral signatures and log signatures. *ACM Transactions on Mathematical Software (TOMS)*, 46(1):8:1–8:21, 2020. doi: 10.1145/3371237.

Otto E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ODEs for irregularly-sampled time series. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 5321–5331, 2019.

Marc Sabate-Vidales, David Šiška, and Lukasz Szpruch. Solving path dependent PDEs with LSTM networks and path signatures, 2020.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12009–12018. PMLR, 18–24 Jul 2021.

Çağatay Yıldız, Markus Heinonen, and Harri Lähdesmäki. ODE$^2$VAE: Deep generative second order odes with Bayesian neural networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 13412–13421, 2019.

Qunxi Zhu, Yao Guo, and Wei Lin. Neural Delay Differential Equations. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*, 2021.

Zulko. Delay Differential Equation Solver. https://github.com/Zulko/ddeint, 2014.

## Appendix A. Properties of the signature

In this section, we introduce the signature transform and its main properties on a broader class of paths. For convenience, we use the classical notation from the stochastic calculus literature.

**Definition 3 (Path)** *Let $a < b$ be two real numbers. A (continuous) path is a (continuous) mapping $X = (X^1, \dots, X^d) : t \in [a, b] \to \mathbb{R}^d$.*

**Definition 4 (Total variation)** *The total variation of a continuous path $X : [a, b] \to \mathbb{R}^d$ is defined by $||X||_{\mathrm{TV}} := \sup_{\mathcal{P} \subset [a,b]} \sum_{i=1}^{n-1} |X_{t_{i+1}} - X_{t_i}|$, where $\mathcal{P} := (t_1, \dots, t_n)$ denotes some partition of the interval $[a, b]$.*

We denote by $BV([a, b], \mathbb{R}^d)$ the set of continuous paths of bounded variation $X : [a, b] \to \mathbb{R}^d$ that satisfy $||X||_{\mathrm{TV}} < +\infty$.

**Definition 5 (Signature transform)** *Let $X \in BV([a, b], \mathbb{R}^d)$. The signature of $X$ is then defined as the collection of iterated integrals*

$$\begin{aligned}\mathrm{Sig}(X) &:= \left( \int_{a<t_1<\cdots<t_k<b} \cdots \int \mathrm{d}X_{t_1} \otimes \cdots \otimes \mathrm{d}X_{t_k} \right)_{k \geq 0} \\ &= \left( \left( \int_{a<t_1<\cdots<t_k<b} \cdots \int \mathrm{d}X_{t_1}^{i_1} \cdots \mathrm{d}X_{t_k}^{i_k} \right)_{1 \leq i_1, \dots, i_k \leq d} \right)_{k \geq 0},\end{aligned}$$

*where $\otimes$ denotes the tensor product, the $k = 0$ term is taken to be $1 \in \mathbb{R}$, and the integral is defined in the sense of Riemann–Stieltjes (or Young) integration (Friz and Victoir, 2010). The truncated signature of depth $N$ of $X$ is defined as*

$$\mathrm{Sig}^N(X) := \left( \int_{a<t_1<\cdots<t_k<b} \cdots \int \mathrm{d}X_{t_1} \otimes \cdots \otimes \mathrm{d}X_{t_k} \right)_{0 \leq k \leq N}.$$

**Definition 6 (Tensor algebra)** *The tensor algebra of* $\mathbb{R}^d$ *is defined as*

$$T((\mathbb{R}^d)) := \prod_{k=0}^{\infty} (\mathbb{R}^d)^{\otimes k}.$$

The signature transform of a path $X$ lives in the tensor algebra space $T((\mathbb{R}^d))$, endowed with the tensor multiplication and the componentwise addition. It exhibits several properties, making it a good candidate for machine learning applications. First, with a time augmentation, the signature transform uniquely determines the encoded path.

**Definition 7 (Time-augmented path)** *Let* $X \in BV([a,b],\mathbb{R}^d)$*. The time-augmented path* $\tilde{X}$ *is defined as* $\tilde{X}_t := (t, X_t)$ *for all* $t \in [a,b]$*.*

**Proposition 8 (Uniqueness)** *If* $\tilde{X}, \tilde{Y}$ *denote the time-augmented paths, then* $\mathrm{Sig}(\tilde{X}) = \mathrm{Sig}(\tilde{Y})$ *implies that* $\tilde{X} = \tilde{Y}$*.*

A proof of Proposition 8 can be found in Hambly and Lyons (2010). In addition, the magnitude of the signature of order $N$ decays factorially.

**Proposition 9 (Factorial decay)** *Let* $X \in BV([a,b],\mathbb{R}^d)$*.*
*Then for any* $N \geq 0$*,*

$$\left\| \int \cdots \int_{a<t_1<\cdots<t_N<b} dX_{t_1} \otimes \cdots \otimes dX_{t_N} \right\|_{(\mathbb{R}^d)^{\otimes N}} \leq \frac{1}{N!} ||X||_{\mathrm{TV}}^N$$

.

A proof of Proposition 9 can be found in Lyons et al. (2007).

**Remark 10** *Using the definition of the signature transform, it is clear that* $\mathrm{Sig}^N(X)$ *lives in a space of dimension*

$$\sum_{i=0}^{N} d^i = \frac{d^{N+1}-1}{d-1},$$

*and therefore grows exponentially with the truncation order.*

The previous proposition is key to apply signatures to machine learning methods. Indeed, as the size of the $N^{th}$-truncated signature grows exponentially with $N$, it is necessary that $\mathrm{Sig}^N(X)$ is close to $\mathrm{Sig}(X)$ for a reasonable $N$ (Fermanian, 2021).

