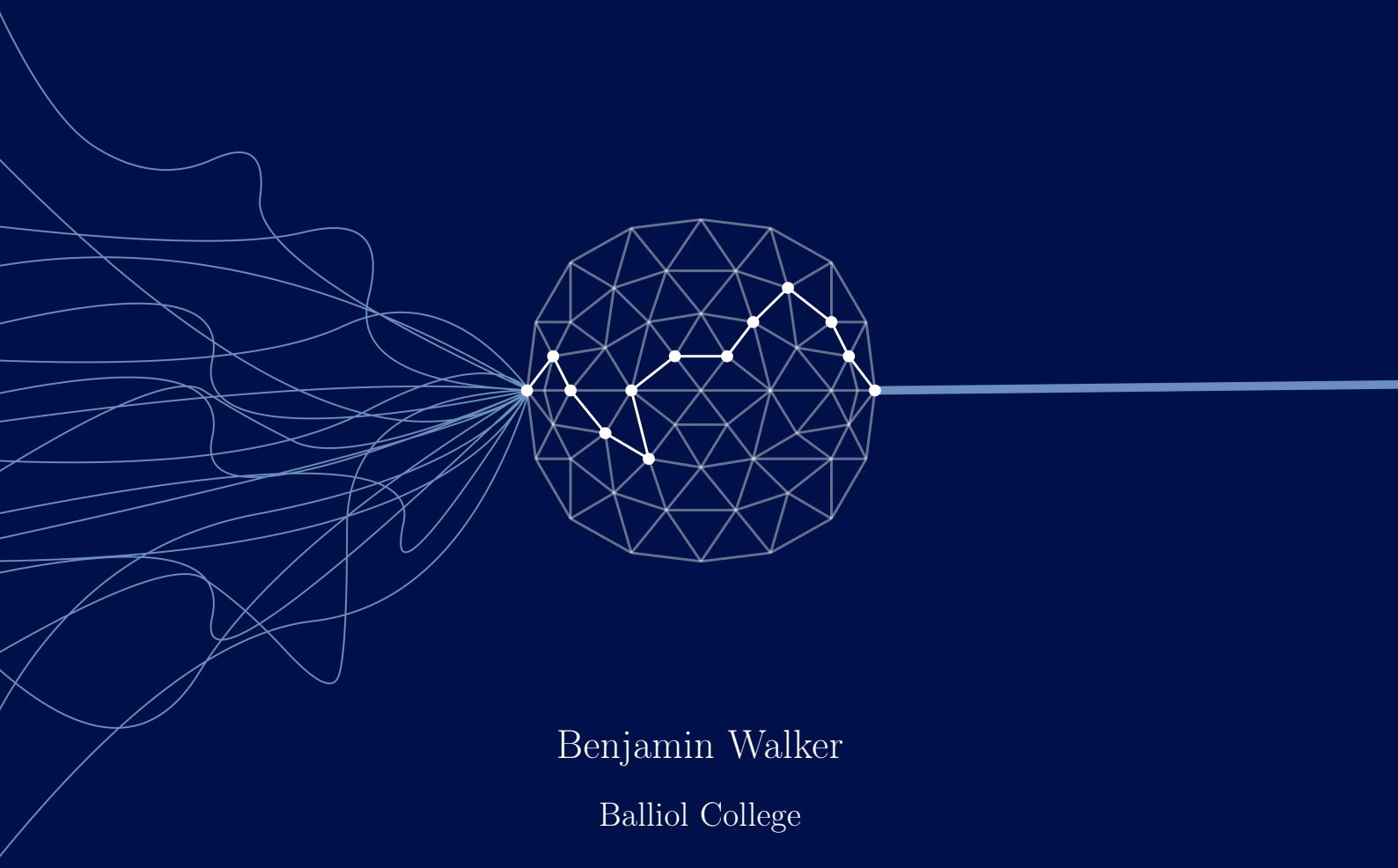
Advances in Neural Controlled Differential Equations



Benjamin Walker

Balliol College

University of Oxford

UNIVERSITY OF OXFORD

A thesis submitted for the degree of

Doctor of Philosophy

November 2025

To Mum and Dad, for everything.

Acknowledgements

Terry, it has been a privilege spending an hour (and often much longer) each week discussing mathematics, machine learning, finance, politics, philosophy, and every other topic we wandered into. Although you "cannot discuss mathematics with someone who doesn't understand differential geometry," you certainly did your best with me. I would not be the researcher I am today without you.

Mum and Dad, I will never be able to express the gratitude I have for your continuous love, encouragement, and guidance. From Mum spending hours revising with me to Dad patiently reminding me for the hundredth time that there are two solutions to $x^2 = 4$, I owe every step of this journey to you both.

Sophie, Naomi, and all of my friends, thank you. From adventures Down Under, to European hiking trips and weekend getaways in London, Selsey, Wales, and Milton Keynes, from late-night board games (and those who had to put up with them) to some truly wonderful Christmas meals. I cannot imagine having done this without your support and friendship.

Thank you to my collaborators and colleagues, Lingyi Yang, Nicola Muça Cirone, Cris Salvi, Christian Bayer, Andrew McLeod, Felix Krones, Adam Mahdi, Tiexin Qin, Haoliang Li, Cora Cartis, Kate Zhu, Ammar Naseer, Torben Berndt, Alexandre Bloch, Sam Morley, and Elena Gal. This thesis, and I personally, owe much to the time, effort, and enthusiasm you have so generously given.

Thank you also to my examiners, Marc Deisenroth and Stephen Roberts, for your time, care, and thoughtful feedback on this thesis.

Finally, Nathalie, the best part of this journey has been sharing it with you. I cannot wait for our next adventure.

There are many human behaviours which unfold over time. It would be folly to try to understand those behaviours without taking into account their temporal nature.

—Jeffrey Elman, *Finding Structure in Time* (1990)

Abstract

Many real-world systems evolve continuously, yet most machine learning models interpret time series as discrete sequences. Continuous-time approaches instead treat time series as samples from an underlying input path, a formulation that naturally accommodates irregularly sampled or oversampled data. Among these, Neural Controlled Differential Equations (NCDEs) are a maximally expressive class of models that parametrise a vector field using a neural network and evolve their hidden state by solving a dynamical system driven by the input path. NCDEs typically use a non-linear vector field, so their expressive power and continuous-time flexibility come at the cost of a forward pass that is both computationally expensive and inherently sequential, limiting their scalability and practical applicability.

This thesis advances the training and scalability of NCDEs through three complementary contributions. First, building on neural rough differential equations, Log-NCDEs apply the Log-ODE method to efficiently approximate an NCDE's solution during training, improving both computational speed and empirical performance. Second, Linear NCDEs replace the non-linear vector field with a linear one, enabling closed-form solutions and parallel-in-time computation without sacrificing theoretical expressivity. Third, Structured Linear NCDEs use structured linear vector fields to further enhance efficiency while maintaining theoretical expressiveness and empirical performance.

Collectively, these methods reduce the time per training step for an NCDE by up to three orders of magnitude while achieving state-of-the-art performance across diverse time series benchmarks.

Contents

Chapter 1

Introduction

Paths — simply everywhere.

—Terry Lyons, *Rough Paths, Signatures and the Modelling of Functions on Streams* (2014)

1.1 Paths

1.1.1 The PhysioNet Challenge 2022

In the first year of my PhD, I took part in the George B. Moody PhysioNet Challenge 2022, an international machine learning competition focused on developing open-source solutions to clinically relevant problems in healthcare [Goldberger et al. 2000; Reyna et al. 2023]. The task was to detect heart murmurs from phonocardiogram (PCG) recordings. Each patient contributed up to six recordings, with between 20,000 and 180,000 samples per recording. Our team, *PathToMyHeart*, placed fourth out of 78 total teams [Walker et al. 2022].

Our approach converted PCGs into log-mel spectrograms and applied a fine-tuned ResNet-50, which had been pre-trained on ImageNet [Deng et al. 2009; He et al. 2016; Walker et al. 2022]. Notably, the five best-performing teams all used spectrograms, and four, including the winner, applied convolutional neural networks to those spectrograms [Xu et al. 2022; Lee et al. 2022; Lu et al. 2022; McDonald et al. 2022]. Despite heart-sound recordings being time series, the dominant approach among the best-performing teams was to change the modality: move from time series to images and apply a convolutional neural network. This is particularly interesting given

the concurrent success of Transformer-based sequence models [Bahdanau et al. 2015; Vaswani et al. 2017; Brown et al. 2020].

There are many reasons why a model operating on spectrograms could outperform one operating on the raw signal. One that stands out is the application of discrete sequence models to samples from a continuous process. For example, assuming the signal is band-limited, then a frequency-cropped spectrogram is theoretically independent of the sampling rate r once it exceeds twice the signal's highest frequency [Nyquist 1928; Shannon 1949]. Therefore, past this sampling rate, the downstream model's performance is independent of r. In contrast, applying a recurrent neural network (RNN) to the raw signal becomes increasingly unstable as r increases due to exploding or vanishing gradients, while applying a Transformer incurs a computational cost that scales as $\mathcal{O}(r^2)$ [Hochreiter 1991; Hochreiter et al. 1997; Vaswani et al. 2017]. Both challenges stem from treating samples of a continuous signal as a sequence of discrete observations. A more natural approach is to model the data as a path, where the evolution is described continuously.

1.1.2 From Paths to Signatures

This thesis is certainly not the first work to advocate for a path-based approach to time series modelling. Differential equations have been used to model epidemics [Kermack et al. 1927], economics [Solow 1956], and ecology [Lotka 1925; Volterra 1926], to name some of the applications beginning with the letter e. In fact, neural networks were used to parametrise the vector field of a differential equation and trained to output paths before the foundation of modern recurrent neural networks [Pearlmutter 1989; Elman 1990].

The approach to path-based modelling this thesis builds upon can be traced back to Chen [1954], who introduced an infinite collection of iterated integrals of a path, now known as the path's signature. This object has several important mathematical properties, which are discussed in detail in Section 2.3. One important feature for time series modelling is that, modulo an equivalence relation discussed in Section 2.3.2, the signature determines a path uniquely [Hambly et al. 2010; Boedihardjo et al. 2016]. This uniqueness, together with the natural grading of the signature terms, motivates the truncated signature as a finite feature set that provides a high-level description of the path over an interval. Early applications of the truncated

signature include handwritten character recognition [Graham 2013; Yang et al. 2016] and the extraction of information from financial data streams [Gyurkó et al. 2014]. Another important feature for time series modelling is that any real-valued continuous function defined on a compact set of signatures can be approximated arbitrarily well by a linear function. Combined with uniqueness, this property implies that continuous functions on compact subsets of path space can be approximated by linear functions of the signature. This motivates the use of truncated signature features in linear regression, as formalised by Levin et al. [2016]. A more recent development has been the introduction of the signature kernel, which allows one to operate with the complete signature [Király et al. 2019; Salvi et al. 2021; Salvi 2021; Lemercier et al. 2021b; Lemercier et al. 2021a; Manten et al. 2025]. In healthcare, signature methods have been applied to distinguishing between bipolar disorder and borderline personality disorder [Perez Arribas et al. 2018], diagnosing Alzheimer's disease [Moore et al. 2019], speech emotion recognition [Wang et al. 2019], early detection of sepsis [Morrill et al. 2019; Cohen et al. 2024], and heart failure prediction from electronic health records [Vauvelle et al. 2022]. In finance, signature methods have been applied to derivative pricing [Perez Arribas 2018], path-dependent stochastic models [Perez Arribas et al. 2020], optimal stopping [Horvath et al. 2023], hedging [Cirone et al. 2025c], and macroeconomic nowcasting [Cohen et al. 2023]. Additionally, signature methods have proven valuable in information theory [Salvi et al. 2023; Shmelev et al. 2024], cybersecurity [Cochrane et al. 2021], and computational neuroscience [Holberg et al. 2024].

The signature has typically been used as a shallow learning tool. A predefined, universal transformation is applied to a path, with the feature extractor being independent of any specific task or dataset. Consequently, learning is confined to a final map from the signature features to the desired output. Even when signatures are incorporated into deep learning architectures, the transformation itself is often kept static, with learnable components occurring before or after the signature computation [Bonnier et al. 2019; Liao et al. 2021; Moreno-Pino et al. 2024]. As will be discussed in Section 2.4.1, the truncated signature is the solution to a certain controlled differential equation (CDE). This perspective naturally raises the question: rather than using the fixed signature as a feature extractor, can we instead learn a task-specific CDE?

1.1.3 Neural Controlled Differential Equations

A CDE describes the relationship between the increments of a control path and the evolution of a solution path using a vector field. Neural CDEs (NCDEs) treat time series as observations from a control path, parametrise a CDE's vector field using a neural network, and use the solution path as a continuously evolving hidden state [Kidger et al. 2020]. An immediate benefit of this path-based approach is detaching the sampling rate of the data from how we evolve the hidden state, which is now controlled by the choice of differential equation solver. This makes NCDEs naturally suited to handling irregularly sampled or oversampled data [Kidger et al. 2020; Walker et al. 2024]. They have been applied to counterfactual prediction in healthcare [Seedat et al. 2022], economic nowcasting [Lim et al. 2024], survival prediction [Zeng et al. 2025], anomaly detection for driving assistance [Lee et al. 2024], and dynamic graphs [Qin et al. 2025; Berndt et al. 2025], with particular success in traffic forecasting [Choi et al. 2022; Choi et al. 2023].

However, despite being aware of this path-based approach, our team did not attempt to train an NCDE to detect heart murmurs from PCG recordings during the PhysioNet Challenge 2022. Each forward pass involves using a differential equation solver to approximate the solution of a non-linear CDE. This process is not only computationally expensive, but also inherently sequential, preventing parallel-in-time training. Furthermore, although some progress has been made on time series with many samples using neural rough differential equations (NRDEs) [Morrill et al. 2021], NCDEs continue to struggle on these tasks.

Addressing these limitations is the central focus of the work presented in this thesis.

1.2 Thesis Outline

1.2.1 Contributions

This thesis makes three key contributions to the study and application of NCDEs:

- First, building on the work of NRDEs [Morrill et al. 2021], we use the Log-ODE method to approximate the solutions of NCDEs during training. Rather than working directly with the raw path, this approach uses the signature to summarise the path over intervals, yielding a more efficient approximation. This

reduces training times and improves empirical performance, but it does not remove the core computational bottleneck: the forward pass still requires solving a non-linear differential equation sequentially.

- Second, we introduce Linear NCDEs, a variant where the non-linear vector field is replaced by a linear one. Linear NCDEs retain the theoretical expressivity of NCDEs while admitting explicit solutions, removing the need for a differential equation solver and enabling parallel-in-time training. Linearity is the key structural innovation that makes NCDEs scalable.

- Third, we develop Structured Linear NCDEs (SLiCEs), which replace the dense matrices of a Linear NCDE with structured variants that preserve the model's expressivity while further improving computational efficiency.

These three advances are complementary: the Log-ODE method provides an efficient path-based approximation, Linear NCDEs provide a scalable model architecture, and SLiCEs improve the computational efficiency. Together, they reduce the time per training step for NCDEs by up to three orders of magnitude, making continuous-time models applicable at scales that were previously infeasible. The core of this thesis brings together and extends three publications, with my specific contributions to each detailed below.

- B. Walker et al. [2024]. "Log Neural Controlled Differential Equations: The Lie Brackets Make a Difference", *Proceedings of the 41st International Conference on Machine Learning (ICML)*

  - Developed the methodology, implementation, and empirical validation of Log-NCDEs presented in Section 4.4.
  - The proof of Lemma 3.35 presented in Section 3.4.2, which explicitly bounds the $\mathrm{Lip}(\gamma)$-norm for the composition of two $\mathrm{Lip}(\gamma)$ functions when $1 < \gamma \leq 2$. This proof was completed collaboratively with Dr. Andrew McLeod.
  - The proof of Theorem 4.8 presented in Section 4.4.2, which uses Lemma 3.35 to bound the $\mathrm{Lip}(\gamma)$-norm for a specific class of neural networks when $1 < \gamma \leq 2$.

- N. M. Cirone et al. [2024]. "Theoretical Foundations of Deep Selective State-Space Models", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*

  - Developed the methodology in collaboration with Nicola Muça Cirone.
  - Solely contributed the implementation and empirical validation of Linear NCDEs presented in Sections 5.3.4 and 5.3.5.

- B. Walker et al. [2025]. "Structured Linear CDEs: Maximally Expressive and Parallel-in-Time Sequence Models", *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS) (Spotlight)*

  - Developed the concept, methodology, implementation, and empirical validation of SLiCEs presented in Section 5.4.

All of my published work has been completed in close collaboration with my co-authors. In particular, I want to highlight Nicola Muça Cirone, who proved Theorems 5.4, 5.5, and 5.8, originally presented in [Cirone et al. 2024], and Theorem 5.9, originally presented in [Walker et al. 2025]. These theorems are presented in this thesis to give a complete picture, with original overviews of the proof techniques provided where appropriate. This thesis also presents the following previously unpublished contributions:

- To the best of our knowledge, Section 3.3 provides the first formal treatment of the Lie bracket for two $\mathrm{Lip}(\gamma)$ functions defined on arbitrary subsets of potentially infinite-dimensional Banach spaces.

- Section 3.4.3 provides an explicit example that lower bounds the norm of the composition of two $\mathrm{Lip}(\gamma)$ functions when $1 < \gamma \leq 2$. This complements the upper bound in Lemma 3.35 originally presented in Walker et al. 2024. We also show that these two bounds converge as $\gamma \to 1$.

- Theorem 4.8 generalises [Walker et al. 2024, Theorem 3.1] by extending the proof that certain neural networks are $\mathrm{Lip}(\gamma)$ from the specific case of Sigmoid Linear Unit (SiLU) activation functions and $\gamma = 2$ to a broader class of activation functions and $1 < \gamma \leq 2$ [Elfwing et al. 2018].