**Theorem 11 (Universal nonlinearity)** *Let* $K$ *be a compact subset of* $BV([a,b],\mathbb{R}^d)$*. Let* $f : K \to \mathbb{R}$ *be a continuous function, and* $\epsilon > 0$*. Then there exists a linear operator* $L : T((\mathbb{R}^d)) \to \mathbb{R}$ *s.t.*

$$\forall \mathbf{x} \in K, \; |f(\tilde{X}) - L(\mathrm{Sig}(\tilde{X}))| < \epsilon$$

*with* $\tilde{X}$ *being the time-augmented path of* $X$*.*

A proof of Theorem 11 can be found in Arribas (2018). This final result is important for machine learning applications. However, two remarks should be made: 1) this is only an existence result and not a constructive one; 2) in practice, we work with truncated signatures, so that this result remains very theoretical. Indeed, linear models on the truncated signature may not be expressive enough, thus suggesting adding nonlinearities like the 1D convolution mentioned in Section 4.

**Proposition 12 (Time reparameterization invariance)** *Let* $X \in BV([a,b],\mathbb{R}^d)$ *and* $\hat{X}_t := X_{\lambda(t)}$ *with* $\lambda : [a,b] \to [a,b]$ *being a time reparameterization. Then* $\mathrm{Sig}(\hat{X}) = \mathrm{Sig}(X)$.

A proof of Proposition 12 can be found in Lyons (1998). Such proposition supports the fact that the signature transform is robust to irregular sampling, a claim already emphasized by Moreno-Pino et al. (2024).

## Appendix B. Neural Laplace and Neural Flow ResNet

In this section, we adopt the notations used in the original papers.

**Neural Laplace (Holt et al., 2022)** The Neural Laplace model is made of three consecutive steps. First, a gated recurrent unit (GRU) (Cho et al., 2014) $h_\gamma$ encodes the trajectory in a latent initial representation. Then the *Laplace representation network* $g_\beta$ learns the dynamics in the Laplace domain, and finally maps it back to the temporal domain with an inverse Laplace transform (ILT). Precisely, given a stream of data $(\mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_n})$, the map $h_\gamma$ produces a latent initial condition representation vector $\mathbf{p} \in \mathbb{R}^l$, representing the encoded version of the input trajectory

$$\mathbf{p} = h_\gamma\left((\mathbf{x}_{t_1}, t_1), \ldots, (\mathbf{x}_{t_n}, t_n)\right),$$

which is then fed to the network $g_\beta$ to obtain the Laplace transform

$$\mathcal{L}\{\mathbf{x}\}(\mathbf{s}) = v\left(g_\beta(\mathbf{p}, u(\mathbf{s}))\right),$$

where $\mathbf{s} \in \mathbb{C}^d$, $u$ is the stereographic projection and $v$ its inverse.

Then, an inverse Laplace transform step is applied to both reconstruct and extrapolate the state estimate $\hat{\mathbf{x}}(t)$ at arbitrary time $t$. Within the framework introduced in Section 3, the encoder is the network $h_\gamma$ that produces the vector $\mathbf{p}$, and the decoder is

$$(\mathbf{p}, t) \to \mathrm{ILT}\left(\mathcal{L}\{\mathbf{x}\}(\cdot), t\right)$$

with $\mathcal{L}\{\mathbf{x}\}(\cdot) = v\left(g_\beta(\mathbf{p}, u(\cdot))\right)$ and $\mathbf{p} = h_\gamma\left((\mathbf{x}_{t_1}, t_1), \ldots, (\mathbf{x}_{t_n}, t_n)\right)$.

**Neural Flow ResNet (Biloš et al., 2021)** The Neural Flow ResNet architecture is more straightforward. Given a stream of data $(\mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_n})$, a GRU-flow encoder outputs an initial state $\mathbf{z_0} \in \mathbb{R}^l$. Then, the flow in the latent space is modelled by

$$F(t, \mathbf{z_0}) = \mathbf{z_0} + \varphi(t) g(t, \mathbf{z_0}),$$

where $\varphi : \mathbb{R} \to \mathbb{R}^l$ (usually `tanh` function) and $g : \mathbb{R}^{l+1} \to \mathbb{R}^l$ is an arbitrary contractive neural network (*i.e.*, $\mathrm{Lip}(g) < 1$). Finally, a neural network is applied to map back to the original space. Within the framework introduced in Section 3, the encoder is the aforementioned RNN-based encoder, and the decoder is the composition of the flow $F$ and the mapping back to the original space.

## Appendix C. Numerical implementation and hyperparameters

Table 3: Number of parameters for the different methods and systems. Encoder refers to the encoder module (e.g., RNN-based or Signature transform); Total refers to the entire model.

| **Method** | **2D systems** | | **3D systems** | |
|---|---|---|---|---|
| | encoder | total | encoder | total |
| NODE | 0 | 19450 | 0 | 19723 |
| ANODE | 0 | 19723 | 0 | 19996 |
| Neural Laplace | 4391 | 21547 | 4454 | 25900 |
| Sig Neural Laplace | 3051 | 20207 | 4261 | 25707 |
| Neural Flow ResNet | 15064 | 18518 | 15272 | 18729 |
| Sig Neural Flow ResNet | 8059 | 11513 | 12635 | 16092 |

To ensure fair comparison, and as our purpose is to focus on the encoder choice, we tuned the models so that the number of parameters of the decoder remains constant across models (see Table 3). If it exists, we set the latent dimension to be 2. We use the Adam optimizer (Kingma and Ba, 2015) along with a learning rate of $10^{-3}$, a batch size of 128, and we train for 1000 epochs before taking the best model. Unless otherwise stated, we use the experimental code provided by Holt et al. (2022) available on the official Neural Laplace GitHub repository to implement baseline models and do the experiments. Except for Neural Laplace model, the baselines are implemented according to the original papers. For the signature transform, we rely on the Pytorch-compatible library `signatory` from Bonnier et al. (2019) available on the official Deep Signature Transforms Github repository. We refer the reader to Bonnier et al. (2019); Reizenstein and Graham (2020) for a tutorial on how to numerically compute the signature transform. All computations were performed on an `Intel(R) Core(TM) i7-14700K` CPU paired with an `NVIDIA RTX 4000` GPU, and `64 Go RAM`.