1.2.2 Organisation

This thesis is structured into two parts. The first half establishes the mathematical foundations for the continuous-time models developed later.

- Chapter 2 introduces the prerequisite mathematics. Topics covered include the tensor algebra, the signature of a path, existence and uniqueness of solutions to CDEs, and the Log-ODE method for approximating solutions to CDEs. These concepts provide the foundation for the path-based approach adopted in this thesis: paths are the fundamental objects, signatures give a principled description of a path over an interval, and the Log-ODE method shows how such descriptions can be used to efficiently approximate continuous-time dynamics.

- Chapter 3 introduces $\mathrm{Lip}(\gamma)$ regularity, which specifies the assumptions on the vector field of a CDE needed to establish existence and uniqueness of solutions. The chapter also provides a formal treatment of the Lie bracket for $\mathrm{Lip}(\gamma)$ functions, a key component of the Log-ODE method. Finally, a novel explicit bound on the $\mathrm{Lip}(\gamma)$ norm of the composition of two $\mathrm{Lip}(\gamma)$ functions when $1 < \gamma \leq 2$ is presented. These results provide the regularity theory needed to justify the application of the Log-ODE method to NCDEs.

The second half of the thesis uses these mathematical foundations to develop scalable continuous-time models.

- Chapter 4 begins by introducing NCDEs. The results of Chapter 3 are used to ensure that the neural network parametrising the NCDE's vector field satisfies $\mathrm{Lip}(\gamma)$ regularity, allowing the application of the Log-ODE method. The chapter then shows that using the Log-ODE method during training improves the empirical performance and computational efficiency of NCDEs across a range of real-world time series benchmarks. The chapter concludes by highlighting a key remaining limitation of this approach: because the dynamics are governed by a non-linear differential equation, the forward pass remains inherently sequential.

- Chapter 5 introduces Linear NCDEs, a subclass of NCDEs with vector fields that depend linearly on the hidden-state. Linear NCDEs are shown to retain the full theoretical expressivity of NCDEs whilst admitting explicit solutions, removing the need for a differential equation solver and enabling parallel-in-time computation. The chapter also shows that modern Structured State-Space Models (SSMs) can be viewed as a restrictive subclass of Linear NCDEs, allowing a formal characterisation of their expressive limitations. Finally, the chapter introduces SLiCEs, which replace the dense matrices of a Linear NCDE with structured variants that reduce the computational burden whilst retaining full theoretical expressivity and achieving equivalent empirical performance.

- Chapter 6 concludes the thesis by summarising the key contributions, reflecting on their implications for scalable continuous-time models, and discussing promising avenues for future research.

Chapter 2

Controlled Differential Equations

> *The winding roads must be made straight and the rough paths made smooth.*
>
> —Luke 3:5, *Good News Translation*

2.1 Introduction

Realising the benefits of continuous-time machine learning requires a mathematical framework for understanding continuous paths. This chapter develops that framework by introducing CDEs and the signature of a path. These objects are closely connected. The signature arises naturally when deriving the flow of a linear CDE, as shown in Theorem 2.37, and is itself the solution of a CDE. Furthermore, the signature has several properties that make it central to the methods developed in this thesis. In particular, it provides a graded representation of a path, its components form a universal feature set, and it composes associatively under concatenation of intervals, enabling parallel-in-time computation via an associative scan.

These ideas already appear in the simplest one-dimensional example, the signature of the time path,

$$X_t = X(t) = t \in \mathbb{R}, \tag{2.1}$$

for $t \in [a, b]$. The signature of a one-dimensional path over an interval $[a, t]$ is given by an infinite collection of real-valued numbers,

$$S_{[a,t]} = [S^0_{[a,t]}, S^1_{[a,t]}, S^2_{[a,t]}, \ldots, S^i_{[a,t]}, \ldots], \tag{2.2}$$

where the superscript $i \in \mathbb{N}_0 = \{0\} \cup \mathbb{N}$ denotes the i^{th} component of the signature and the subscript $[a,t]$ records the interval over which it is computed. For the time path, the components of the signature are defined recursively by the system of ordinary differential equations

$$\mathrm{d}S^i_{[a,t]} = S^{i-1}_{[a,t]}\,\mathrm{d}t, \tag{2.3}$$

where $S^0_{[a,t]} = 1$ for all $t \in [a,b]$ and $S^i_{[a,a]} = 0$ for $i \geq 1$. The solution to this system is

$$S_{[a,t]} = \left[1, t-a, \frac{(t-a)^2}{2!}, \ldots, \frac{(t-a)^i}{i!}, \ldots\right] \tag{2.4}$$

for $t \in [a,b]$. The components of the time path's signature are scaled monomials corresponding to the terms in the exponential series. Hence, the sequence of partial sums $\sum_{i=0}^{N} S^i_{[a,t]}$ converges uniformly to the exponential,

$$\sup_{t\in[a,b]} \left| \mathrm{e}^{t-a} - \sum_{i=0}^{N} S^i_{[a,t]} \right| \xrightarrow{N\to\infty} 0. \tag{2.5}$$

More generally, the signature retains this graded, exponential-like structure for arbitrary paths, although in higher dimensions its components no longer take values in $\mathbb{R}$, but in different tensor spaces, as will be seen in Section 2.3.

Since the components of the time path's signature are scaled monomials, a straightforward application of the Weierstrass Approximation Theorem shows that any continuous function on $[a,b]$ can be approximated to arbitrary precision by a linear combination of the terms of $S_{[a,t]}$.

Theorem 2.1 (Weierstrass Approximation Theorem [Weierstrass 1885])**.** *Let $f : [a,b] \to \mathbb{R}$ be a continuous function. Then, for every $\epsilon > 0$, there exists a polynomial P such that*

$$\sup_{t\in[a,b]} |f(t) - P(t)| < \epsilon. \tag{2.6}$$

Corollary 2.2. *Let $S_{[a,t]} = [1, (t-a), \frac{1}{2!}(t-a)^2, \ldots, \frac{1}{i!}(t-a)^i, \ldots]$ for $t \in [a,b]$. Then, for any continuous function $f : [a,b] \to \mathbb{R}$ and any $\epsilon > 0$, there exists a finite sequence of coefficients $L = [L_0, L_1, \ldots, L_n]$ such that*

$$\sup_{t\in[a,b]} \left| f(t) - \sum_{i=0}^{n} L_i S^i_{[a,t]} \right| < \epsilon. \tag{2.7}$$

Proof. The elements $S^i_{[a,t]}$ are scaled monomials, and their finite linear combinations form the polynomials. Therefore, by the Weierstrass Approximation Theorem, any

continuous function can be uniformly approximated by a finite sum of $S^i_{[a,t]}$ to arbitrary precision. □

Hence, linear maps of the time path's signature are dense in the space of continuous functions of time. Section 2.3 will extend this same universality from continuous functions of time to continuous functions of paths.

We can define a product of $S_{[a,b]}$ and $S_{[b,c]}$, which corresponds to concatenating the intervals $[a, b]$ and $[b, c]$.

Lemma 2.3 (One-Dimensional Concatenation Product)**.** *For intervals* $[\alpha, \beta]$ *and* $[\gamma, \delta]$*, let* $*$ *denote the concatenation product defined by*

$$(S_{[\alpha,\beta]} * S_{[\gamma,\delta]})^k = \sum_{j=0}^{k} S^j_{[\alpha,\beta]} S^{k-j}_{[\gamma,\delta]}, \tag{2.8}$$

for $k \in \mathbb{N}_0$*. Then, for any* $a \leq b \leq c$*,*

$$S_{[a,c]} = S_{[a,b]} * S_{[b,c]}. \tag{2.9}$$

Proof. Using (2.8),

$$(S_{[a,b]} * S_{[b,c]})^k = \sum_{j=0}^{k} S^j_{[a,b]} S^{k-j}_{[b,c]} \tag{2.10}$$

$$= \sum_{j=0}^{k} \frac{(b-a)^j (c-b)^{k-j}}{j!(k-j)!}, \tag{2.11}$$

$$= \frac{1}{k!} \sum_{j=0}^{k} \binom{k}{j} (b-a)^j (c-b)^{k-j} \tag{2.12}$$

Applying the Binomial Theorem,

$$(S_{[a,b]} * S_{[b,c]})^k = \frac{(c-a)^k}{k!} = S^k_{[a,c]}. \tag{2.13}$$

Since the k-th components of $S_{[a,c]}$ and $S_{[a,b]} * S_{[b,c]}$ are equal for all $k \geq 0$,

$$S_{[a,c]} = S_{[a,b]} * S_{[b,c]}. \tag{2.14}$$

□

This simple setting already demonstrates an important computational advantage of signatures. To compute the signature over a long interval, one may first partition

the interval into smaller subintervals, compute the signature on each subinterval independently, and then combine the results using the product. Since this product is associative, these local computations can be aggregated in parallel via an associative scan [Blelloch 1993]. The same idea will also be used in Chapter 5 to make Linear NCDEs parallel-in-time.

The one-dimensional case already illustrates the core properties of the signature that make it a powerful tool. It provides a universal, graded representation of a path that composes naturally under concatenation of intervals. For general paths, these components are no longer real-valued, but instead take values in tensor spaces. Formulating this precisely requires a brief detour into the tensor algebra.

2.2 The Tensor Algebra

2.2.1 Tensor Product Space

For paths taking values in a vector space, the higher levels of the signature are defined by iterated integrals of the path increments. At the second level, these terms involve pairs of increments and so naturally define bilinear quantities. Higher levels similarly involve several increments and hence give rise to multilinear quantities. To treat such objects within a linear framework, we seek a vector space through which these bilinear interactions may be represented linearly. This motivates the following universality property.

Definition 2.4 (Universal for Bilinearity [Roman 2007])**.** *Let U and V be vector spaces. A vector space T with a bilinear map $t : U \times V \to T$ is universal for bilinearity if for every vector space W and every bilinear map $\kappa : U \times V \to W$, there exists a unique linear map $\tau : T \to W$ satisfying $\kappa = \tau \circ t$.*

We now construct a vector space with this property. Let U and V be vector spaces over F, and let $F_{U\times V}$ be the vector space with basis $\{(u, v) : u \in U, v \in V\}$. Let R be the subspace of $F_{U\times V}$ spanned by the expressions

$$\begin{aligned} &(u + u', v) - (u, v) - (u', v), \\ &(ru, v) - r(u, v), \\ &(u, v + v') - (u, v) - (u, v'), \\ &(u, rv) - r(u, v), \end{aligned} \tag{2.15}$$

where $r \in F$, $u, u' \in U$, and $v, v' \in V$. These expressions measure the failure of bilinearity, so passing to the quotient identifies them with zero and thereby forces bilinearity to hold. The quotient space

$$U \otimes V = \frac{F_{U \times V}}{R}, \tag{2.16}$$

is called the tensor product of U and V. The map $\iota : U \times V \to U \otimes V$ is defined by projecting (u, v) onto $U \otimes V$, with the result denoted $\iota(u, v) = u \otimes v$.

Theorem 2.5. *Let U and V be vector spaces. Up to isomorphism, the tensor product space $U \otimes V$ with the bilinear map $\iota : U \times V \to U \otimes V$ is the unique space that is universal for bilinearity [Roman 2007].*

Theorem 2.5 shows that the space constructed above is, up to isomorphism, the only vector space with the required universality property. Thus the tensor product is the natural space for bilinear quantities. Repeating this construction yields the higher tensor powers needed for the multilinear terms appearing in the signature.

Figure 2.1 is a schematic representation of the universal bilinearity satisfied by the tensor product space. To illustrate the tensor product and universal property, take $U = \mathbb{R}^2$ and $V = \mathbb{R}^3$ with bases (e_1, e_2) and (f_1, f_2, f_3), respectively. Let $\Phi : U \otimes V \to \mathbb{R}^{2\times 3}$ be the isomorphism defined by mapping $e_i \otimes f_j$ to the matrix E_{ij}, which has a 1 at (i, j) and zeros elsewhere. Let $u = [u_1, u_2]^T \in \mathbb{R}^2$ and $v = [v_1, v_2, v_3]^T \in \mathbb{R}^3$. The image of their tensor product under Φ is the outer product of the vectors,

$$\Phi(u \otimes v) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \end{bmatrix}. \tag{2.17}$$

Every bilinear expression $\kappa(u, v)$ depends on products $u_i v_j$ of components of u and v. In $U \otimes V$, these products are represented linearly by the basis elements $e_i \otimes f_j$. Thus, given a bilinear map $\kappa : U \times V \to W$, the universal property yields a unique linear map $\tau : U \otimes V \to W$ such that $\tau(u \otimes v) = \kappa(u, v)$.

2.2.2 Definition

The previous subsection introduced tensor product spaces as the natural linear spaces for the multilinear terms in the signature. To study the signature analytically, we now assume that the path takes values in a Banach space V. The higher levels of the signature then take values in the iterated tensor products of V. To analyse these

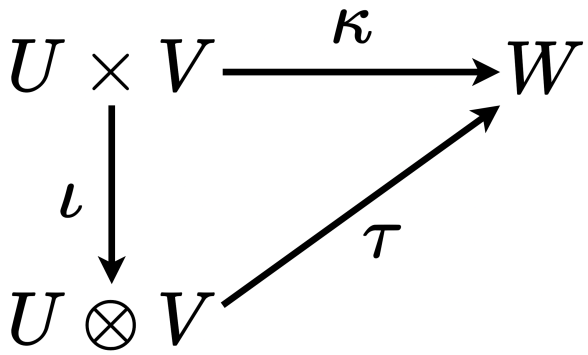


Figure 2.1: Commutative diagram expressing the universal bilinearity of the tensor product, where each bilinear map $\kappa : U \times V \to W$ factors uniquely through a linear map $\tau : U \otimes V \to W$.

terms, we must equip the tensor powers of V with suitable norms.

Let V be a Banach space. For each $n \geq 1$, let $V^{\otimes n}$ denote the completion of the $n-$fold tensor product of V with respect to a norm $\|\cdot\|_{V^{\otimes n}}$. Throughout this thesis, we assume that the family of norms $\{\|\cdot\|_{V^{\otimes n}}\}_{n=1}^{\infty}$ satisfies the following conditions for all $m, n \geq 1$, $v_1, \ldots, v_n \in V$, $v \in V^{\otimes n}$, and $w \in V^{\otimes m}$.

1. For every σ in the symmetric group over $\{1, \ldots, n\}$, let π_σ denote the permutation operator defined on simple tensors by $\pi_\sigma(v_1 \otimes \cdots \otimes v_n) = v_{\sigma(1)} \otimes \cdots \otimes v_{\sigma(n)}$ and extended to generic v by linearity and completion. Then π_σ is an isometry.

2. The norm is sub-multiplicative, $||v \otimes w||_{V^{\otimes(n+m)}} \leq ||v||_{V^{\otimes n}} \, ||w||_{V^{\otimes m}}$.

3. For any bounded linear functional f on $V^{\otimes n}$ and g on $V^{\otimes m}$, there exists a unique bounded linear functional $f \otimes g$ on $V^{\otimes(m+n)}$ such that $(f \otimes g)(v \otimes w) = f(v)g(w)$.

4. For a bounded linear functional h on a Banach space X, let

$$\|h\|_{\text{op}} = \sup_{x \neq 0} \frac{|h(x)|}{\|x\|_X} \tag{2.18}$$

be the operator norm. Then for bounded linear functionals f on $V^{\otimes n}$ and g on $V^{\otimes m}$, let $f \otimes g$ denote the linear functional defined by

$$(f \otimes g)(v \otimes w) = f(v)g(w) \tag{2.19}$$

for simple tensors and extended by linearity. Then

$$\|f \otimes g\|_{\text{op}} \leq \|f\|_{\text{op}} \|g\|_{\text{op}}. \tag{2.20}$$

Families of norms which satisfy conditions 1 and 2 are called admissible tensor norms [Lyons et al. 2007]. Requiring conditions 1, 2, and 3 is the setting of Lyons et al. [2002] and Boedihardjo et al. [2016]. Families of norms satisfying conditions 1, 2, and 4 are known as reasonable tensor algebra norms [Chang et al. 2018]. In fact, condition 4 implies condition 3 by continuity, so reasonable tensor algebra norms automatically satisfy all four properties. Families of reasonable tensor algebra norms will be sufficient for the contents of this thesis [Chang et al. 2018]. Moreover, conditions 2 and 4 ensure that for each decomposition $n = r + s$ with $r, s \geq 1$, the norm on $V^{\otimes n}$ induces a reasonable cross norm on $V^{\otimes r} \otimes V^{\otimes s}$. Therefore, the bounds in conditions 2 and 4 hold with equality [Ryan 2002; Lyons et al. 2025]. Equipping each $V^{\otimes n}$ with either the projective or injective tensor norm yields a family of reasonable tensor algebra norms [Chang et al. 2018].