**Neural ODE (Chen et al., 2018)** We parameterize the ODE function $f$ using a 3-layer multilayer perceptron (MLP) with 136 hidden units and `tanh` activation functions. For training, the initial value is set to be the first trajectory value at the first observed time point. For extrapolation, it is the last observed trajectory value at the last observed time point. We use the 'euler' solver with the semi-norm trick (Kidger et al., 2021).

**Augmented Neural ODE (Dupont et al., 2019)** We parameterize the ODE function $f$ as a 3-layer MLP with 136 hidden units and `tanh` activation functions. One dimension, initialized to zeros, is appended to the input. For training, the initial value is set to be the first trajectory value at the first observed time point. For extrapolation, it is the last observed trajectory value at the last observed time point. We again use the 'euler' solver with the semi-norm trick (Kidger et al., 2021).

**Neural Flow ResNet (Biloš et al., 2021)** We use the model of the original paper with 26 units. For the signature encoder, the augmentation $\phi_\theta$ is a 2-layer MLP with 25 hidden units. We set the number of features to be 4, and enable both `include_original` and `include_time`. This results in an output size $e = 5+$ (dimension of the system). The sliding window size (kernel size) is set to $m = 40$. The signature is truncated to depth 3, and the projection map $g_\xi$ is a linear layer. Both encoders output mean and standard deviation of the latent initial condition within the variational framework of the original paper, and the training loss for this model is the variational loss.

**Neural Laplace (Holt et al., 2022)** The Laplace representation model is a 3-layer MLP with 42 units per layer and `tanh` activations, as in the original paper. The original encoder is a GRU with 2 layers and 21 units, with a linear layer on the final hidden state. For the signature encoder, the augmentation $\phi_\theta$ is a 2-layer MLP with 25 hidden units. We set the number of features to be 4, and enable both `include_original` and `include_time`. This results in an output size $e = 5+$ (dimension of the system). The sliding window size (kernel size) is set to $m = 40$. The signature is truncated to depth 3, and the projection map $g_\xi$ is a linear layer. Both encoders output the initial representation $p$.

## Appendix D. Sampling datasets and evaluation method

For the experiments, we use the exact same method as Holt et al. (2022). We evaluate model performance in an extrapolation setting by splitting each sampled trajectory into two equal parts $[0, T/2]$ and $[T/2, T]$. The first half is used for encoding, and the second half for prediction. For each dynamical system, 1,000 trajectories are sampled, each initialized from a different initial condition. We use a train-validation-test split of 80:10:10. For repeated experiments with the same method, we set a different random seed. Finally, all datasets are normalized using statistics computed from the training set only, to prevent data leakage (between training and test sets). To sample our datasets, we use the `ddeint` numerical solver (Zulko, 2014).

**Delayed Lotka-Volterra** We simulate trajectories of the delayed Lotka–Volterra system using a fixed delay of $\tau = 0.1$. The trajectories are computed over the interval $t \in [2, 30]$ with 1,000 time points, then subsampled uniformly to 200 time points. We vary the initial conditions over a grid in $[0.1, 2] \times [0.1, 2]$.

**Spiral DDE** We simulate trajectories of the spiral delay differential equation system using a fixed delay of $\tau = 2.5$. The matrix $A$ is set to $\begin{bmatrix} -1 & 1 \\ -1 & -1 \end{bmatrix}$. The trajectories are computed over the interval $t \in (0, 20]$ with 1,000 time points, then subsampled uniformly to 200 time points. We vary the initial conditions over a grid in $[-2, 2] \times [-2, 2]$.

**Delayed Fitzhugh-Nagumo** We simulate trajectories of the delayed FitzHugh–Nagumo system using a fixed delay of $\tau = 1$. The system parameters are set to $a = 0.5$, $b = 0.8$, $\varepsilon = 0.02$, $I = 0.5$. The trajectories are computed over the interval $t \in [2, 30]$ with 1,000 time points, and uniformly subsampled to 200 time points. We vary the initial conditions over a grid in $[-5, 5] \times [-5, 5]$.

**Delayed Rössler** We simulate trajectories of the delayed Rössler system using a fixed delay of $\tau = 2.5$. The system parameters are set to $a = 0.2$, $b = 0.2$, and $c = 4.5$. The trajectories are computed over the interval $t \in [2, 20]$ with 1,000 time points, and uniformly subsampled to 200 time points. We vary the initial conditions over a grid in $[0.1, 1.5]^3$.