Definition 2.6 (The Tensor Algebra [Lyons et al. 2007])**.** *Let* $\{V^{\otimes n}\}_{n=1}^{\infty}$ *be equipped with a family of norms* $\{\|\cdot\|_{V^{\otimes n}}\}_{n=1}^{\infty}$ *satisfying the above conditions, and define* $V^{\otimes 0} = \mathbb{R}$*. The tensor algebra is the set*

$$T((V)) = \{\mathbf{x} = (x^0, x^1, \ldots) | x^n \in V^{\otimes n}\} \tag{2.21}$$

with product $\mathbf{z} = \mathbf{x} \otimes \mathbf{y}$ *defined by*

$$z^k = (\mathbf{x} \otimes \mathbf{y})^k = \sum_{j=0}^{k} x^j \otimes y^{k-j}. \tag{2.22}$$

The tensor algebra product is associative and has unit $\mathbf{1} = (1, 0, 0, \ldots)$. As $T((V))$ is an associative algebra, it has a Lie algebra structure, with Lie bracket

$$[\mathbf{x}, \mathbf{y}] = \mathbf{x} \otimes \mathbf{y} - \mathbf{y} \otimes \mathbf{x} \tag{2.23}$$

for $\mathbf{x}, \mathbf{y} \in T((V))$ [Reutenauer 1993]. When $V = \mathbb{R}$, each tensor power $V^{\otimes n}$ is canonically identified with $\mathbb{R}$, and an element of $T((V))$ may be viewed simply as a sequence of real numbers. In this case, the tensor algebra product is exactly the one-dimensional concatenation product introduced in Lemma 2.3. Four substructures of the tensor algebra which will be relevant in this thesis are:

1. The space with only finitely many non-zero terms, denoted $T(V)$.

2. The truncated tensor algebra, denoted $T^n(V)$, whose elements are of the form $\mathbf{x} = (x^0, x^1, \ldots, x^n)$.

3. The space with $x^0 = 1$, denoted $\tilde{T}((V))$. This space is a group with inverse

$$\mathbf{x}^{-1} = 1 - (\mathbf{x} - 1) + (\mathbf{x} - 1)^{\otimes 2} - (\mathbf{x} - 1)^{\otimes 3} + \cdots . \tag{2.24}$$

4. The space of Lie series generated by V,

$$\mathfrak{L}((V)) = \{(l_0, l_1, \ldots) : l_i \in L_i\}, \tag{2.25}$$

where $L_0 = 0$, $L_1 = V$, and L_{i+1} is the span of $[v, l]$ for $v \in V$ and $l \in L_i$, which is denoted $[V, L_i]$.

2.2.3 The Tensor Algebra and $l^2(\mathbb{W}_d)$

Let V be a $d-$dimensional Banach space and $(e_1, \ldots, e_d)$ be a basis of V. A basis of $V^{\otimes n}$ is given by

$$\{e_I = e_{i_1} \otimes \cdots \otimes e_{i_n} | i_j \in \{1, \ldots, d\}\}, \tag{2.26}$$

where $I = (i_1, i_2, \ldots, i_n)$ is a multi-index. Let $\mathbb{W}_d$ denote the set of all finite sequences, or words, composed of the letters $\{1, 2, \ldots, d\}$, with the empty word $\emptyset$ included. There is a natural identification between the basis of $V^{\otimes n}$ and the words of length n via the map $\phi(e_{i_1} \otimes \cdots \otimes e_{i_n}) = i_1 \cdots i_n$. Furthermore, elements of the tensor algebra can be identified with real-valued functions $A : \mathbb{W}_d \to \mathbb{R}$.

Definition 2.7 ($l^2(\mathbb{W}_d)$ Space [Kreyszig 1978])**.** *The space $l^2(\mathbb{W}_d)$ consists of all real-valued functions $A : \mathbb{W}_d \to \mathbb{R}$ such that the sum of the squares of their values is finite:*

$$\|A\|_{l^2} = \left(\sum_{I \in \mathbb{W}_d} A_I^2 \right)^{\frac{1}{2}} < \infty. \tag{2.27}$$

This space is a Hilbert space when equipped with the inner product:

$$\langle A, B \rangle_{l^2} = \sum_{I \in \mathbb{W}_d} A_I B_I. \tag{2.28}$$

Using the above identification of $T((V))$ with real-valued functions on $\mathbb{W}_d$ and considering each as purely sets, we have the following inclusions: $T(V) \subset l^2(\mathbb{W}_d) \subset T((V))$.

2.3 The Signature

2.3.1 Definition

Armed with the tensor product, we can now extend the notion of a signature to paths taking values in a Banach space. As in the one-dimensional example of Section 2.1,

the construction is based on iterated integrals. Therefore, we begin by specifying a suitable notion of integration for the setting of this thesis, the Young integral.

Definition 2.8 (p-variation [Young 1936; Lyons et al. 2007])**.** *Let V be a Banach space and $p \geq 1$. The p-variation of a path $X : [a,b] \to V$ is defined by*

$$\|X\|_{p,[a,b]} = \left(\sup_{\mathcal{D}} \sum_{j=0}^{r-1} \|X_{t_{j+1}} - X_{t_j}\|^p \right)^{1/p}, \tag{2.29}$$

where $\mathcal{D}$ is the set of all finite partitions $(t_j)_{j=0}^r$ satisfying $a = t_0 < t_1 < \cdots < t_r = b$.

The set of paths from $[a,b]$ to V with finite $p-$variation is denoted $\mathcal{V}^p([a,b],V)$.

Definition 2.9 (Bounded Linear Maps [Kreyszig 1978])**.** *Let V and W be Banach spaces. A linear map $A : V \to W$ is bounded if there exists a constant $c > 0$ such that*

$$\|A(v)\|_W \leq c\|v\|_V \tag{2.30}$$

for all $v \in V$.

We denote the set of all bounded linear maps from V to W by $\mathbf{L}(V,W)$.

Theorem 2.10 (Young Integration [Young 1936; Lyons et al. 2007])**.** *Let V and W be Banach spaces, $p, q \geq 1$ satisfy $\frac{1}{p} + \frac{1}{q} > 1$, $X \in \mathcal{V}^p([a,b],V)$, and $Y \in \mathcal{V}^q([a,b],\mathbf{L}(V,W))$. Let $\{d^n = (t_0^n, \ldots, t_{N_n}^n)\}_{n=0}^\infty$ be a sequence of finite partitions of $[a,b]$ with $\sup_j |t_{j+1}^n - t_j^n| \to 0$ as $n \to \infty$. Then the Young integral defined by the limit*

$$\int_a^b Y_s \mathrm{d}X_s = \lim_{n\to\infty} \sum_{i=0}^{N_n-1} Y_{t_i^n}[X_{t_{i+1}^n} - X_{t_i^n}] \tag{2.31}$$

exists and does not depend on the choice of partitions $\{d^n\}_{n=0}^\infty$.

Proof. See [Young 1936] for real-valued paths and [Lyons et al. 2007] for paths taking values in Banach spaces. □

The integral defined by the limit in (2.31) also exists when both X and Y are continuous and at least one of them has finite $1-$variation. In this case, it is known as a Riemann-Stieltjes integral [Stieltjes 1894; Apostol 1974].

Definition 2.11 (The Signature [Lyons et al. 2007])**.** *Let $X \in \mathcal{V}^p([a,b],V)$ with $p < 2$ and define*

$$X_{[a,b]}^n = \underbrace{\int \cdots \int}_{\substack{u_1 < \cdots < u_n \\ u_1,\ldots,u_n \in [a,b]}} \mathrm{d}X_{u_1} \otimes \cdots \otimes \mathrm{d}X_{u_n} \in V^{\otimes n}, \tag{2.32}$$

where the integral is defined in the Young sense [Young 1936]. The signature of the path X is defined as

$$S_{[a,b]}(X) = (1, X^1_{[a,b]}, \dots, X^n_{[a,b]}, \dots) \in \tilde{T}((V)). \tag{2.33}$$

To illustrate this definition, consider the case $V = \mathbb{R}^d$ with standard basis $(e_1, \dots, e_d)$ and $X_t = (X^1_t, \dots, X^d_t)$. Then

$$X^n_{[a,b]} = \sum_{i_1,\dots,i_n=1}^{d} S^{(i_1,\dots,i_n)}_{[a,b]}\, e_{i_1} \otimes \cdots \otimes e_{i_n}, \tag{2.34}$$

where

$$S^{(i_1,\dots,i_n)}_{[a,b]} = \underbrace{\int \cdots \int}_{\substack{u_1<\cdots<u_n \\ u_1,\dots,u_n\in[a,b]}} \mathrm{d}X^{i_1}_{u_1} \cdots \mathrm{d}X^{i_n}_{u_n}. \tag{2.35}$$

In particular, the first-level terms $S^{(i)}_{[a,b]} = X^i_b - X^i_a$ are the coordinate increments of the path, while higher-order terms capture ordered interactions between coordinates.

If $X : [a, b] \to \mathbb{R}$ is one-dimensional, then for each $n \geq 1$,

$$X^n_{[a,b]} = \frac{(X_b - X_a)^n}{n!}. \tag{2.36}$$

As for the time path, the signature of a one-dimensional path depends only on the total increment $X_b - X_a$. In dimension two and higher, the order of the increments matters. For example, the piecewise linear paths $(0,0) \to (1,0) \to (1,1)$ and $(0,0) \to (0,1) \to (1,1)$ have the same first level, since they have the same increment, but different second levels. In $\mathbb{R}^2$, this difference is captured by the antisymmetric part of the second level,

$$\mathcal{A} = \frac{1}{2}\left(S^{(1,2)}_{[a,b]} - S^{(2,1)}_{[a,b]}\right), \tag{2.37}$$

which is the signed area enclosed by the path and its chord. So, beyond one dimension, the signature records not only where a path starts and ends, but also information about the route it traverses, as illustrated in Figure 2.2.

The condition $\frac{1}{p} + \frac{1}{q} > 1$ in Theorem 2.10 restricts our definition of the signature to paths with finite $p-$variation for p less than 2. Although this is sufficient for the paths considered in this thesis, many important examples, such as Brownian motion, have infinite $p-$variation for all $p < 2$. Rough path theory provides a generalisation of the construction presented here to include paths that have finite p-variation for some $p \geq 2$, but for no $p < 2$ [Lyons 1998]. An accessible introduction to rough path theory can be found in [Lyons et al. 2007].

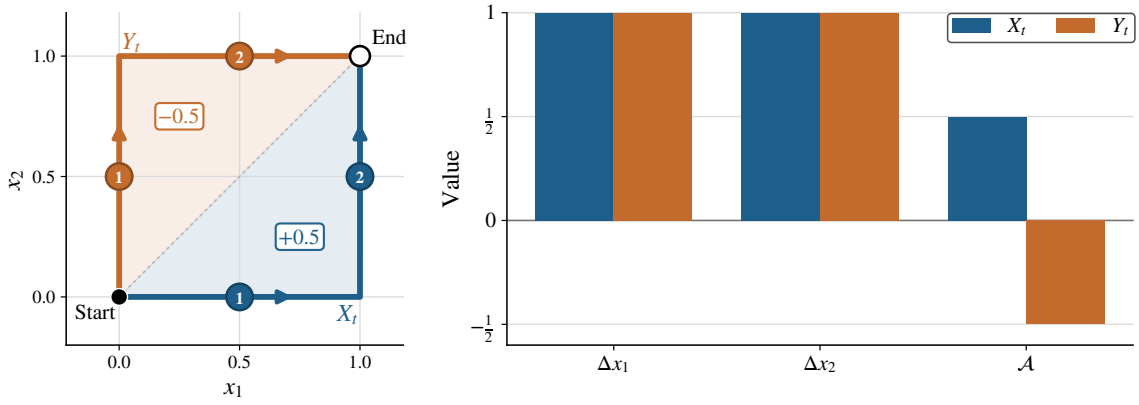


Figure 2.2: Two piecewise linear paths from Start to End with the same net displacement but different ordering: the blue path X_t moves first in the x_1 direction and then in x_2, while the orange path Y_t moves first in the x_2 direction and then in x_1, as indicated by the numbered circles. The shaded triangles show the corresponding signed areas and the bar chart compares the common first-level terms $S^{(1)}$ and $S^{(2)}$ with the signed area $\mathcal{A} = \frac{1}{2}\big(S^{(1,2)} - S^{(2,1)}\big)$ for the two paths.

2.3.2 Properties

In his seminal work on the signature of finite-dimensional bounded-variation paths, Chen [1954] introduced a number of important properties, including invariance to reparametrisation and translation. Lyons et al. [2007] provide an exposition of these properties for paths in $\mathcal{V}^p([a,b],V)$ with $p < 2$.

Lemma 2.12 (Invariance to Reparametrisation [Chen 1954; Lyons et al. 2007]). *Let $X \in \mathcal{V}^p([a,b],V)$ with $p < 2$ and $\alpha : [a,b] \to [a,b]$ be a continuous, strictly increasing function satisfying $\alpha(a) = a$ and $\alpha(b) = b$. Then*

$$S_{[a,b]}(X) = S_{[a,b]}(X \circ \alpha). \tag{2.38}$$

Lemma 2.13 (Invariance to Translation [Chen 1954; Lyons et al. 2007]). *Let $X \in \mathcal{V}^p([a,b],V)$ with $p < 2$, and $v \in V$. Define $Y : [a,b] \to V$ by $Y_t = X_t + v$ for all $t \in [a,b]$. Then*

$$S_{[a,b]}(X) = S_{[a,b]}(Y). \tag{2.39}$$

For paths that do not self-intersect, in the sense that $s \neq t$ implies $X_s \neq X_t$, translation and reparametrisation are the only transformations under which the signature is invariant. This makes the signature a natural representation when the object of interest is the shape of a path, since it treats the infinitely many paths related by translation or reparametrisation as equivalent. However, when a path does self-intersect,

further invariances appear. A basic example is that the signature is unchanged by a portion of the path that traces out an excursion and then retraces it exactly. In the bounded-variation setting, Hambly et al. [2010] showed that the full class of such invariances is captured by tree-like equivalence. Boedihardjo et al. [2016] subsequently extended this result to the $p < 2$ setting. For our purposes, it is enough to note that translation invariance is removed by working with paths that share a common initial point, while reparametrisation invariance and tree-like equivalence are removed by augmenting the path with time. In particular, the added time channel is strictly increasing, so time-augmented paths cannot self-intersect.

Definition 2.14. *Let $\mathcal{V}_0^p([a,b],V)$ denote the set of paths $X \in \mathcal{V}^p([a,b],V)$ where $X_a = 0$.*

Definition 2.15 (Time augmentation)**.** *For $X \in \mathcal{V}^p([a,b],V)$, define the time-augmented path $\hat{X} : [a,b] \to \mathbb{R} \times V$ by*

$$\hat{X}_t = (t, X_t). \tag{2.40}$$

Another fundamental property established by Chen [1954] is that the signature is multiplicative under concatenation of intervals.

Theorem 2.16 (Chen's Identity [Chen 1954; Lyons et al. 2007])**.** *Let $X \in \mathcal{V}^p([a,b],V)$ with $p < 2$. Then for all $t \in [a,b]$,*

$$S_{[a,b]}(X) = S_{[a,t]}(X) \otimes S_{[t,b]}(X). \tag{2.41}$$

Theorem 2.16 is the general analogue of the concatenation property in Lemma 2.3 for the signature of the time path. It shows that the signature over a long interval may be decomposed into signatures over smaller subintervals and then recombined via the tensor product. This makes the signature amenable to memory-efficient recurrent computation, since the current signature encodes all necessary information from previous intervals, and to parallel computation via an associative scan [Blelloch 1993].

As shown in (2.36), the terms of the signature of a one-dimensional path decay factorially. The following theorem shows that there is an analogous decay estimate for general paths.

Definition 2.17. *Let Γ denote the gamma function, defined for $x > 0$ by*

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} \, \mathrm{d}t. \tag{2.42}$$

The gamma function extends the factorial function to the positive real line, in the sense that $\Gamma(1+m) = m!$ for $m \in \mathbb{N}_0$.

Theorem 2.18 (Factorial decay [Lyons 1994, Theorem 2.2.1]). *Let $X \in \mathcal{V}^p([a,b],V)$ with $p < 2$. Then there exists a finite constant $C_p > 1$, depending only on p, such that for every $[s,t] \subset [a,b]$ and every $n \geq 1$,*

$$\left\| X^n_{[s,t]} \right\| \leq C_p \frac{\|X\|^n_{p\text{-}var;[s,t]}}{\Gamma\left(1+\frac{n}{p}\right)}. \tag{2.43}$$

Theorem 2.18 shows that the signature decays rapidly with tensor level, and hence defines an element of the completed tensor algebra. Letting $S(\mathcal{V}^p([a,b],V))$ denote the set of signatures of paths in $\mathcal{V}^p([a,b],V)$ and using the identification introduced in Section 2.2.3, we have

$$S(\mathcal{V}^p([a,b],V)) \subset \ell^2(\mathbb{W}_d), \tag{2.44}$$

when V is $d-$dimensional. Equation (2.36) also shows that each term of the signature of a one-dimensional path is a monomial in the first term. For general paths, an analogous algebraic structure persists in a more subtle form. As first shown by Ree [1958], products of linear functionals applied to lower-order signature terms can be rewritten as a single linear functional applied to a higher-order term. This result is known as the shuffle product identity.

Definition 2.19. *Let $\mathrm{Sh}(\alpha,\beta)$ denote the set of permutations $\sigma \in S_{\alpha+\beta}$ such that $\sigma(1) < \cdots < \sigma(\alpha)$ and $\sigma(\alpha+1) < \cdots < \sigma(\alpha+\beta)$.*

Definition 2.20 (Shuffle of functionals). *Let V be a Banach space and let $\alpha, \beta \in \mathbb{N}_0$. Let f be a bounded linear functional on $V^{\otimes\alpha}$ and g on $V^{\otimes\beta}$. Their shuffle $f \sqcup\!\sqcup g$ is the bounded linear functional on $V^{\otimes(\alpha+\beta)}$ defined by*

$$(f \sqcup\!\sqcup g)(w) = \sum_{\sigma \in \mathrm{Sh}(\alpha,\beta)} (f \otimes g)\big(\pi_\sigma w\big), \qquad w \in V^{\otimes(\alpha+\beta)}. \tag{2.45}$$

Theorem 2.21 (Shuffle Product Identity [Ree 1958; Lyons et al. 2007]). *Let V be a Banach space and $X \in \mathcal{V}^p([a,b],V)$ with $p < 2$. Then for all $\alpha, \beta \in \mathbb{N}_0$ and for every pair of bounded linear functionals f on $V^{\otimes\alpha}$ and g on $V^{\otimes\beta}$,*

$$f\left(X^\alpha_{[a,b]}\right) g\left(X^\beta_{[a,b]}\right) = (f \sqcup\!\sqcup g)\left(X^{\alpha+\beta}_{[a,b]}\right). \tag{2.46}$$

The shuffle product identity is the key algebraic result underlying the expressivity of linear functionals of the signature, since it shows that polynomial functions of the

signature can be re-expressed as linear functions of the signature. This is the direct analogue of linear maps on the signature of the time path generating all polynomials in t. Moreover, although these functionals are linear on the tensor algebra, when composed with the signature they define highly non-linear functions of the underlying path. Furthermore, the shuffle product identity shows that these linear functionals form an algebra, placing them in the setting of the Stone–Weierstrass theorem.