## Appendix E. Coupling factor experiment

For the numerical experiment presented in Section 3, we introduce a coupling factor $\gamma > 0$ in the Delayed Fitzhugh-Nagumo model

$$\begin{aligned} \dot{x_1}(t) &= x_1(t) - \frac{x_1^3(t)}{3} - \gamma x_2(t-\tau) + I, \\ \dot{x_2}(t) &= \varepsilon\left(\gamma x_1(t) + a - b x_2(t)\right) \end{aligned}$$

As the coupling parameter $\gamma$ increases, component-wise correlations play a more significant role, since $\gamma$ directly governs the strength of the coupling. We therefore expect the performance of the signature-based model to degrade less than that of the original model. To test this hypothesis, we evaluate the extrapolation performance of both models across varying values of $\gamma$. The experimental setup is identical to that described in Appendices C and D, with the only difference being that $\gamma$ is allowed to vary.

## Appendix F. Training losses

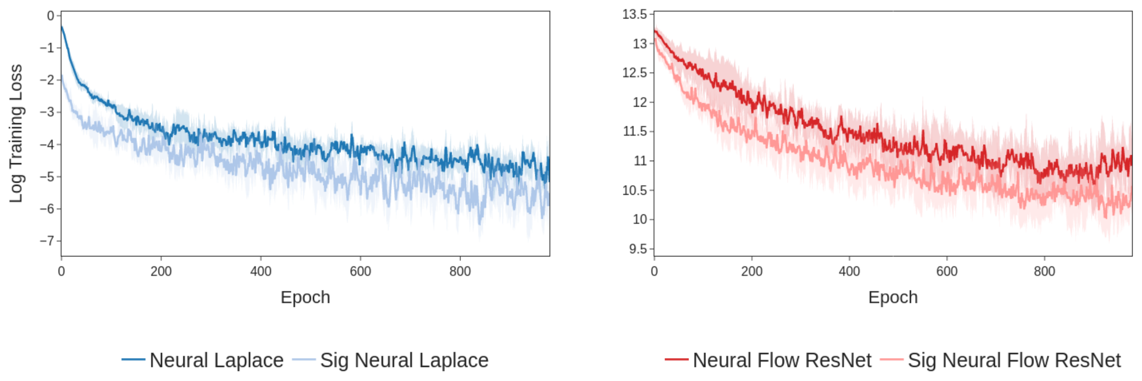

Figure 4: Training loss (log) vs. epochs averaged over 5 runs for Delayed Lotka Volterra system. Left : Neural Laplace models, right : Neural Flow ResNet models.

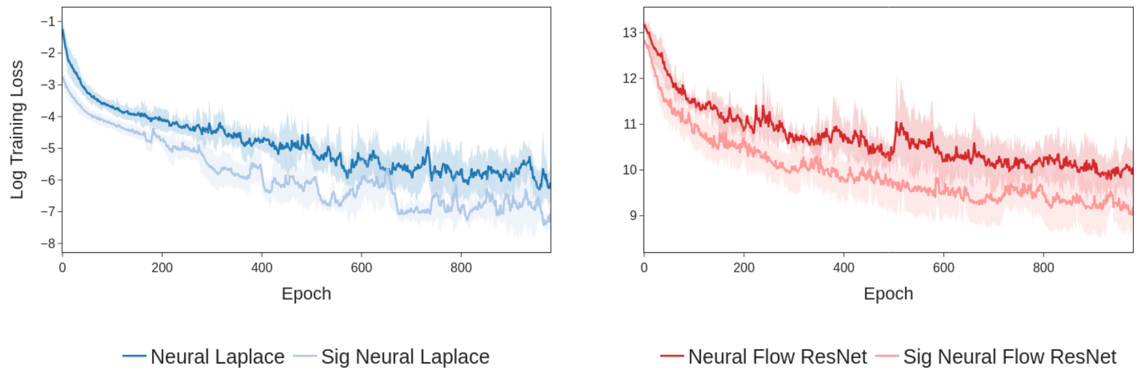

Figure 5: Training loss (log) vs. epochs averaged over 5 runs for Spiral DDE system. Left : Neural Laplace models, right : Neural Flow ResNet models.

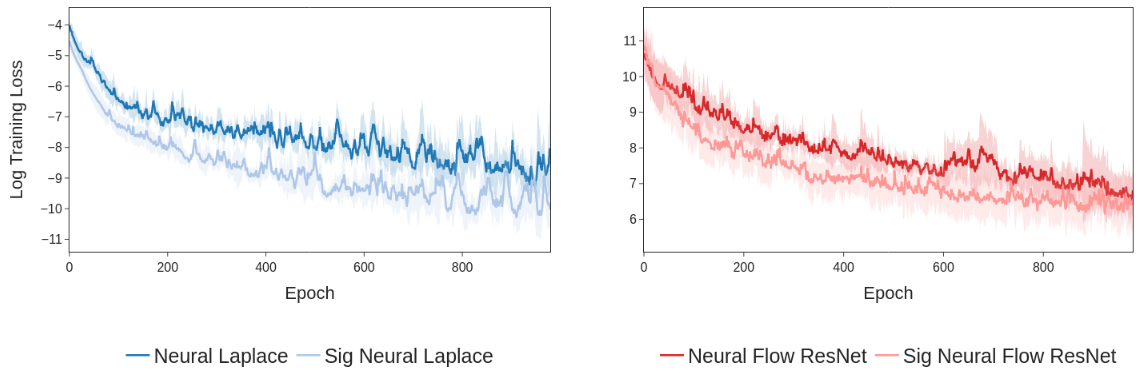

Figure 6: Training loss (log) vs. epochs averaged over 5 runs for Delayed Fitzhugh-Nagumo system. Left : Neural Laplace models, right : Neural Flow ResNet models.