Theorem 2.22 (Stone–Weierstrass [Stone 1948])**.** *Let (X, d) be a compact metric space and A be a subalgebra of $C(X, \mathbb{R})$ which contains the constant functions. Then A is dense in $C(X, \mathbb{R})$ if and only if it separates the points.*

To apply Theorem 2.22, we first introduce the family of linear functionals on the tensor algebra obtained by lifting bounded linear functionals from each tensor level.

Definition 2.23. *For $\alpha \in \mathbb{N}_0$ and a bounded linear functional f on $V^{\otimes\alpha}$, define $\ell^{(\alpha,f)} : T((V)) \to \mathbb{R}$ by*

$$\ell^{(\alpha,f)}(\mathbf{x}) = f(x^\alpha), \tag{2.47}$$

where $\mathbf{x} = (x^0, x^1, \ldots) \in T((V))$.

Given an interval $[s, t] \subset [a, b]$ and a path $X \in \mathcal{V}^p([a, b], V)$ with $p < 2$, we view the corresponding function on paths as the pullback of an $\ell^{(\alpha,f)}$.

Definition 2.24. *Let $[s, t] \subset [a, b]$, $\alpha \in \mathbb{N}_0$, and f be a bounded linear functional on $V^{\otimes\alpha}$. Define $\phi^{(\alpha,f)}_{[s,t]} = \ell^{(\alpha,f)} \circ S_{[s,t]} : \mathcal{V}^p([a, b], V) \to \mathbb{R}$ by*

$$\phi^{(\alpha,f)}_{[s,t]}(X) = f\big(X^\alpha_{[s,t]}\big). \tag{2.48}$$

We now verify that these linear functionals satisfy the hypotheses of the Stone–Weierstrass theorem on compact subsets of signature space.

Lemma 2.25. *Let $\mathcal{K}_S \subset S(\mathcal{V}^p([a, b], V))$ be compact and $\mathcal{F}_\alpha$ be the space of bounded linear functionals on $V^{\otimes\alpha}$. The family $\{\ell^{(\alpha,f)}|_{\mathcal{K}_S} : \alpha \in \mathbb{N}_0,\ f \in \mathcal{F}_\alpha\}$ separates the points of $\mathcal{K}_S$.*

Proof. If $\mathbf{x}, \mathbf{y} \in \mathcal{K}_S$ are distinct, then there exists a minimal $n \geq 0$ with $x^n \neq y^n$ in $V^{\otimes n}$. By the Hahn–Banach Theorem, there exists $f \in \mathcal{F}_n$ with $f(x^n) \neq f(y^n)$, so $\ell^{(n,f)}(\mathbf{x}) \neq \ell^{(n,f)}(\mathbf{y})$ [Rudin 1991, Theorem 3.3]. □

Lemma 2.26. *Let $\mathcal{K}_S \subset S(\mathcal{V}^p([a, b], V))$ be compact. The set of finite linear combinations of products of the restrictions $\ell^{(\alpha,f)}|_{\mathcal{K}_S}$ is a subalgebra of $C(\mathcal{K}_S, \mathbb{R})$ which contains the constants.*

Proof. Constants are obtained from $\alpha = 0$, since $V^{\otimes 0} = \mathbb{R}$ and $X^0_{[a,b]} = 1$. Closure under pointwise multiplication on $\mathcal{K}_S$ follows from the shuffle product: if $\mathbf{s} = S_{[a,b]}(X) \in \mathcal{K}_S$, then by Theorem 2.21

$$\ell^{(\alpha,f)}(\mathbf{s})\,\ell^{(\beta,g)}(\mathbf{s}) = f\big(X^{\alpha}_{[a,b]}\big)\,g\big(X^{\beta}_{[a,b]}\big) = (f \sqcup\!\sqcup g)\big(X^{\alpha+\beta}_{[a,b]}\big) = \ell^{(\alpha+\beta,\, f\sqcup\!\sqcup g)}(\mathbf{s}). \tag{2.49}$$

□

Theorem 2.27 (Stone–Weierstrass on signature space [Levin et al. 2016])**.** *Let $\mathcal{K}_S \subset S(\mathcal{V}^p([a,b],V))$ be compact with $p < 2$. Then the algebra generated by $\{\ell^{(\alpha,f)}|_{\mathcal{K}_S}\}$ is dense in $C(\mathcal{K}_S, \mathbb{R})$.*

Proof. By Lemma 2.26, we have a subalgebra of $C(\mathcal{K}_S, \mathbb{R})$ which contains the constants. By Lemma 2.25, it separates points. Therefore, we can apply Stone–Weierstrass, Theorem 2.22. □

Theorem 2.27 shows that, on any compact subset of signature space, functions of the signature can be approximated uniformly by finite linear combinations of the functionals $\ell^{(\alpha,f)}$. In other words, linear functionals of the signature are rich enough to approximate arbitrary continuous functions on compact subsets of signature space. However, our primary interest is in functions defined on paths rather than on their signatures. Therefore, we now transfer this approximation result from signature space to path space by pulling these functionals back along the signature map.

Let $\mathcal{K} \subset \mathcal{V}^p([a,b],V)$ be compact in a standard path topology, such as the p-variation topology, and let $\mathcal{K}_S = S_{[a,b]}(\mathcal{K})$. Since $S_{[a,b]}$ is continuous for $p < 2$, the set $\mathcal{K}_S$ is compact.

Corollary 2.28 (Universality on path space)**.** *Let $\mathcal{K} \subset \mathcal{V}^p([a,b],V)$ be compact with $p < 2$. Define the equivalence relation*

$$X \sim Y \iff S_{[a,b]}(X) = S_{[a,b]}(Y). \tag{2.50}$$

Then:

1. *The algebra generated by $\{\phi^{(\alpha,f)}_{[a,b]}|_{\mathcal{K}}\}$ is dense in the space*

$$\{F \in C(\mathcal{K}, \mathbb{R}) \mid X \sim Y \implies F(X) = F(Y)\}. \tag{2.51}$$

2. *If $S_{[a,b]}$ is injective on $\mathcal{K}$, then the same algebra is dense in $C(\mathcal{K}, \mathbb{R})$.*

Proof. (1) The induced map $\bar{S} : \mathcal{K}/\sim \to \mathcal{K}_S$ is a homeomorphism. By Theorem 2.27, polynomials in $\{\ell^{(\alpha,f)}\}$ are dense on $\mathcal{K}_S$; pulling back along $\bar{S}$ yields density of polynomials in $\{\phi_{[a,b]}^{(\alpha,f)}\}$ on $\mathcal{K}/\sim$.
(2) If $S_{[a,b]}$ is injective on $\mathcal{K}$, then $\mathcal{K} \cong \mathcal{K}_S$ via $S_{[a,b]}$, and the conclusion follows directly from Theorem 2.27. □

Corollary 2.28 shows that, on a general compact set of paths, linear functionals of the signature approximate exactly those continuous functionals that depend only on the signature. To strengthen this to universality for all continuous functionals on path space, it suffices to work in a setting where the signature map is injective. By the discussion earlier in this section, this can be achieved by restricting to paths with a common initial point, which removes translation invariance, and by augmenting with time, which removes reparametrisation and tree-like equivalence.

Lemma 2.29 (Injectivity via augmentation)**.** *Let $\mathcal{K} \subset \mathcal{V}_0^p([a,b],V)$ be compact. Then $S_{[a,b]}(\hat{X})$ is injective on $\mathcal{K}$, and hence the algebra generated by $\{\phi_{[a,b]}^{(\alpha,f)}(\hat{X})\}$ is dense in $C(\mathcal{K},\mathbb{R})$.*

Proof. Since each $X \in \mathcal{K}$ begins at the same point and the time channel of $\hat{X}$ is strictly monotone, no two distinct time-augmented paths can be tree-like equivalent [Levin et al. 2016]. Therefore $S_{[a,b]}(\hat{X})$ determines X uniquely [Boedihardjo et al. 2016; Hambly et al. 2010] and Corollary 2.28 can be applied. □

Remark 2.30. *The definition of a compact set $\mathcal{K} \subset \mathcal{V}^p([a,b],V)$ depends on the chosen path topology [Bonnier et al. 2019]. In most data-driven applications, it suffices to take $\mathcal{K}$ to be a finite, and hence compact, set.*

Lemma 2.29 is a desirable property when using the signature of a path as a feature set for linear regression [Levin et al. 2016]. However, this linearisation of non-linear functions leads to algebraic redundancy in the signature, as shown in Theorem 2.21. For one-dimensional paths, every higher-order term is a monomial in the first-level term. For general paths, not all higher-order terms are determined by lower-order ones, but the shuffle product identity shows that certain linear functionals of the higher-order terms are. Thus part of the information appearing at higher tensor levels is merely a linear encoding of polynomial functions of the lower-order terms. The transformation which removes these algebraic redundancies is the logarithm.

2.3.3 The Log-Signature

We begin with the formal definitions of the logarithm and exponential on the tensor algebra.

Definition 2.31. *For $\mathbf{x} \in \tilde{T}((V))$, the logarithm is defined by*

$$\log(\mathbf{x}) = \log(1 + \mathbf{t}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} \mathbf{t}^{\otimes n}, \tag{2.52}$$

where $\mathbf{t} = (0, x^1, x^2, \ldots)$.

Since $\mathbf{x} \in \tilde{T}((V))$, all non-zero elements of $\mathbf{t}^{\otimes n}$ have degree at least n. Therefore, the logarithm converges degree-wise.

Definition 2.32. *For $\mathbf{x} \in T((V))$ with $x^0 = 0$, the exponential is defined by*

$$\exp(\mathbf{x}) = \sum_{n=0}^{\infty} \frac{\mathbf{x}^{\otimes n}}{n!}. \tag{2.53}$$

Similarly to the logarithm, the exponential converges degree-wise. Furthermore, for $\mathbf{x} \in \tilde{T}((V))$,

$$\exp(\log(\mathbf{x})) = \mathbf{x} \tag{2.54}$$

and for $\mathbf{x} \in T((V))$ with $x^0 = 0$,

$$\log(\exp(\mathbf{x})) = \mathbf{x}. \tag{2.55}$$

The effect of the logarithm is already visible in one dimension. If $V = \mathbb{R}$, then by (2.36),

$$S_{[a,b]}(X) = \left(1, X_b - X_a, \frac{(X_b - X_a)^2}{2!}, \frac{(X_b - X_a)^3}{3!}, \ldots\right) = \exp(X_b - X_a), \tag{2.56}$$

and hence

$$\log(S_{[a,b]}(X)) = (0, X_b - X_a, 0, 0, \ldots). \tag{2.57}$$

In one dimension the signature records all monomials in the increment as separate tensor levels, whereas the log-signature retains only the increment itself. This makes the log-signature a more economical representation, but it also removes the linearisation property of the signature, since linear functionals of the signature can recover polynomials in $X_b - X_a$, whereas linear functionals of the log-signature cannot.

For general paths, the same idea first appears at second level. Substituting $S_{[a,b]}(X) = (1, X^1_{[a,b]}, X^2_{[a,b]}, \ldots)$ into the logarithm gives

$$\log(S_{[a,b]}(X)) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} \left(0, X^1_{[a,b]}, X^2_{[a,b]}, \ldots\right)^{\otimes n}, \tag{2.58}$$

so at the first two levels,

$$\log(S_{[a,b]}(X)) = \left(0, X^1_{[a,b]}, X^2_{[a,b]} - \frac{1}{2} X^1_{[a,b]} \otimes X^1_{[a,b]}, \ldots\right). \tag{2.59}$$

The shuffle product identity shows exactly why this removes redundancy. If f and g are bounded linear functionals on V, then Theorem 2.21 gives

$$f(X^1_{[a,b]}) g(X^1_{[a,b]}) = (f \sqcup\!\sqcup g)(X^2_{[a,b]}) = (f \otimes g + g \otimes f)(X^2_{[a,b]}). \tag{2.60}$$

On the other hand,

$$(f \otimes g + g \otimes f)\left(\frac{1}{2} X^1_{[a,b]} \otimes X^1_{[a,b]}\right) = f(X^1_{[a,b]}) g(X^1_{[a,b]}). \tag{2.61}$$

Therefore

$$(f \otimes g + g \otimes f)\left(X^2_{[a,b]} - \frac{1}{2} X^1_{[a,b]} \otimes X^1_{[a,b]}\right) = 0 \tag{2.62}$$

for every pair of bounded linear functionals f and g on V. So every second-level term detected by a shuffle product of first-level functionals vanishes after taking the logarithm. In other words, the part of the second level that was already determined by the first level has been removed.

This is particularly transparent when $V = \mathbb{R}^2$. Writing

$$X^2_{[a,b]} = S^{(1,1)}_{[a,b]} e_1 \otimes e_1 + S^{(1,2)}_{[a,b]} e_1 \otimes e_2 + S^{(2,1)}_{[a,b]} e_2 \otimes e_1 + S^{(2,2)}_{[a,b]} e_2 \otimes e_2, \tag{2.63}$$

the shuffle product identity gives

$$S^{(1,1)}_{[a,b]} = \frac{1}{2}\left(S^{(1)}_{[a,b]}\right)^2, \tag{2.64}$$

$$S^{(2,2)}_{[a,b]} = \frac{1}{2}\left(S^{(2)}_{[a,b]}\right)^2, \tag{2.65}$$

and

$$S^{(1,2)}_{[a,b]} + S^{(2,1)}_{[a,b]} = S^{(1)}_{[a,b]} S^{(2)}_{[a,b]}. \tag{2.66}$$

Substituting these identities into (2.59) shows that the coefficients of $e_1 \otimes e_1$ and $e_2 \otimes e_2$ become

$$S^{(1,1)}_{[a,b]} - \frac{1}{2}\left(S^{(1)}_{[a,b]}\right)^2 = 0 \tag{2.67}$$

and

$$S^{(2,2)}_{[a,b]} - \frac{1}{2}\big(S^{(2)}_{[a,b]}\big)^2 = 0. \tag{2.68}$$

Moreover, the sum of the coefficients of $e_1 \otimes e_2$ and $e_2 \otimes e_1$ becomes

$$\left(S^{(1,2)}_{[a,b]} - \frac{1}{2}S^{(1)}_{[a,b]}S^{(2)}_{[a,b]}\right) + \left(S^{(2,1)}_{[a,b]} - \frac{1}{2}S^{(2)}_{[a,b]}S^{(1)}_{[a,b]}\right) = 0. \tag{2.69}$$

So all the second-level terms determined by the first level are cancelled by the logarithm. The only part that remains is

$$\frac{1}{2}\left(S^{(1,2)}_{[a,b]} - S^{(2,1)}_{[a,b]}\right)(e_1 \otimes e_2 - e_2 \otimes e_1), \tag{2.70}$$

which is exactly the signed area term. Thus, at second level, the log-signature removes the algebraic redundancy in the signature and retains only the genuinely new information.

Chen [1957] formalised this for all levels of the signature by showing that signatures are group-like, and hence that their logarithms lie in the free Lie algebra. As noted in Section 2.2.2, $\tilde{T}((V))$ is a group. We define

$$\mathcal{G} = \left\{\mathbf{x} \in \tilde{T}((V)) \mid \log(\mathbf{x}) \in \mathfrak{L}((V))\right\}. \tag{2.71}$$

This is a subgroup of $\tilde{T}((V))$, and its elements are called group-like [Reutenauer 1993]. Chen's result shows that, for bounded-variation paths,

$$S_{[a,b]}(X) \in \mathcal{G}, \tag{2.72}$$

or equivalently,

$$\log(S_{[a,b]}(X)) \in \mathfrak{L}((V)). \tag{2.73}$$

Thus the signature may be represented equivalently by its logarithm in $\mathfrak{L}((V))$, a strict subspace of $T((V))$. This shows that no information is lost by passing to the log-signature. The trade-off is that the linearisation property is no longer present, because the higher-order tensor terms which encode polynomial functions of the lower-order terms have been removed.

If a path is linear on an interval with increment $v \in V$, then its log-signature on that interval is simply $(0, v, 0, 0, \ldots)$, and so its signature is

$$\exp(0, v, 0, 0, \ldots) = \left(1, v, \frac{v^{\otimes 2}}{2!}, \frac{v^{\otimes 3}}{3!}, \ldots\right). \tag{2.74}$$

Hence, if a path is represented by a piecewise linear interpolation with increments $v_1, \ldots, v_m$, then the signature on each subinterval is obtained from the corresponding exponential, and the full signature is computed as

$$S_{[a,b]}(X) = \exp(0, v_1, 0, 0, \ldots) \otimes \cdots \otimes \exp(0, v_m, 0, 0, \ldots). \tag{2.75}$$

Thus signatures of discrete data can be computed efficiently by working interval by interval and then combining the results with Chen's identity.

As the signature and log-signature are infinite-dimensional, it can be useful to consider the truncated signature

$$S^N_{[a,b]}(X) = (1, X^1_{[a,b]}, \ldots, X^N_{[a,b]}) \in \tilde{T}^N(V), \tag{2.76}$$

or truncated log-signature,

$$\log(S^N_{[a,b]}(X)) \in \mathfrak{L}^N(V). \tag{2.77}$$

As discussed in Section 1.1.2, the truncated signature has natural applications in machine learning, as it is a vector embedding of a multivariate path which captures interactions between the channels of the path. Reviews of the machine learning applications can be found in [Chevyrev et al. 2016a], [Cass et al. 2024b], and [McLeod et al. 2025].

So far, the signature has been treated as a representation of a path built from iterated integrals. However, it also admits a dynamical interpretation, since it arises as the solution of a linear CDE driven by the path. More generally, CDEs provide a framework for describing how a state evolves in response to a driving signal, with the signature appearing naturally when one studies their flow over an interval. The continuous-time machine learning models developed in this thesis adopt this perspective by treating time series data as the driving signal and a CDE's evolving state as the hidden state.

2.4 Controlled Differential Equations

This section introduces CDEs, states the basic well-posedness results, and then specialises to the linear case, where the solution can be written explicitly in terms of the signature. It then returns to the log-signature and shows how it leads to the Log-ODE method, an efficient and accurate method for approximating the solution of a CDE.

2.4.1 Definition

Let V and W be Banach spaces, $X : [a, b] \to V$ and $Y : [a, b] \to W$ be continuous paths, and $f : W \to \mathbf{L}(V, W)$ be a continuous function. The vector field f can equivalently be viewed as a linear map from $v \in V$ to a vector field on W denoted $f(\cdot)v$. This second formulation will prove useful when defining the Log-ODE method. Assume that X, Y, and f are sufficiently regular for the integral

$$\int_a^t f(Y_s)\mathrm{d}X_s \tag{2.78}$$

to be defined for all $t \in [a, b]$ in the Young sense [Young 1936]. The path Y is said to obey a CDE if

$$Y_t = Y_a + \int_a^t f(Y_s)\mathrm{d}X_s, \tag{2.79}$$

for $t \in [a, b]$, where $Y_a \in W$ is the initial condition and X is the control [Lyons et al. 2007].

Just as the signature of time is the solution to an infinite set of differential equations (2.3), the signature of a path $X \in \mathcal{V}^p([a, b], V)$ for $p < 2$ is the solution to a tensor CDE,

$$\mathrm{d}S_{[a,s]}(X) = S_{[a,s]}(X) \otimes \mathrm{d}X_s, \tag{2.80}$$

where $S_{[a,s]}(X) : [a, b] \to T((V))$ and $S_{[a,a]}(X) = (1, 0, 0, \ldots) \in T((V))$ [Salvi et al. 2021]. When V is finite-dimensional with dimension d, we may identify $T((V))$ with the space of real-valued functions on $\mathbb{W}_d$, as described in Section 2.2.3, and hence represent elements of $T((V))$ as vectors in $\mathbb{R}^{\mathbb{W}_d}$. In this representation,

$$\mathrm{d}S_{[a,s]}(X) = AS_{[a,s]}(X)\mathrm{d}X_s, \tag{2.81}$$

where $A \in \mathbf{L}(\mathbb{R}^{\mathbb{W}_d}, \mathbf{L}(\mathbb{R}^d, \mathbb{R}^{\mathbb{W}_d}))$. This can be equivalently represented by a set of d matrices $A^i \in \mathbb{R}^{\mathbb{W}_d \times \mathbb{W}_d}$ satisfying

$$AS_{[a,s]}(X)\mathrm{d}X_s = \sum_{i=1}^{d} A^i S_{[a,s]}(X)\mathrm{d}X_s^i. \tag{2.82}$$

Ordering the elements of $\mathbb{W}_d$ first by size and then alphabetically, the $(j, k)^{\text{th}}$ element of A^i satisfies

$$A^i_{jk} = \begin{cases} 1, & j = 1 + i + d(k - 1), \\ 0, & \text{otherwise.} \end{cases} \tag{2.83}$$

This representation is the linear algebraic analogue of the recursive structure of the iterated integrals defining the signature. As seen in (2.35), the differential of the signature term corresponding to the word $(i_1, \ldots, i_n)$ is determined by the lower-order term indexed by $(i_1, \ldots, i_{n-1})$ together with the increment in the final channel. The matrices A^i encode exactly this operation on the word basis by appending the letter i to a word, thereby mapping each basis element to the corresponding basis element at the next tensor level.

2.4.2 Existence and Uniqueness

Whether a CDE is well posed depends on the regularity of both the control path X and the vector field f. Rougher driving signals require stronger regularity of the vector field in order for the dynamics to remain well defined. We measure the regularity of a path by the smallest $p \geq 1$ for which its p-variation is finite, and the regularity of a vector field by the largest $\gamma > 0$ such that it is $\mathrm{Lip}(\gamma)$. We defer the formal definition of $\mathrm{Lip}(\gamma)$ functions to Chapter 3, which is devoted to them.

Theorem 2.33 (CDE existence [Lyons 1994; Lyons et al. 2007])**.** *Let $1 \leq p < 2$ and $p - 1 < \gamma$. If W is finite-dimensional, X has finite $p-$variation, and f is Lip(γ), then (2.79) admits a solution for every $Y_a \in W$.*

Theorem 2.34 (CDE uniqueness [Lyons 1994; Lyons et al. 2007])**.** *Let $1 \leq p < 2$ and $p < \gamma$. If X has finite $p-$variation and f is Lip(γ), then (2.79) admits a unique solution for every $Y_a \in W$.*

Theorems 2.33 and 2.34 extend the classical existence and uniqueness theory for ordinary differential equations to controls with unbounded variation but finite p-variation for $p < 2$. They show that as the driving path becomes less regular, stronger assumptions on f are needed to ensure that the CDE remains well posed.

2.4.3 Solutions of Linear CDEs

Linear CDEs play a central role in this thesis for two reasons. First, when the driving path is piecewise linear, the flow on each linear segment can be found explicitly and independently. This will later underpin the scalable, parallel-in-time computational methods developed for Linear NCDEs. Second, their solutions can be expressed in terms of the signature of the driving path, allowing their expressivity to be understood through this explicit solution and the expressivity of the signature.

For a driving path $X : [a, b] \to V$ and output path $Y : [a, b] \to W$, a linear CDE takes the form

$$Y_t = Y_a + \int_a^t AY_s \mathrm{d}X_s, \tag{2.84}$$

where $A \in \mathbf{L}(W, \mathbf{L}(V, W))$. Alternatively,

$$Y_t = Y_a + \int_a^t A(\mathrm{d}X_s)Y_s, \tag{2.85}$$

with $A \in \mathbf{L}(V, \mathbf{L}(W, W))$. As shown in (2.82), the signature of a path is an example of a linear CDE. However, the existence and uniqueness Theorems 2.33 and 2.34 do not apply, as the value of the vector fields is unbounded, and hence they are not $\mathrm{Lip}(\gamma)$. Corresponding existence and uniqueness results for linear vector fields are given as Theorems 3.7 and 3.8 in [Friz et al. 2010]. Here, we present the explicit form of the unique solution when the norm on each $V^{\otimes n}$ is the projective tensor norm.

Definition 2.35 (Projective tensor norm [Boedihardjo et al. 2016])**.** *Let V be a Banach space. The projective tensor norm on $V^{\otimes n}$ is defined by*

$$\|v\| = \inf\left\{\sum_k \|v_k^1\| \cdots \|v_k^n\| : v = \sum_k v_k^1 \otimes \cdots \otimes v_k^n\right\} \tag{2.86}$$

for $v \in V^{\otimes n}$ and $v_k^i \in V$.

Equipping each $V^{\otimes n}$ with the projective tensor norm yields a family of tensor norms satisfying the assumptions outlined in Section 2.2.2.

Definition 2.36 (Operator tensor powers)**.** *Let V, W be Banach spaces. For $n \geq 1$ and $A \in \mathbf{L}(V, \mathbf{L}(W, W))$, let $A^{\otimes n} \in \mathbf{L}\big(V^{\otimes n}, \mathbf{L}(W, W)\big)$ be the operator defined on simple tensors by*

$$A^{\otimes n}(v^1 \otimes \cdots \otimes v^n) = A(v^n) \cdots A(v^1), \tag{2.87}$$

and extended by linearity and continuity to $V^{\otimes n}$. Set $A^{\otimes 0} = I_W$.

Since $V^{\otimes n}$ is equipped with the projective tensor norm, $\|A^{\otimes n}(v)\| \leq \|A\|^n \|v\|$ for $v \in V^{\otimes n}$.

Theorem 2.37 (Linear CDE solution)**.** *Let V, W be Banach spaces, $A \in \mathbf{L}(V, \mathbf{L}(W, W))$, $X \in \mathcal{V}^p([a, b], V)$ with $p < 2$, and $V^{\otimes n}$ be equipped with the projective tensor norm. Define the evolution operator*

$$\Phi_{t,s} = \sum_{n=0}^{\infty} A^{\otimes n}\big(X^n_{[s,t]}\big) \in \mathbf{L}(W, W). \tag{2.88}$$

Then the series (2.88) *converges absolutely in operator norm and the unique solution to* (2.84) *is*

$$Y_t = \Phi_{t,a}\, Y_a. \tag{2.89}$$

Proof. This proof has four key steps.

1. Apply Picard iteration and show that the iterates correspond to truncated versions of the sum in (2.88).
2. Use the factorial decay of the signature to show that the Picard iterates converge.
3. Upgrade the convergence of the Picard iterates from uniform to p-variation and pass the limit through the Young integral to obtain $Y_t = \Phi_{t,a}\, Y_a$.
4. Demonstrate uniqueness by reapplying the Picard iteration with initial datum zero.

1) Picard iteration Define $Y_t^{(0)} = Y_a$ and, recursively,

$$Y_t^{(m+1)} = Y_a + \int_a^t A(\mathrm{d}X_s) Y_s^{(m)}, \tag{2.90}$$

for $m \geq 0$. We claim that for every $m \geq 0$ and all $t \in [a, b]$,

$$Y_t^{(m)} \;=\; \sum_{n=0}^{m} A^{\otimes n}\big(X_{[a,t]}^n\big)\, Y_a. \tag{2.91}$$

This is proved by induction on m. For $m = 0$, the identity is $Y_t^{(0)} = A^{\otimes 0}(X_{[a,t]}^0)Y_a = Y_a$. Assume (2.91) holds for some $m \geq 0$. Then,

$$\begin{aligned} Y_t^{(m+1)} &= Y_a + \int_a^t A(\mathrm{d}X_s) Y_s^{(m)}, \\ &= Y_a + \sum_{n=0}^{m} \int_a^t A(\mathrm{d}X_s) A^{\otimes n}(X_{[a,s]}^n) Y_a, \end{aligned} \tag{2.92}$$

and,

$$\int_a^t A(\mathrm{d}X_s) A^{\otimes n}\big(X_{[a,s]}^n\big) = A^{\otimes n+1}\left(\int_a^t X_{[a,s]}^n \otimes \mathrm{d}X_s\right) = A^{\otimes (n+1)}\big(X_{[a,t]}^{n+1}\big). \tag{2.93}$$

Hence,

$$Y_t^{(m+1)} \;=\; Y_a + \sum_{n=0}^{m} A^{\otimes (n+1)}\big(X_{[a,t]}^{n+1}\big) Y_a \;=\; \sum_{n=0}^{m+1} A^{\otimes n}\big(X_{[a,t]}^n\big) Y_a, \tag{2.94}$$

which completes the induction and proves (2.91).

2) Convergence of the Picard iterates By Theorem 2.18, there exists a finite constant $C_p > 1$ depending only on p such that for $n \geq 1$

$$\left\|X^n_{[s,t]}\right\| \leq C_p \frac{\|X\|^n_{p\text{-var};[s,t]}}{\Gamma(1+\frac{n}{p})}. \tag{2.95}$$

Since $\|A^{\otimes n}(v)\| \leq \|A\|^n \|v\|$ for $v \in V^{\otimes n}$,

$$\left\|A^{\otimes n}\big(X^n_{[a,t]}\big)\right\| \leq \|A\|^n C_p \frac{\|X\|^n_{p\text{-var};[a,t]}}{\Gamma(1+\frac{n}{p})}, \tag{2.96}$$

for $n \geq 1$. Thus the series

$$\Phi_{t,a} = \sum_{n=0}^{\infty} A^{\otimes n}\big(X^n_{[a,t]}\big) \ \in \ \mathbf{L}(W,W) \tag{2.97}$$

converges absolutely and uniformly for $t \in [a,b]$. In particular, $t \mapsto \Phi_{t,a}$ is continuous and

$$Y_t = \Phi_{t,a} Y_a = \sum_{n=0}^{\infty} A^{\otimes n}\big(X^n_{[a,t]}\big) Y_a \tag{2.98}$$

is well-defined and continuous.

3) Convergence in p-variation By (2.91), we have $Y^{(m)} \to Y$ uniformly on $[a,b]$. Furthermore, by Chen's identity, for $u < v$,

$$(Y - Y^{(m)})_v - (Y - Y^{(m)})_u = \sum_{\substack{k\geq 0,\ l\geq 1 \\ k+l \geq m+1}} A^{\otimes l}\big(X^l_{[u,v]}\big) A^{\otimes k}\big(X^k_{[a,u]}\big) Y_a. \tag{2.99}$$

The factorial decay estimate (2.95) implies that this double series is absolutely convergent and that there exists a sequence $\eta_m \to 0$ such that

$$\|(Y - Y^{(m)})_v - (Y - Y^{(m)})_u\| \leq \|Y_a\| \eta_m \|X\|_{p\text{-var};[u,v]}. \tag{2.100}$$

Therefore, for any partition $\{t_i\}_{i=0}^N$ of $[s,t]$,

$$\begin{aligned} \sum_{i=0}^{N-1} \|(Y - Y^{(m)})_{t_{i+1}} - (Y - Y^{(m)})_{t_i}\|^p &\leq \|Y_a\|^p \eta_m^p \sum_{i=0}^{N-1} \|X\|^p_{p\text{-var};[t_i,t_{i+1}]}, \\ &\leq \|Y_a\|^p \eta_m^p \|X\|^p_{p\text{-var};[s,t]}. \end{aligned} \tag{2.101}$$

Taking the supremum over partitions gives

$$\|Y - Y^{(m)}\|_{p\text{-var};[s,t]} \leq \|Y_a\| \eta_m \|X\|_{p\text{-var};[s,t]} \longrightarrow 0. \tag{2.102}$$

Therefore, $Y^{(m)}$ converges to Y in the $p-$variation sense. Furthermore, by the bilinear continuity of the Young integral [Lyons et al. 2007, Theorem 1.16],

$$\int_a^t AY_s^{(m)}\mathrm{d}X_s \longrightarrow \int_a^t AY_s\mathrm{d}X_s \tag{2.103}$$

for all $t \in [a,b]$. Passing to the limit in $Y_t^{(m+1)} = Y_a + \int_a^t A\,Y_s^{(m)}\,\mathrm{d}X_s$ yields

$$Y_t \;=\; Y_a \;+\; \int_a^t A\,Y_s\,\mathrm{d}X_s, \tag{2.104}$$

so Y solves (2.84).

4) Uniqueness Suppose $\tilde{Y}$ is another solution of (2.84) with the same initial condition Y_a. Then $Z = Y - \tilde{Y}$ satisfies $Z_a = 0$ and

$$Z_t = \int_a^t AZ_s\mathrm{d}X_s. \tag{2.105}$$

Applying the Picard expansion of Step 1 with initial datum 0 gives $Z_t \equiv 0$ on $[a,b]$. Thus the solution is unique. □

Theorem 2.37 shows that the solution of a linear CDE is determined by the signature of the driving path. More generally, the same holds for any CDE satisfying the uniqueness conditions of Theorem 2.34 [Hambly et al. 2010; Boedihardjo et al. 2016]. This motivates the construction of numerical methods based on the signature, and the Log-ODE method is one such approach.

2.5 The Log-ODE Method

The Log-ODE method is an efficient and accurate method for approximating the solution of a CDE [Castell et al. 1995; Boutaib et al. 2013]. On a given interval, it replaces the original CDE by an autonomous ODE whose vector field is constructed from the log-signature of the driving path over that interval together with differential information about the vector field. This construction is motivated by the fact that the log-signature $\log(S_{[a,b]}(X))$ belongs to the Lie series $\mathfrak{L}((V))$.

2.5.1 The Free Lie Algebra

Definition 2.38 (Free Lie Algebra [Reutenauer 1993])**.** *Let X be a non-empty set, L_0 and L be Lie algebras, and $\phi : X \to L_0$ be a map. The Lie algebra L_0 is said to be*

the free Lie algebra generated by X if for all maps $f : X \to L$, there exists a unique Lie algebra homomorphism $g : L_0 \to L$ such that $g \circ \phi = f$.