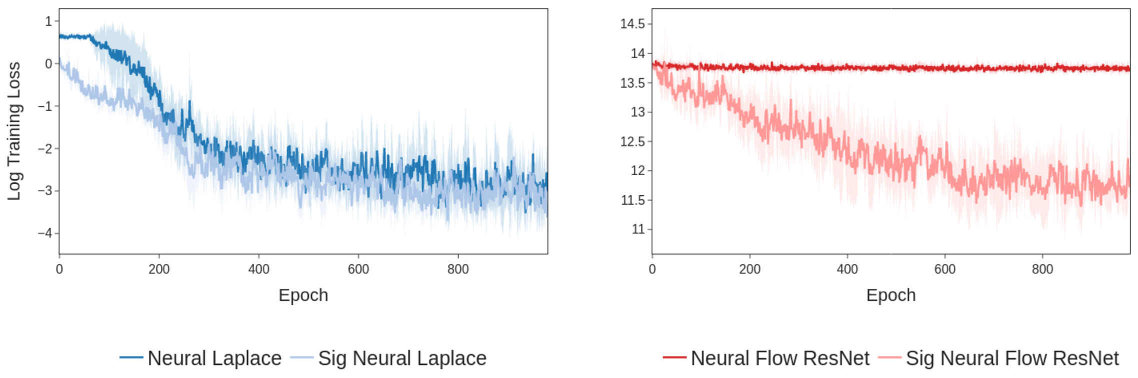

Figure 7: Training loss (log) vs. epochs averaged over 5 runs for Delayed Rössler system. Left : Neural Laplace models, right : Neural Flow ResNet models.

## Appendix G. Additional benchmarks

Table 4: Sensitivity analysis and ablation study for Sig Neural Flow ResNet. Test RMSE (mean ± std) over 5 runs. Best results bolded.

| Study | Config. | Delayed Lotka-Volterra | Spiral DDE | Delayed FitzHugh-Nagumo | Delayed Rössler |
|---|---|---|---|---|---|
| Depth $N$ | 1 | $.3423 \pm .1028$ | $.1985 \pm .0404$ | $.0601 \pm .0200$ | $.6897 \pm .1276$ |
| | 2 | $\mathbf{.2409 \pm .0665}$ | $.2026 \pm .0572$ | $.0403 \pm .0210$ | $.5810 \pm .0924$ |
| | 3 | $\mathbf{.2685 \pm .0717}$ | $\mathbf{.1892 \pm .0531}$ | $\mathbf{.0313 \pm .0081}$ | $\mathbf{.4939 \pm .0856}$ |
| $\Phi_\theta$ | $\times$ | $\mathbf{.2486 \pm .0354}$ | $.2283 \pm .0629$ | $.0504 \pm .0148$ | $.8470 \pm .2476$ |
| | ✓ | $\mathbf{.2833 \pm .0770}$ | $\mathbf{.1664 \pm .0331}$ | $\mathbf{.0393 \pm .0100}$ | $\mathbf{.5073 \pm .2330}$ |

Table 5: Mean epoch duration (in seconds). Best results per group bolded.

| **Method** | Delayed Lotka-Volterra | Spiral DDE | Delayed Fitzhugh-Nagumo | Delayed Rössler |
|---|---|---|---|---|
| ANODE (euler) | .4683 | .4720 | .4754 | .5388 |
| NODE (euler) | .4693 | .4734 | .4907 | .5369 |
| Neural Laplace | .0907 | .0927 | .0907 | .1200 |
| Sig Neural Laplace | **.0756** | **.0773** | **.0751** | **.1093** |

We measured the mean epoch duration with a batch size of 128, and we observe that Sig Neural Laplace is roughly 15% faster than Neural Laplace for 2D systems (Table 5).

Table 6: Test RMSE on Delayed Lotka-Volterra across increasing noise levels $\varepsilon$, averaged over 5 runs.

| **Method** | $\varepsilon = 0$ | $\varepsilon = 0.02$ | $\varepsilon = 0.05$ | $\varepsilon = 0.1$ |
|---|---|---|---|---|
| ANODE | $.4472 \pm .0641$ | $.3214 \pm .0391$ | $.4930 \pm .0465$ | $.5120 \pm .0282$ |
| NODE | $.6061 \pm .2521$ | $.4700 \pm .1963$ | $.6422 \pm .1847$ | $.6236 \pm .1551$ |
| Neural Laplace | $.1063 \pm .0184$ | $.0955 \pm .0174$ | $\mathbf{.1118 \pm .0102}$ | $\mathbf{.1632 \pm .0140}$ |
| Sig Neural Laplace | $\mathbf{.0541 \pm .0214}$ | $\mathbf{.0514 \pm .0169}$ | $\mathbf{.1159 \pm .0010}$ | $.1813 \pm .0135$ |
| Neural Flow ResNet | $.3360 \pm .1038$ | $\mathbf{.2774 \pm .0609}$ | $.3208 \pm .0670$ | $.3823 \pm .0487$ |
| Sig Neural Flow ResNet | $\mathbf{.2693 \pm .0343}$ | $\mathbf{.2546 \pm .0393}$ | $\mathbf{.2728 \pm .0313}$ | $\mathbf{.3108 \pm .0298}$ |

Table 7: Test RMSE on Spiral DDE across increasing noise levels $\varepsilon$, averaged over 5 runs.