Theorem 2.39. *The space of elements of $\mathfrak{L}((V))$ with only finitely many nonzero terms, denoted $\mathfrak{L}(V)$, is the free Lie algebra generated by V [Reutenauer 1993].*

For each $N \geq 1$, the truncated log-signature $\log(S^N_{[a,b]}(X)) \in \mathfrak{L}^N(V)$ may be viewed as an element of $\mathfrak{L}(V)$ by setting all terms of degree greater than N equal to 0. Furthermore, as will be discussed in detail in Section 3.2.1, the smooth vector fields on W form a Lie algebra with Lie bracket defined pointwise by

$$[f, g](p) = Dg(p)[f(p)] - Df(p)[g(p)] \tag{2.106}$$

for smooth $f, g : W \to W$ and all $p \in W$, where Df is the Fréchet derivative of f. For example, if $W = \mathbb{R}^d$, $f(p) = Bp$, and $g(p) = Cp$ with $B, C \in \mathbb{R}^{d\times d}$, then

$$[f, g](p) = (CB - BC)p = -[B, C]\, p, \tag{2.107}$$

where the left hand side Lie bracket is for vector fields and the right hand side Lie bracket is for matrices. Therefore, assuming that $f(\cdot)v$ is a smooth vector field on W for all $v \in V$, Theorem 2.39 implies there exists a Lie algebra homomorphism $\bar{f}$ from $\mathfrak{L}(V)$ to the smooth vector fields on W. Since the map $\bar{f}$ is a Lie algebra homomorphism, it is determined recursively by

$$\bar{f}(\cdot)v = f(\cdot)v, \quad v \in V \tag{2.108}$$

and

$$\bar{f}(\cdot)[v_1, v_2] = [\bar{f}(\cdot)v_1, \bar{f}(\cdot)v_2] \tag{2.109}$$

for $v_1, v_2 \in \mathfrak{L}(V)$ [Lyons 2014]. For each $N \geq 1$,

$$F^N(\cdot) = \bar{f}\,(\cdot) \log(S^N(X)_{[a,b]}) \tag{2.110}$$

is a well-defined vector field on W. To illustrate the structure, consider a finite-dimensional driving path $X_t = (X^1_t, \ldots, X^d_t)$. Then $F^2(\cdot)$ can be written explicitly as

$$\begin{aligned}
F^2(\cdot) &= \bar{f}(\cdot) \log(S^2(X)_{[a,b]}), \\
&= \bar{f}(\cdot) \left(X^1_{[a,b]} + X^2_{[a,b]} - \frac{1}{2} X^1_{[a,b]} \otimes X^1_{[a,b]} \right), \\
&= \bar{f}(\cdot) \left(\sum_{i=1}^{d} S^{(i)}_{[a,b]} e_i + \frac{1}{2} \sum_{1\leq i<j\leq d} \left(S^{(i,j)}_{[a,b]} - S^{(j,i)}_{[a,b]} \right) [e_i, e_j] \right), \\
&= \sum_{i=1}^{d} S^{(i)}_{[a,b]} f(\cdot)e_i + \frac{1}{2} \sum_{1\leq i<j\leq d} \left(S^{(i,j)}_{[a,b]} - S^{(j,i)}_{[a,b]} \right) [f(\cdot)e_i, f(\cdot)e_j],
\end{aligned} \tag{2.111}$$

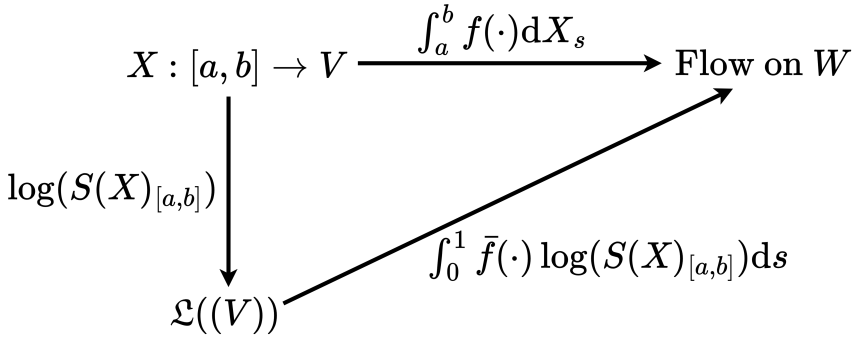


Figure 2.3: A schematic diagram of the Log-ODE method, where $\int_a^b f(\cdot)\mathrm{d}X$ is the CDE to be solved, the log-signature of X lives in $\mathfrak{L}((V))$, and the Log-ODE method constructs an approximating autonomous ODE using the unique Lie algebra homomorphism extending f from V to the free Lie algebra generated by V, denoted $\bar{f}$.

where the second line uses the explicit form of the log-signature up to level 2 derived in (2.59), the third line rewrites the first and second levels in the basis $(e_1, \ldots, e_d)$ of V, and the final line applies the Lie algebra homomorphism $\bar{f}$.

2.5.2 The Log-ODE Approximation

Given appropriate conditions on the convergence of F^N as $N \to \infty$, the Log-ODE method recovers the solution of the CDE

$$Y_b = Y_a + \int_a^b f(Y_s)\mathrm{d}X_s \tag{2.112}$$

by solving the autonomous ODE

$$\bar{Y}_1 = \bar{Y}_0 + \int_0^1 \bar{f}\left(\bar{Y}_s\right)\log(S(X)_{[a,b]})\mathrm{d}s \tag{2.113}$$

with initial condition $\bar{Y}_0 = Y_a$, and then setting $Y_b = \bar{Y}_1$. This is represented schematically in Figure 2.3.

When X is continuously differentiable and $f(\cdot)v$ is a linear vector field for all $v \in V$, this construction coincides with the classical Magnus expansion [Magnus 1954]. To see this, let $\dot{X}_t$ denote the time derivative of X_t and let $A \in \mathbf{L}(V, \mathbf{L}(W, W))$ be such that

$$f(y)v = A(v)y \tag{2.114}$$

for all $y \in W$ and $v \in V$. Then the CDE may be written as

$$\frac{\mathrm{d}Y_t}{\mathrm{d}t} = A(\dot{X}_t)Y_t, \tag{2.115}$$

and the classical Magnus expansion expresses the solution in the form

$$Y_b = \exp\Big(\Omega^{(1)}_{b,a} + \Omega^{(2)}_{b,a} + \cdots\Big)\, Y_a, \tag{2.116}$$

Equivalently, if $\bar{Y} : [0,1] \to W$ solves

$$\bar{Y}_1 = \bar{Y}_0 + \int_0^1 \Big(\Omega^{(1)}_{b,a} + \Omega^{(2)}_{b,a} + \cdots\Big)\, \bar{Y}_s \,\mathrm{d}s, \tag{2.117}$$

with $\bar{Y}_0 = Y_a$, then $\bar{Y}_1 = Y_b$. Since A is linear and $X^1_{[a,b]} = \int_a^b \dot{X}_{u_1}\,\mathrm{d}u_1$, the first term in the infinite series of (2.116) is simply

$$\Omega^{(1)}_{b,a} = \int_a^b A(\dot{X}_{u_1})\,\mathrm{d}u_1 = A(X^1_{[a,b]}) = \sum_{i=1}^{d} S^{(i)}_{[a,b]} A(e_i), \tag{2.118}$$

aligning with the first term of (2.111). The second term is

$$\begin{aligned}\Omega^{(2)}_{b,a} &= \frac{1}{2}\int_a^b \int_a^{u_1} [A(\dot{X}_{u_1}), A(\dot{X}_{u_2})]\,\mathrm{d}u_2\,\mathrm{d}u_1 \\ &= \frac{1}{2}\int_a^b \int_a^{u_1} \Big(A(\dot{X}_{u_1})A(\dot{X}_{u_2}) - A(\dot{X}_{u_2})A(\dot{X}_{u_1})\Big)\,\mathrm{d}u_2\,\mathrm{d}u_1.\end{aligned} \tag{2.119}$$

Expanding $A(\dot{X}_{u_1})$ and $A(\dot{X}_{u_2})$ in the basis $(e_1, \ldots, e_d)$ and using bilinearity of the Lie bracket gives

$$\begin{aligned}\Omega^{(2)}_{b,a} &= \frac{1}{2}\int_a^b \int_a^{u_1} \Big(A(\dot{X}_{u_1})A(\dot{X}_{u_2}) - A(\dot{X}_{u_2})A(\dot{X}_{u_1})\Big)\,\mathrm{d}u_2\,\mathrm{d}u_1 \\ &= \frac{1}{2}\sum_{i,j=1}^{d} \int_a^b \int_a^{u_1} \dot{X}^i_{u_1}\dot{X}^j_{u_2}\,[A(e_i), A(e_j)]\,\mathrm{d}u_2\,\mathrm{d}u_1 \\ &= \frac{1}{2}\sum_{i,j=1}^{d} S^{(j,i)}_{[a,b]}\,[A(e_i), A(e_j)].\end{aligned} \tag{2.120}$$

We now group together the terms indexed by (i,j) and (j,i). Since $[A(e_i), A(e_i)] = 0$ and $[A(e_j), A(e_i)] = -[A(e_i), A(e_j)]$, this becomes

$$\Omega^{(2)}_{b,a} = \frac{1}{2}\sum_{1\le i<j\le d} \Big(S^{(i,j)}_{[a,b]} - S^{(j,i)}_{[a,b]}\Big)\,[A(e_j), A(e_i)], \tag{2.121}$$

which is exactly the second sum in (2.111), since for the linear vector fields $f(\cdot)e_i(y) = A(e_i)y$,

$$[f(\cdot)e_i, f(\cdot)e_j](y) = [A(e_j), A(e_i)]y, \tag{2.122}$$

where the first Lie bracket is for vector fields and the second Lie bracket is for matrices. Hence, at depth 2, the Log-ODE vector field reproduces exactly the first two

terms of the Magnus expansion. In fact, this agreement persists term by term at every order, with each term built from iterated integrals of the driving path and iterated Lie brackets of the matrices $A(v)$.

For finite-dimensional V and W, a sufficient condition ensuring local convergence of the Magnus expansion is [Moan et al. 2008],

$$\int_a^b \|A(\dot{X}_s)\|_2 \mathrm{d}s < \pi. \tag{2.123}$$

When X is continuously differentiable and $f(\cdot)v$ is non-linear, the same construction yields the Chen-Strichartz series [Strichartz 1987]. For finite-dimensional V and W, local convergence of Chen-Strichartz holds under an analytic growth condition on the non-linear vector fields [Strichartz 1987]. The approach was first extended to the rough-path setting by Castell et al. [1995], where it is known as the Log-ODE method.

Assuming it converges, exponentiating the infinite series in (2.116) recovers the exact flow, which agrees with the direct solution formula of Theorem 2.37,

$$\Phi_{t,a} = \sum_{n=0}^{\infty} A^{\otimes n}\big(X^n_{[a,t]}\big). \tag{2.124}$$

Truncating the signature expansion of the flow

$$\Phi^N_{t,a} = \sum_{n=0}^{N} A^{\otimes n}\big(X^n_{[a,t]}\big), \tag{2.125}$$

gives an alternative approach to approximating the solution of a CDE. This approach truncates an expansion of the flow, whereas the Log-ODE method truncates the flow in its logarithmic, or generator, representation. Equivalently, the Log-ODE method may be interpreted as replacing the original driving path on each interval by a rough path whose truncated log-signature agrees with that of the original path up to degree N, and whose higher-order terms are zero. The approximation is then obtained by solving the same class of controlled system against this approximating path, rather than by truncating the flow map after it has been computed. This is often desirable, since it keeps the approximation within the class of solutions of the same CDE, better preserving any structural or physical properties encoded by the model.

Given a set of intervals $a = r_0 < \cdots < r_m = b$, a depth$-N$ Log-ODE method

approximates the solution of a CDE via

$$\tilde{Y}_{r_{i+1}} = \tilde{Y}_{r_i} + \int_{r_i}^{r_{i+1}} \bar{f}\left(\tilde{Y}_s\right) \frac{\log(S^N(X)_{[r_i, r_{i+1}]})}{r_{i+1} - r_i} \mathrm{d}s, \tag{2.126}$$

where $\tilde{Y}_a = Y_a$ and the integral's time has been rescaled to match that of the original CDE. Since this approach truncates to the depth$-N$ log-signature, the vector field no longer needs to be smooth, but only $\mathrm{Lip}(\gamma)$ for $\gamma > N - 1$. Conversely, the smoothness of the vector field determines the highest truncation depth usable in the Log-ODE method. For a given set of intervals, this approximation is not guaranteed to converge as $N \to \infty$. However, the error $\|Y_b - \tilde{Y}_b\|$ can be quantified, even for rough driving paths and infinite dimensional Banach spaces [Boutaib et al. 2013]. A recent development has been the introduction of an algorithm which adaptively updates N and $\{r_i\}_{i=0}^m$ [Bayer et al. 2023].

2.6 Conclusion

This chapter developed the mathematical framework used throughout this thesis for modelling continuously evolving data. It introduced CDEs as a way to describe dynamics driven by a continuous path, showed how the signature provides a graded representation of that path with strong algebraic and approximation properties, and explained how the log-signature removes the algebraic redundancy of the signature while retaining the genuinely new geometric information. In the linear setting, the solution of a CDE was shown to be determined explicitly by the signature of the driving path. The chapter then showed how the log-signature leads naturally to the Log-ODE method, which replaces the original CDE on each interval by an autonomous ODE built from the truncated log-signature of the driving path and iterated Lie brackets of the vector field. Together, these ideas establish the main mathematical objects and constructions used in the remainder of the thesis: paths are the fundamental data type, signatures and log-signatures summarise them over intervals, and CDEs describe how states evolve in response to them.

The existence and uniqueness theorems, Theorems 2.33 and 2.34, together with the Log-ODE method, show that $\mathrm{Lip}(\gamma)$ regularity plays an important role in the theory of CDEs. The next chapter studies this notion of regularity in detail. Furthermore, it formalises the Lie bracket of two $\mathrm{Lip}(\gamma)$ functions, a key ingredient in the Log-ODE method.

Chapter 3

Lip(γ) Functions

> *I am convinced that it is impossible to know the parts without knowing the whole, any more than we can know the whole without a detailed knowledge of the parts.*

—Blaise Pascal, *Penseés* (1670), translation by Martin Turnell (1962)

3.1 Introduction

The analysis of CDEs, particularly when employing numerical techniques such as the Log-ODE method, depends on the regularity of the driving vector field. As established by Lyons [1994], Lip(γ) is the correct notion of regularity for guaranteeing the existence and uniqueness of solutions to CDEs. Furthermore, the degree of regularity γ directly determines the maximum truncation depth N that can be used by the Log-ODE method. This chapter relates Lip(γ) to more classical notions of differentiability, develops the Lie bracket of Lip(γ) vector fields, and proves Lemma 3.35, which gives a new explicit bound for the Lip(γ) norm of the composition of two Lip(γ) functions in the case $1 < \gamma \leq 2$.

Unlike Chapter 2, which introduced the main mathematical objects used throughout the thesis, this chapter is concerned primarily with the regularity theory needed to apply those constructions when the vector field of a CDE is parametrised by a neural network. In particular, Chapter 4 uses Lemma 3.35 to control the Lip(γ) norm of the neural network vector fields, thereby justifying the use of the Lie brackets of those vector fields when applying the Log-ODE method to NCDEs. Although this

chapter contains material of independent mathematical interest, it is not essential for readers primarily interested in the continuous-time machine learning models developed in Chapters 4 and 5.

The notion of $\mathrm{Lip}(\gamma)$ functions originates in the work of Whitney [1934], who studied collections of derivative data on closed subsets of $\mathbb{R}^n$. The modern formulation was later given by Stein [1970]. A core result is the Stein-Whitney extension theorem, given in Section 3.3.4 as Theorem 3.31, which states that any $\mathrm{Lip}(\gamma)$ function defined on a closed subset E of a finite-dimensional Banach space can be extended to the whole space, with a bound on the norm of the extension that is independent of E.

The definition of $\mathrm{Lip}(\gamma)$ regularity given by Stein [1970] applies to functions defined on arbitrary subsets of a Banach space. Lemma 3.18 shows that on open convex subsets, the space of $\mathrm{Lip}(\gamma)$ functions coincides with the space of k times Fréchet differentiable functions with bounded value and derivatives, whose k^{th} derivative satisfies an α-Hölder bound, denoted $C_b^{k,\alpha}$. However, the natural inclusion of a $C_b^{k+1,\alpha}$ function in $C_b^{k,\alpha}$ does not hold globally, whereas Boutaib [2016] proved that $\mathrm{Lip}(\gamma)$ spaces are globally nested with respect to γ. Similarly, the Lie bracket of two $C_b^{k,\alpha}$ vector fields need not be globally $C_b^{k-1,\alpha}$.

To the best of the author's knowledge, Section 3.3 gives the first formal treatment of the Lie bracket for $\mathrm{Lip}(\gamma)$ vector fields on arbitrary subsets of potentially infinite-dimensional Banach spaces. We prove that $[f, g] \in \mathrm{Lip}(\gamma - 1)$ for $f, g \in \mathrm{Lip}(\gamma)$ with $\gamma > 1$, and that the Jacobi identity holds in $\mathrm{Lip}(\gamma - 2)$ for $\gamma > 2$. This work builds on [Boutaib 2016, Chapter 3], which proved a number of fundamental results for $\mathrm{Lip}(\gamma)$ functions, including that the composition of two $\mathrm{Lip}(\gamma)$ functions is $\mathrm{Lip}(\gamma)$ and that these spaces satisfy the nesting property with respect to γ.

The final section of this chapter proves Lemma 3.35, an explicit bound on the norm of the composition of two $\mathrm{Lip}(\gamma)$ functions for $1 < \gamma \leq 2$. This builds on the work of Cass et al. [2012] and Boutaib [2016], who proved using different techniques that the composition of two $\mathrm{Lip}(\gamma)$ functions is $\mathrm{Lip}(\gamma)$, with a norm bounded up to a finite unknown constant C_γ. Our result allows us to explicitly bound the norm of a certain class of neural networks in Section 4.4.2, laying the theoretical groundwork for applying the Log-ODE method to NCDEs.