| **Method** | $\varepsilon = 0$ | $\varepsilon = 0.02$ | $\varepsilon = 0.05$ | $\varepsilon = 0.1$ |
|---|---|---|---|---|
| ANODE | $.0447 \pm .0097$ | $.0631 \pm .0082$ | $.1108 \pm .0042$ | $.2113 \pm .0038$ |
| NODE | $.0590 \pm .0255$ | $.0724 \pm .0223$ | $.1182 \pm .0170$ | $.2162 \pm .0076$ |
| Neural Laplace | $.0430 \pm .0107$ | $.0508 \pm .0120$ | $.0791 \pm .0089$ | $\mathbf{.1394 \pm .0122}$ |
| Sig Neural Laplace | $\mathbf{.0265 \pm .0044}$ | $\mathbf{.0380 \pm .0038}$ | $\mathbf{.0749 \pm .0034}$ | $\mathbf{.1404 \pm .0068}$ |
| Neural Flow ResNet | $.1902 \pm .0191$ | $.1860 \pm .0059$ | $.2406 \pm .0203$ | $\mathbf{.2244 \pm .0283}$ |
| Sig Neural Flow ResNet | $\mathbf{.1317 \pm .0149}$ | $\mathbf{.1322 \pm .0184}$ | $\mathbf{.1572 \pm .0292}$ | $\mathbf{.1955 \pm .0380}$ |

## Appendix H. Dataset plots

We don't plot Neural Flow ResNet trajectories for the Delayed Rössler system as it makes the figures unreadable due to diverging trajectories.

Table 8: Test RMSE on Delayed FitzHugh–Nagumo across increasing noise levels $\varepsilon$, averaged over 5 runs. Best results per group bolded.

| Model | $\varepsilon = 0$ | $\varepsilon = 0.02$ | $\varepsilon = 0.05$ | $\varepsilon = 0.1$ |
|---|---|---|---|---|
| ANODE | $.1001 \pm .0531$ | $.1015 \pm .0412$ | $.1314 \pm .0330$ | $.1649 \pm .0318$ |
| NODE | $.0653 \pm .0226$ | $.0628 \pm .0195$ | $.0949 \pm .0254$ | $.1412 \pm .0069$ |
| Neural Laplace | $.0250 \pm .0296$ | $.0343 \pm .0240$ | $.0650 \pm .0194$ | $\mathbf{.1071 \pm .0095}$ |
| Sig Neural Laplace | $\mathbf{.0076 \pm .0016}$ | $\mathbf{.0224 \pm .0007}$ | $\mathbf{.0554 \pm .0017}$ | $\mathbf{.1070 \pm .0024}$ |
| Neural Flow ResNet | $\mathbf{.0513 \pm .0184}$ | $\mathbf{.0500 \pm .0135}$ | $\mathbf{.0742 \pm .0063}$ | $\mathbf{.1175 \pm .0099}$ |
| Sig Neural Flow ResNet | $\mathbf{.0401 \pm .0235}$ | $\mathbf{.0451 \pm .0173}$ | $\mathbf{.0777 \pm .0100}$ | $.1426 \pm .0085$ |

Table 9: Test RMSE on FitzHugh–Nagumo across decreasing input sequence length $n$, averaged over 5 runs. Best results per group bolded.

| Model | $n = 100$ | $n = 80$ | $n = 70$ | $n = 50$ |
|---|---|---|---|---|
| ANODE | $.1001 \pm .0531$ | $.1070 \pm .0523$ | $.0954 \pm .0530$ | $.0975 \pm .0559$ |
| NODE | $.0653 \pm .0226$ | $.0826 \pm .0338$ | $.0528 \pm .0172$ | $.0557 \pm .0187$ |
| Neural Laplace | $.0250 \pm .0296$ | $.0528 \pm .0196$ | $.0537 \pm .0230$ | $.1120 \pm .0229$ |
| Sig Neural Laplace | $\mathbf{.0076 \pm .0016}$ | $\mathbf{.0159 \pm .0036}$ | $\mathbf{.0197 \pm .0025}$ | $\mathbf{.0449 \pm .0106}$ |
| Neural Flow ResNet | $.0513 \pm .0185$ | $.0484 \pm .0121$ | $\mathbf{.0453 \pm .0143}$ | $\mathbf{.0687 \pm .0244}$ |
| Sig Neural Flow ResNet | $\mathbf{.0401 \pm .0235}$ | $\mathbf{.0388 \pm .0225}$ | $\mathbf{.0446 \pm .0212}$ | $\mathbf{.0684 \pm .0216}$ |

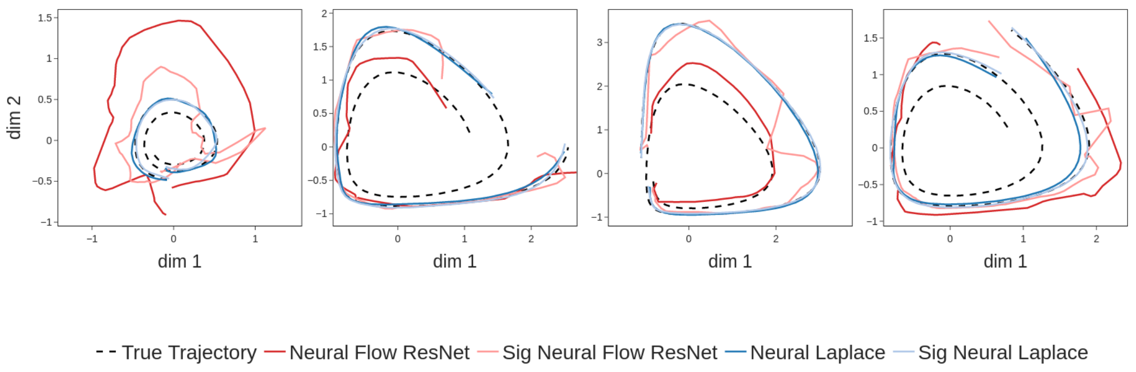

Figure 8: Delayed Lotka-Volterra randomly sampled test trajectory.

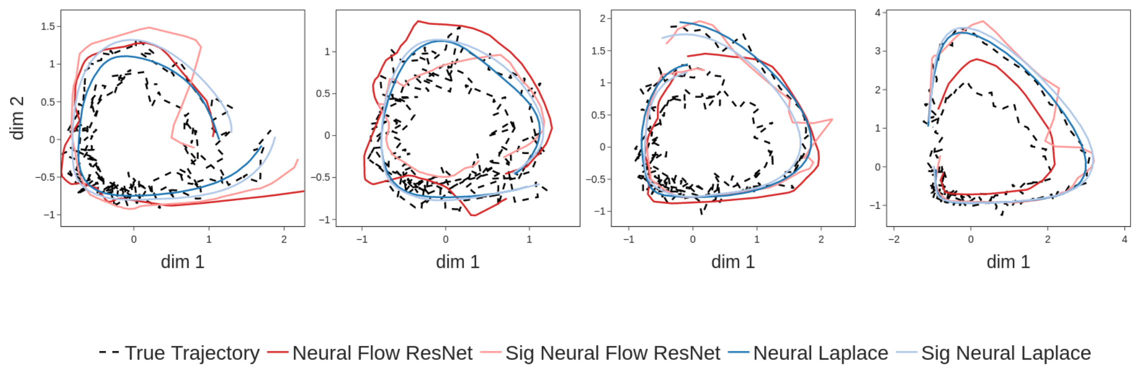

Figure 9: Delayed Lotka-Volterra randomly sampled test trajectory, with gaussian noise $\mathcal{N}(0, 0.1)$ added.

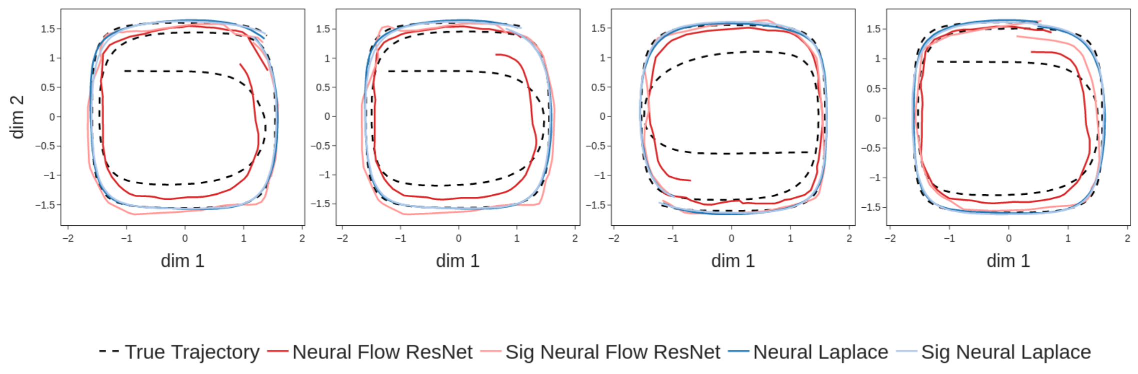

Figure 10: Spiral DDE randomly sampled test trajectory.

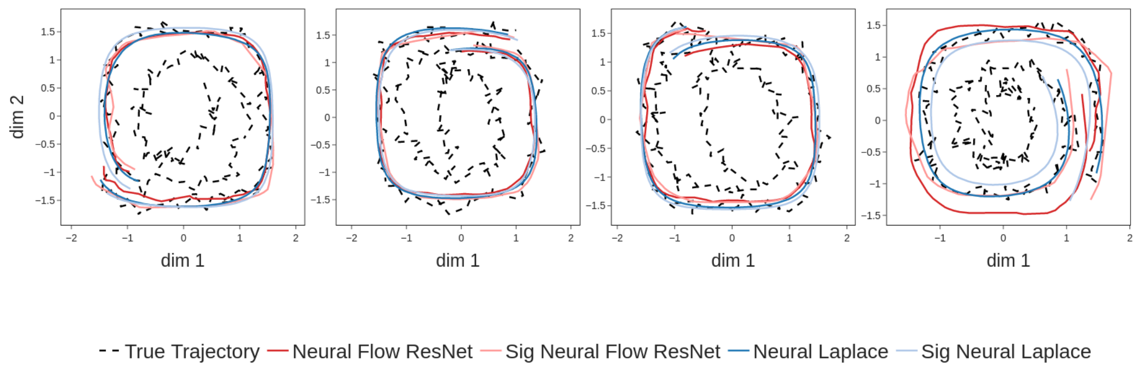

Figure 11: Spiral DDE randomly sampled test trajectory, with gaussian noise $\mathcal{N}(0, 0.1)$ added.