For a Banach space V, this chapter assumes that the $\{V^{\otimes n}\}_{n=1}^{\infty}$ are equipped with a family of reasonable tensor algebra norms, as outlined in Section 2.2.2. As previously discussed, these conditions ensure that for all $v \in V^{\otimes n}$ and $w \in V^{\otimes m}$,

$$\|v \otimes w\|_{V^{\otimes(n+m)}} = \|v\|_{V^{\otimes n}} \|w\|_{V^{\otimes m}}. \tag{3.1}$$

Therefore, the equivalence property of Boutaib [2016],

$$\|v\|_{V^{\otimes n}} \|w\|_{V^{\otimes m}} \leq c\|v \otimes w\|_{V^{\otimes(n+m)}}, \tag{3.2}$$

holds with equality and $c = 1$, allowing us to directly use their results.

3.2 Differentiable Functions

This section reviews three standard notions of function regularity: smooth, C^k, and $C^{k,\alpha}$, together with the Lie bracket of two functions. We develop their basic properties and provide proofs in this familiar setting, both to fix notation and to introduce the ideas and techniques that will later be applied when establishing results for the Lie bracket of $\mathrm{Lip}(\gamma)$ functions.

3.2.1 Smooth Vector Fields

Just as the ordered interactions captured by the signature give rise to multilinear objects, the iterated derivatives of a function naturally take values in spaces of multilinear maps. The first derivative records the best linear approximation to a function, while the second and higher derivatives describe how this approximation changes under simultaneous perturbations in multiple directions. For sufficiently regular functions, these higher derivatives are symmetric, since the order in which the input directions are differentiated does not matter.

Let U and V be Banach spaces. A symmetric k-linear map is a map

$$A : U \times \cdots \times U \to V \tag{3.3}$$

which is linear in each argument and unchanged by permuting its inputs. By the universal property of the tensor product, bounded symmetric k-linear maps may be identified with bounded linear maps $A : U^{\otimes k} \to V$ such that

$$A(u_{\sigma(1)} \otimes \cdots \otimes u_{\sigma(k)}) = A(u_1 \otimes \cdots \otimes u_k) \tag{3.4}$$

for all $u_1, \dots, u_k \in U$ and every permutation σ of $\{1, \dots, k\}$. We write $\mathbf{L}_{\mathrm{sym}}(U^{\otimes k}, V)$ for this space, and for $k = 0$ set

$$\mathbf{L}_{\mathrm{sym}}(U^{\otimes 0}, V) \cong V. \tag{3.5}$$

This is the space in which the successive derivatives of a sufficiently regular map between Banach spaces take values.

Definition 3.1 (Fréchet derivative [Luenberger 1969])**.** *Let U and V be Banach spaces and $E \subset U$ open. A map $f : E \to V$ is Fréchet differentiable at a point $p \in E$ if there exists a bounded linear operator $Df(p) \in \mathbf{L}(U, V)$ such that*

$$\lim_{\substack{h \to 0 \\ p+h \in E}} \frac{\|f(p+h) - f(p) - Df(p)[h]\|}{\|h\|} = 0. \tag{3.6}$$

The map $Df(p)$ is called the Fréchet derivative of f at p.

Definition 3.2 (Smooth function)**.** *Let U and V be Banach spaces and $E \subset U$ be an open set. A map $f : E \to V$ is called smooth if the iterated Fréchet derivatives*

$$D^j f : E \to \mathbf{L}_{\mathrm{sym}}(U^{\otimes j}, V) \tag{3.7}$$

exist and are continuous for all $j \in \mathbb{N}_0$, where $D^0 f = f$.

Let $C^\infty(E, V)$ denote the space of all smooth functions $f : E \to V$.

Definition 3.3 (Lie bracket on a Banach space)**.** *Let U be a Banach space and $E \subset U$ an open set. For $f, g \in C^\infty(E, U)$, their Lie bracket $[f, g] \in C^\infty(E, U)$ is defined pointwise by*

$$[f, g](p) = Dg(p)[f(p)] - Df(p)[g(p)], \tag{3.8}$$

for $p \in E$.

Remark 3.4. *The chosen sign convention*

$$[f, g](p) = Dg(p)[f(p)] - Df(p)[g(p)] \tag{3.9}$$

agrees with the usual definition of the Lie bracket of vector fields in differential geometry, ensuring that

$$[f, g] = f \circ g - g \circ f \tag{3.10}$$

when f and g are viewed as derivations acting on smooth functions [Lee 2013].

The space $C^\infty(E,U)$, endowed with the bracket operation defined above, is a Lie algebra [Lang 1999, Chapter 5]:

1. The bracket is bilinear:

$$[af+bg,h]=a[f,h]+b[g,h], \tag{3.11}$$

and

$$[h,af+bg]=a[h,f]+b[h,g], \tag{3.12}$$

hold for all $a,b\in\mathbb{R}$ and $f,g,h\in C^\infty(E,U)$.

2. The bracket is antisymmetric:

$$[f,g]=-[g,f], \tag{3.13}$$

holds for all $f,g\in C^\infty(E,U)$.

3. The bracket satisfies the Jacobi identity:

$$[f,[g,h]]+[g,[h,f]]+[h,[f,g]]=0, \tag{3.14}$$

holds for all $f,g,h\in C^\infty(E,U)$.

3.2.2 C^k Vector Fields

We now move from smooth vector fields to the weaker setting of finite differentiability.

Definition 3.5 (C^k function)**.** *Let U and V be Banach spaces, $E\subset U$ be open, and $k\in\mathbb{N}_0$. A map $f:E\to V$ is called C^k if the iterated Fréchet derivatives*

$$D^jf:E\to\mathbf{L}_{\mathrm{sym}}\big(U^{\otimes j},V\big),\qquad j=0,\dots,k, \tag{3.15}$$

exist and are continuous.

Let $C^k(E,V)$ denote the set of all C^k functions from E to V. A first basic observation is that a C^k function is C^l for all $l\in\mathbb{N}_0$ satisfying $l<k$. A second is that differentiating a C^k function produces a C^{k-1} function. To compare the derivatives of Df with the higher derivatives of f, we use the canonical currying identification

$$\mathbf{L}_{\mathrm{sym}}\big(U^{\otimes j},\mathbf{L}(U,V)\big)\cong\mathbf{L}\big(U^{\otimes(j+1)},V\big), \tag{3.16}$$

given by

$$A(u_1,\dots,u_j)(u_{j+1})\longleftrightarrow\widetilde{A}(u_1,\dots,u_{j+1}). \tag{3.17}$$

Lemma 3.6. *Let U and V be Banach spaces, $E \subset U$ open, and $k \in \mathbb{N}$. If $f \in C^k(E,V)$, then its Fréchet derivative*

$$Df : E \longrightarrow \mathbf{L}(U,V) \tag{3.18}$$

belongs to $C^{k-1}\big(E, \mathbf{L}(U,V)\big)$.

Proof. By definition of C^k, the iterated derivatives $D^j f : E \to \mathbf{L}_{\mathrm{sym}}\big(U^{\otimes j}, V\big)$ exist and are continuous for all $0 \le j \le k$. For every $0 \le j \le k-1$,

$$D^j\big(Df\big) = D^{j+1} f, \tag{3.19}$$

under the currying convention $\mathbf{L}_{\mathrm{sym}}\big(U^{\otimes j}, \mathbf{L}(U,V)\big) \cong \mathbf{L}\big(U^{\otimes (j+1)}, V\big)$. The right-hand side exists and is continuous by assumption, therefore Df possesses continuous iterated derivatives up to order $k-1$, i.e. $Df \in C^{k-1}\big(E, \mathbf{L}(U,V)\big)$. □

To define the bounded C^k spaces, we use a uniform norm on each derivative level. For $j \in \mathbb{N}_0$, let $\|\cdot\|_j$ denote the norm on $\mathbf{L}_{\mathrm{sym}}(U^{\otimes j}, V)$ given by

$$\|\cdot\|_j = \begin{cases} \|\cdot\|_V, & j = 0, \\ \|\cdot\|_{\mathrm{op}}, & j \in \mathbb{N}. \end{cases} \tag{3.20}$$

Under the identification $\mathbf{L}_{\mathrm{sym}}(U^{\otimes 0}, V) \cong V$, this treats the case $j = 0$ uniformly with the higher-order cases.

Definition 3.7 (C_b^k space and norm)**.** *Let U and V be Banach spaces, let $E \subset U$ be open, and fix $k \in \mathbb{N}_0$. For $f \in C^k(E,V)$ define*

$$\|f\|_{C^k} = \max_{0 \le j \le k} \sup_{p \in E} \|D^j f(p)\|_j. \tag{3.21}$$

The bounded C^k space is

$$C_b^k(E,V) = \{ f \in C^k(E,V) : \|f\|_{C^k} < \infty \}. \tag{3.22}$$

Definition 3.8 (C^k Lie bracket)**.** *Let $f \in C^{k_f}(E,U)$ and $g \in C^{k_g}(E,U)$ with $k_f, k_g \in \mathbb{N}$. Their Lie bracket is defined pointwise by*

$$[f,g](p) = Dg(p)[f(p)] - Df(p)[g(p)], \qquad p \in E. \tag{3.23}$$

The definition of the Lie bracket is the same as that for smooth functions, Definition 3.3. However, for C^k functions the Lie bracket reduces the regularity by one.

Lemma 3.9. *Let $f \in C^{k_f}(E,U)$ and $g \in C^{k_g}(E,U)$ with $k_f, k_g \in \mathbb{N}$. Then $[f,g] \in C^{\min(k_f,k_g)-1}(E,U)$.*

Proof. Let $k = \min(k_f, k_g)$. Then $f, g \in C^k(E,U)$. By Lemma 3.6 and the definition of C^k, we have

$$f, g \in C^{k-1}(E,U) \quad \text{and} \quad Dg, Df \in C^{k-1}(E, \mathbf{L}(U,U)). \tag{3.24}$$

Let $B : \mathbf{L}(U,U) \times U \to U$ be defined by

$$B(A,v) = A[v]. \tag{3.25}$$

Then B is a continuous bilinear function, and hence smooth [Lang 1999, Chapter 1, Proposition 3.8]. Since the functions

$$(Dg, f)(p) = (Dg(p), f(p)) \quad \text{and} \quad (Df, g)(p) = (Df(p), g(p)) \tag{3.26}$$

are $C^{k-1}(E, \mathbf{L}(U,U) \times U)$, then $B \circ (Dg, f)$ and $B \circ (Df, g)$ are both $C^{k-1}(E,U)$ [Lang 1999, Chapter 1, Proposition 3.2]. Since all terms on the RHS of (3.23) are C^{k-1}, then $[f,g] \in C^{k-1}(E,U)$. □

Lemma 3.9 means the Lie bracket of two C^k functions does not necessarily belong to C^k, and it is therefore not a Lie algebra. However, the Lie bracket does satisfy bilinearity and anti-symmetry, both of which follow directly from the definition. Furthermore, the Lie bracket satisfies the Jacobi identity.

Lemma 3.10. *Let $f, g, h \in C^k(E,U)$ with $k \in \mathbb{N}$ satisfying $k \geq 2$. Then*

$$[f,[g,h]] + [g,[h,f]] + [h,[f,g]] = 0. \tag{3.27}$$

Proof. Fix $p \in E$ and define:

$$\begin{aligned} x = f(p), \quad y = g(p), \quad z = h(p), \\ A = Df(p), \quad B = Dg(p), \quad C = Dh(p), \\ \mathbb{A} = D^2 f(p), \quad \mathbb{B} = D^2 g(p), \quad \mathbb{C} = D^2 h(p), \end{aligned} \tag{3.28}$$

where $A, B, C \in \mathbf{L}(U,U)$ and $\mathbb{A}, \mathbb{B}, \mathbb{C} \in \mathbf{L}_{\text{sym}}(U^{\otimes 2}, U)$. Then

$$\begin{aligned} [f,[g,h]](p) &= -\mathbb{B}[z,x] - B[C[x]] + \mathbb{C}[y,x] + C[B[x]] + A[B[z]] - A[C[y]], \\ [g,[h,f]](p) &= -\mathbb{C}[x,y] - C[A[y]] + \mathbb{A}[z,y] + A[C[y]] + B[C[x]] - B[A[z]], \\ [h,[f,g]](p) &= -\mathbb{A}[y,z] - A[B[z]] + \mathbb{B}[x,z] + B[A[z]] + C[A[y]] - C[B[x]]. \end{aligned} \tag{3.29}$$

By the symmetry of the second derivatives,

$$\mathbb{B}[x,z] - \mathbb{B}[z,x] + \mathbb{C}[y,x] - \mathbb{C}[x,y] + \mathbb{A}[z,y] - \mathbb{A}[y,z] = 0, \tag{3.30}$$

and all the other terms cancel. Therefore

$$[f,[g,h]] + [g,[h,f]] + [h,[f,g]] = 0 \tag{3.31}$$

for all p, concluding the proof. □

3.2.3 $C^{k,\alpha}$ Vector Fields

We now strengthen finite differentiability by requiring the highest derivative to vary in a controlled way. This leads to the classical Hölder spaces $C^{k,\alpha}$, which refine the C^k spaces by measuring not only the existence of derivatives up to order k, but also the regularity of the k-th derivative itself. These spaces will be useful for comparison with $\mathrm{Lip}(\gamma)$ regularity in Section 3.3, since the two notions agree on open convex domains but behave differently at a global level.

Definition 3.11 ($C^{k,\alpha}$ function)**.** *Let U and V be Banach spaces, $E \subset U$ be open, $k \in \mathbb{N}_0$, and $\alpha \in (0,1]$. A map $f : E \to V$ is $C^{k,\alpha}$ if*

1. *$f \in C^k(E,V)$ and*
2. *its k-th derivative is α-Hölder continuous on E:*

$$[D^k f]_\alpha = \sup_{\substack{p,q\in E\\ p\neq q}} \frac{\|D^k f(p) - D^k f(q)\|_k}{\|p-q\|_U^\alpha} < \infty. \tag{3.32}$$

The space of all $C^{k,\alpha}$ functions from E to V is denoted $C^{k,\alpha}(E,V)$.

Lemma 3.12. *If $f \in C^{k,\alpha}(E,V)$ with $k \in \mathbb{N}$, then its Fréchet derivative*

$$Df : E \longrightarrow \mathbf{L}(U,V) \tag{3.33}$$

belongs to $C^{k-1,\alpha}\big(E, \mathbf{L}(U,V)\big)$.

Proof. Because $f \in C^k(E,V)$, Lemma 3.6 gives $Df \in C^{k-1}\big(E, \mathbf{L}(U,V)\big)$. Under the currying convention $\mathbf{L}_{\mathrm{sym}}\big(U^{\otimes j}, \mathbf{L}(U,V)\big) \cong \mathbf{L}\big(U^{\otimes(j+1)}, V\big)$, we have

$$\sup_{\substack{p,q\in E\\ p\neq q}} \frac{\|D^{k-1}(Df)(p) - D^{k-1}(Df)(q)\|_k}{\|p-q\|_U^\alpha} = \sup_{\substack{p,q\in E\\ p\neq q}} \frac{\|D^k f(p) - D^k f(q)\|_k}{\|p-q\|_U^\alpha} = [D^k f]_\alpha. \tag{3.34}$$

Hence $[D^{k-1}(Df)]_\alpha = [D^k f]_\alpha$ and the claim follows. □

Definition 3.13 ($C_b^{k,\alpha}$ space and norm)**.** *For $f \in C^{k,\alpha}(E,V)$ set*

$$\|f\|_{C^{k,\alpha}} := \max\left\{\max_{0\le j\le k}\sup_{p\in E}\|D^j f(p)\|_j, [D^k f]_\alpha\right\}. \tag{3.35}$$

The bounded $C^{k,\alpha}$ space is

$$C_b^{k,\alpha}(E,V) = \left\{f \in C^{k,\alpha}(E,V) : \|f\|_{C^{k,\alpha}} < \infty\right\}. \tag{3.36}$$

Definition 3.14 ($C^{k,\alpha}$ Lie bracket)**.** *Let $f, g \in C^{k,\alpha}(E,U)$ with $k \in \mathbb{N}$. Their Lie bracket is defined pointwise by*

$$[f,g](p) = Dg(p)[f(p)] - Df(p)[g(p)], \qquad p \in E. \tag{3.37}$$

However, unlike C^k functions, the Lie bracket of two $C^{k,\alpha}$ functions is not necessarily $C^{k-1,\alpha}$. In fact, $C^{k,\alpha}$ functions don't have an inclusion with respect to k. For example, consider $E \subset \mathbb{R}$ with $E = (-\infty,0)\cup(0,\infty)$ and $f : E \to \mathbb{R}$ defined by

$$f(x) = \begin{cases} x-1, & x \in (-\infty,0), \\ x+1, & x \in (0,\infty), \end{cases} \tag{3.38}$$

with $Df = 1$. Then $[Df]_1 = 0$ and $f \in C^{1,1}(E,\mathbb{R})$. However, $[f]_1$ is infinite, due to the discontinuity at 0. Therefore, $f \notin C^{0,1}(E,\mathbb{R})$. Furthermore, letting $g : E \to \mathbb{R}$ be defined by $g(x) = 2x$, then

$$[f,g] = \begin{cases} -2, & x \in (-\infty,0), \\ +2, & x \in (0,\infty), \end{cases} \tag{3.39}$$

so $[f,g] \notin C^{0,1}$ despite $f, g \in C^{1,1}$. The same results hold for $C_b^{k,\alpha}$ functions, which can be seen by replacing E with $(-1,0)\cup(0,1)$.

In the next section, we introduce $\mathrm{Lip}(\gamma)$ regularity, a global notion of regularity defined on both open and closed subsets, which will allow us to define the Lie bracket globally.