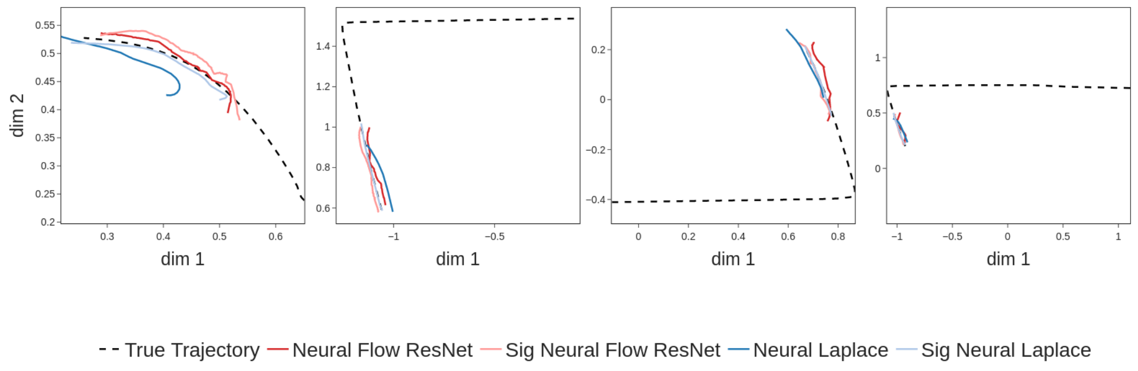

Figure 12: Delayed Fitzhugh-Nagumo randomly sampled test trajectory.

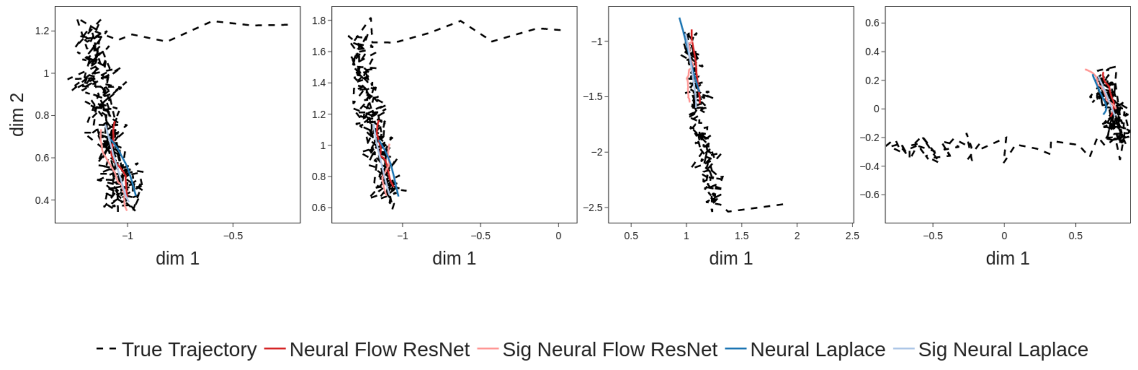

Figure 13: Delayed Fitzhugh-Nagumo randomly sampled test trajectory, with gaussian noise $\mathcal{N}(0, 0.05)$ added.

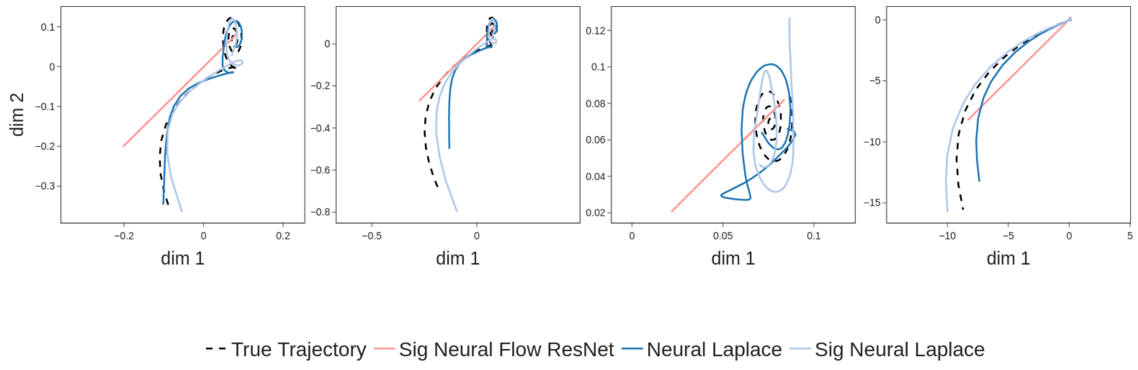

Figure 14: Delayed Rössler (first two dimensions) randomly sampled test trajectory.

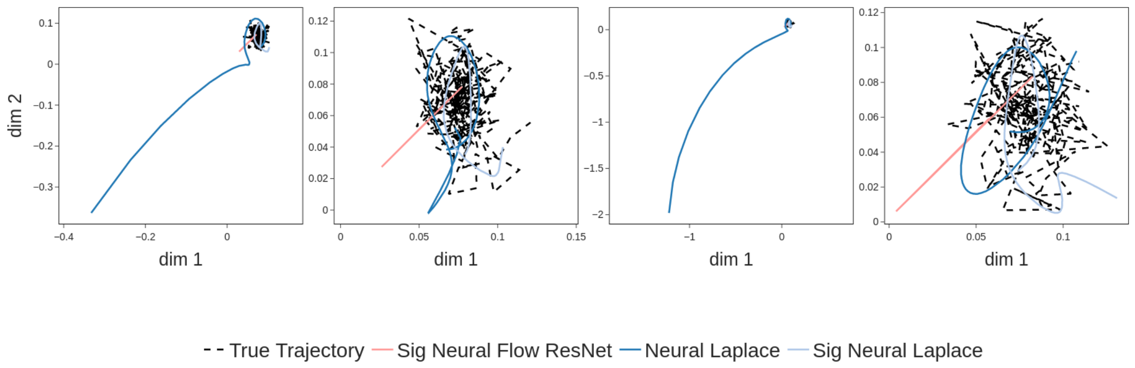

Figure 15: Delayed Rössler (first two dimensions) randomly sampled test trajectory, with gaussian noise $\mathcal{N}(0, 0.01)$ added.