3.3 $\mathrm{Lip}(\gamma)$ Functions

3.3.1 Definition

Definition 3.15 ($\mathrm{Lip}(\gamma,E,V)$ [Stein 1970])**.** *Let U and V be two Banach spaces, $\gamma > 0$ be a real number, k be the non-negative integer such that $\gamma \in (k, k+1]$, E be a*

subset of U, and $f^0 : E \to V$ a function. For $j = 1, \ldots, k$, let $f^j : E \to \mathbf{L}_{\text{sym}}(U^{\otimes j}, V)$ be functions. The collection $(f^0, f^1, \ldots, f^k)$ is an element of $\text{Lip}(\gamma, E, V)$ *if there exists $M \geq 0$ such that the following conditions hold:*

- *For $j = 0, \ldots, k$,*

$$\sup_{p \in E} ||f^j(p)||_j \leq M, \tag{3.40}$$

- *For $j = 0, \ldots, k$, all $p, q \in E$,*

$$\left|\left| f^j(q) - \sum_{l=0}^{k-j} \frac{f^{j+l}(p)[(q-p)^{\otimes l}]}{l!} \right|\right|_j \leq M||q-p||_U^{\gamma - j}. \tag{3.41}$$

where

$$||\cdot||_j = \begin{cases} ||\cdot||_V, & j = 0, \\ ||\cdot||_{\text{op}}, & j = 1, \ldots, k. \end{cases} \tag{3.42}$$

When there is no confusion over E and V, the shorthand $\text{Lip}(\gamma)$ will be used. If a collection $f = (f^0, f^1, \ldots, f^k)$ is $\text{Lip}(\gamma)$, then the $\text{Lip}(\gamma)-$norm is the smallest M for which (3.40) and (3.41) hold. The $\text{Lip}(\gamma)-$norm is denoted by $||f||_{\text{Lip}(\gamma)}$. $\text{Lip}(\gamma, E, V)$ functions can be equivalently defined by considering a function defined on E which takes its values in the polynomials from U to V.

Definition 3.16 ($\text{Lip}(\gamma, E, V)$ as polynomials)**.** *Let U and V be two Banach spaces, $\gamma > 0$ be a real number, k be the non-negative integer such that $\gamma \in (k, k+1]$, E be a subset of U, and $\mathbf{P}^k(U, V)$ be the space of k^{th} order polynomial functions from U to V. The function $f : E \to \mathbf{P}^k(U, V)$ defined by*

$$f(x)(y) = P_x(y) = \sum_{l=0}^{k} \frac{f^l(x)[(y-x)^{\otimes l}]}{l!}. \tag{3.43}$$

for $f^j : E \to \mathbf{L}_{\text{sym}}(U^{\otimes j}, V)$ is an element of $\text{Lip}(\gamma, E, V)$ *if*

$$\sup_{x \in E} ||P_x^j(x)||_j \leq M \tag{3.44}$$

for $j = 0, \ldots, k$ and

$$\left|\left| P_y^j(y) - P_x^j(y) \right|\right|_j \leq M\|x-y\|_U^{\gamma - j} \tag{3.45}$$

for $j = 0, \ldots, k$ and all $x, y \in E$, where

$$\begin{aligned} P_x^j(y)[u_1 \otimes \cdots \otimes u_j] &= D_y^j P_x(y)[u_1 \otimes \cdots \otimes u_j], \\ &= \sum_{l=0}^{k-j} \frac{f^{j+l}(x)[u_1 \otimes \cdots \otimes u_j \otimes (y-x)^{\otimes l}]}{l!}. \end{aligned} \tag{3.46}$$

To further illustrate the definition of $\text{Lip}(\gamma)$, we will compare and contrast it with $C_b^{k,\alpha}$.

3.3.2 Comparison with $C_b^{k,\alpha}$

On open convex subsets of a Banach space, the definition of $\mathrm{Lip}(\gamma)$ coincides with the definition of $C_b^{k,\alpha}$, as will be shown in Lemma 3.18. This result will make use of the following lemma, which demonstrates that the constant in the bound of any Taylor remainder for a $C_b^{k,\alpha}$ function defined on an open convex set is bounded by the constant of the $\alpha-$Hölder bound on the k^{th} derivative.

Lemma 3.17. *Let U and V be Banach spaces, $E \subseteq U$ be an open convex set, and $f \in C_b^{k,\alpha}(E,V)$. Let $D^0 f = f$ and $M = [D^k f]_\alpha$. Then*

$$\sup_{p,q\in E} \frac{\left|\left|D^{k-i}f(q) - \sum_{l=0}^{i} \frac{D^{k-i+l}f(p)[(q-p)^{\otimes l}]}{l!}\right|\right|_{k-i}}{||q-p||_U^{\alpha+i}} \leq M, \tag{3.47}$$

for $i = 0, \ldots, k$.

Proof. We proceed by induction on i. Let $i \in \{0, \ldots, k-1\}$ and assume that

$$\sup_{p,q\in E} \frac{\left|\left|D^{k-i}f(q) - \sum_{l=0}^{i} \frac{D^{k-i+l}f(p)[(q-p)^{\otimes l}]}{l!}\right|\right|_{k-i}}{||q-p||_U^{\alpha+i}} \leq M. \tag{3.48}$$

We now show that the same estimate holds with i replaced by $i+1$. By the definition of $\|\cdot\|_{k-(i+1)}$, we have

$$\begin{aligned} &\sup_{p,q\in E} \frac{\left|\left|D^{k-(i+1)}f(q) - \sum_{l=0}^{i+1} \frac{D^{k-(i+1)+l}f(p)[(q-p)^{\otimes l}]}{l!}\right|\right|_{k-(i+1)}}{||q-p||_U^{\alpha+(i+1)}} \\ &\quad = \sup_{p,q\in E}\sup_{u\neq 0} \frac{\left|\left|D^{k-(i+1)}f(q)[u] - \sum_{l=0}^{i+1} \frac{D^{k-(i+1)+l}f(p)[u\otimes(q-p)^{\otimes l}]}{l!}\right|\right|_V}{||u||_{U^{\otimes(k-(i+1))}}\ ||q-p||_U^{\alpha+(i+1)}}. \end{aligned} \tag{3.49}$$

We next use the fundamental theorem of calculus to rewrite the difference $D^{k-(i+1)}f(q)[u] - D^{k-(i+1)}f(p)[u]$, giving

$$\sup_{p,q\in E}\sup_{u\neq 0} \frac{\left|\left| \int_0^1 D^{k-i}f(p+t(q-p))[u\otimes(q-p)]\mathrm{d}t - \sum_{l=1}^{i+1} \frac{D^{k-(i+1)+l}f(p)[u\otimes(q-p)^{\otimes l}]}{l!}\right|\right|_V}{||u||_{U^{\otimes(k-(i+1))}}\ ||q-p||_U^{\alpha+(i+1)}}. \tag{3.50}$$

We then re-index the sum by replacing l with $l+1$,

$$\sup_{p,q\in E}\sup_{u\neq 0} \frac{\left|\left| \int_0^1 D^{k-i}f(p+t(q-p))[u\otimes(q-p)]\mathrm{d}t - \sum_{l=0}^{i} \frac{D^{k-i+l}f(p)[u\otimes(q-p)^{\otimes l+1}]}{(l+1)!}\right|\right|_V}{||u||_{U^{\otimes(k-(i+1))}}\ ||q-p||_U^{\alpha+(i+1)}}. \tag{3.51}$$

Letting $u' = u \otimes (q-p) \in U^{\otimes(k-i)}$, bringing the sum inside the integral, and using $\int_0^1 t^l\,dt = \frac{1}{l+1}$, we obtain

$$\begin{aligned}\sup_{p,q\in E}\sup_{u\neq 0}&\frac{\left|\left|\int_0^1 D^{k-i}f(p+t(q-p))[u'] - \sum_{l=0}^{i}\frac{1}{l!}D^{k-i+l}f(p)[u'\otimes(t(q-p))^{\otimes l}]\mathrm{d}t\right|\right|_V}{||u||_{U^{\otimes(k-(i+1))}}\,||q-p||_U^{\alpha+(i+1)}}\\ &\overset{(3.48)}{\leq}\sup_{p,q\in E}\frac{M\int_0^1||u'||_{U^{\otimes(k-i)}}\,||t(q-p)||_U^{\alpha+i}\mathrm{d}t}{||u||_{U^{\otimes(k-(i+1))}}\,||q-p||_U^{\alpha+(i+1)}}.\end{aligned} \tag{3.52}$$

Applying $||u'||_{U^{\otimes(k-i)}} \leq ||u||_{U^{\otimes(k-(i+1))}}\,||q-p||_U$ we deduce that

$$\begin{aligned}\sup_{p,q\in E}\frac{\left|\left|D^{k-(i+1)}f(q) - \sum_{l=0}^{i+1}\frac{D^{k-(i+1)+l}f(p)[(q-p)^{\otimes l}]}{l!}\right|\right|_{k-(i+1)}}{||q-p||_U^{\alpha+(i+1)}} &\leq \sup_{p,q\in E}\frac{M\int_0^1||t(q-p)||_U^{\alpha+i}\mathrm{d}t}{||q-p||_U^{\alpha+i}}\\ &\leq M\int_0^1 t^{\alpha+i}\mathrm{d}t\\ &= \frac{M}{\alpha+(i+1)} \leq M.\end{aligned} \tag{3.53}$$

The case $i=0$ is true by the assumption that $[D^k f]_\alpha = M$, completing the induction. □

With this Taylor remainder estimate, we can now show that on open convex sets the classical $C_b^{k,\alpha}$ notion of regularity agrees exactly with $\mathrm{Lip}(\gamma)$ regularity.

Lemma 3.18. *Let U and V be Banach spaces, $E \subseteq U$ be open and convex, $k \in \mathbb{N}_0$, $\alpha \in (0,1]$, and $\gamma = k+\alpha$. Then $C_b^{k,\alpha}(E,V) \equiv \mathrm{Lip}(\gamma,E,V)$.*

Proof. **Part 1, $C_b^{k,\alpha}$ implies $\mathrm{Lip}(\gamma)$:** Let $f \in C_b^{k,\alpha}(E,V)$, and choose $\hat{f}^0 = f$ and $\hat{f}^i = D^i f$ for $i = 1,\dots,k$. Then

$$\|\hat{f}\|_{\mathrm{Lip}(\gamma)} = \max\left\{\max_{0\leq j\leq k}\sup_{p\in E}||\hat{f}^j(p)||_j, \max_{0\leq j\leq k}\sup_{p,q\in E}\frac{\|\hat{f}^j(q) - \sum_{l=0}^{k-j}\frac{\hat{f}^{j+l}(p)[(q-p)^{\otimes l}]}{l!}\|_j}{||q-p||_U^{\gamma-j}}\right\}, \tag{3.54}$$

where

$$||\cdot||_j = \begin{cases}||\cdot||_V, & j=0,\\ ||\cdot||_{\mathrm{op}}, & j=1,\dots,k.\end{cases} \tag{3.55}$$

Applying Lemma 3.17, this simplifies to

$$\begin{aligned}\|\hat{f}\|_{\mathrm{Lip}(\gamma)} &= \max\left\{\max_{0\leq j\leq k}\sup_{p\in E}||\hat{f}^j(p)||_j, \sup_{p,q\in E}\frac{\|\hat{f}^k(q)-\hat{f}^k(p)\|_k}{||q-p||_U^\alpha}\right\},\\ &= \max\left\{\max_{0\leq j\leq k}\sup_{p\in E}\|D^j f(p)\|_j, [D^k f]_\alpha\right\},\\ &= \|f\|_{C^{k,\alpha}},\end{aligned} \tag{3.56}$$

which is finite by assumption. Therefore, $\hat{f} \in \text{Lip}(\gamma, E, V)$ and $\|\hat{f}\|_{\text{Lip}(\gamma)} = \|f\|_{C^{k,\alpha}}$.

Part 2, Lip(γ) implies $C_b^{k,\alpha}$: Let $\hat{f} \in \text{Lip}(\gamma, E, V)$, $v \in U$ be fixed and $q = p + tv$ for $t \in [0, T]$, where T is small enough that $q \in E$ for all t. From (3.41) with $j = 0$,

$$\left\| \hat{f}^0(p+tv) - \sum_{l=0}^{k} \frac{\hat{f}^l(p)[(tv)^{\otimes l}]}{l!} \right\|_V = \|R_k(p, tv)\| \leq M\|tv\|^\gamma, \tag{3.57}$$

for all $p \in E$. Since

$$\lim_{t\to 0} \frac{\|R_k(p, tv)\|}{\|tv\|^k} \leq \lim_{t\to 0} M\|tv\|^{\gamma-k} = 0 \tag{3.58}$$

uniformly in p, then $\hat{f}^0$ is C^k with $\hat{f}^i = D^i\hat{f}^0$ for $i = 1, \ldots, k$ by the converse of Taylor's Theorem [Albrecht et al. 1971]. From (3.41) with $j = k$,

$$\sup_{p,q\in E} \frac{\left\| D^k\hat{f}^0(q) - D^k\hat{f}^0(p) \right\|_k}{||q-p||_U^{\gamma-k}} = [D^k\hat{f}^0]_{\gamma-k} \leq M. \tag{3.59}$$

Therefore, $\hat{f}^0 \in C_b^{k,\alpha}(E, V)$ with $\alpha = \gamma - k$. Applying the same argument as in part 1, $\|\hat{f}^0\|_{C^{k,\alpha}} = \|\hat{f}\|_{\text{Lip}(\gamma)}$. □

Although $\text{Lip}(\gamma, E, V)$ and $C_b^{k,\alpha}(E, V)$ are equivalent on open convex subsets E, and therefore the entire space U, they can differ on generic open sets E. Taking our example from before, where $E = (-\infty, 0) \cup (0, \infty)$ and $f : E \to \mathbb{R}$ is defined by

$$f(x) = \begin{cases} x - 1, & x \in (-\infty, 0), \\ x + 1, & x \in (0, \infty), \end{cases} \tag{3.60}$$

then $f \in C^{1,1}(E, \mathbb{R})$, but $f \notin \text{Lip}(2, E, \mathbb{R})$, as (3.41) diverges for $j = 0$ as x approaches 0. When E is an open subset, the core difference between Lip(γ) and $C_b^{k,\alpha}$ is requiring a Hölder type bound on each level of the derivative, as opposed to only the highest derivative. The additional regularity given by assuming your function is Lip(γ) is sufficient to ensure nesting.

Lemma 3.19 (Lip(γ) function nesting [Boutaib 2016; Lyons et al. 2025])**.** *Let U and V be Banach spaces, $E \subset U$, and $f \in \text{Lip}(\gamma, E, V)$, and $0 < \theta < \gamma$. Then $f \in \text{Lip}(\theta, E, V)$ and*

$$\|f\|_{\text{Lip}(\theta)} \leq (1 + e)\|f\|_{\text{Lip}(\gamma)}. \tag{3.61}$$

The nesting property is a key advantage of Lip(γ) regularity over $C_b^{k,\alpha}$ regularity on general domains, since it allows the Lie bracket of two Lip(γ) vector fields to be defined globally as a Lip($\gamma - 1$) vector field.

3.3.3 Lie Bracket

To define the Lie bracket, we first introduce the derivative of a $\mathrm{Lip}(\gamma)$ function and the composition $g[f] : E \to W$ of $f \in \mathrm{Lip}(\gamma_f, E, V)$ and $g \in \mathrm{Lip}(\gamma_g, E, \mathbf{L}(V, W))$. These are the basic ingredients from which the Lie bracket will be constructed.

Definition 3.20 ($\mathrm{Lip}(\gamma)$ Derivative)**.** *Let $f = (f^0, \ldots, f^k) \in \mathrm{Lip}(\gamma, E, V)$ with $\gamma > 1$. The derivative of f is defined by*

$$Df = ((Df)^0, (Df)^1, \ldots, (Df)^{k-1}) = (f^1, f^2, \ldots, f^k). \tag{3.62}$$

Lemma 3.21. *If $f \in \mathrm{Lip}(\gamma, E, V)$ with $\gamma > 1$ and $\|f\|_{\mathrm{Lip}(\gamma)} = M_f$, then $Df \in \mathrm{Lip}(\gamma - 1, E, \mathbf{L}(U, V))$ with $\|Df\|_{\mathrm{Lip}(\gamma-1)} \le M_f$.*

Proof. Trivially, Df satisfies (3.40) with bound M_f. Under the currying convention $\mathbf{L}_{\mathrm{sym}}\big(U^{\otimes j}, \mathbf{L}(U, V)\big) \cong \mathbf{L}\big(U^{\otimes(j+1)}, V\big)$, we have

$$(Df)^j = f^{j+1} \tag{3.63}$$

for $0 \le j \le k-1$. Hence, for all $x, y \in E$,

$$\begin{aligned} \left\| (Df)^j(y) - \sum_{l=0}^{k-1-j} \frac{(Df)^{j+l}(x)[(y-x)^{\otimes l}]}{l!} \right\|_j & \\ = \left\| f^{j+1}(y) - \sum_{l=0}^{k-(j+1)} \frac{f^{j+1+l}(x)[(y-x)^{\otimes l}]}{l!} \right\|_{j+1} & \\ \le M_f \|x - y\|_U^{\gamma-1-j}. & \end{aligned} \tag{3.64}$$

Therefore, Df satisfies (3.41) with bound M_f and $Df = (f^1, \ldots, f^k) \in \mathrm{Lip}(\gamma - 1, E, \mathbf{L}(U, V))$. □

Definition 3.22. *Let U, V, W be Banach spaces, $E \subset U$, $f \in \mathrm{Lip}(\gamma_f, E, V)$, $g \in \mathrm{Lip}(\gamma_g, E, \mathbf{L}(V, W))$, $\gamma = \min(\gamma_f, \gamma_g)$, and k be the integer such that $k < \gamma \le k + 1$. Then*

$$h = (h^0, \ldots, h^k) = g[f] \tag{3.65}$$

is defined pointwise by

$$h^i(p)[u_1 \otimes \cdots \otimes u_i] = \sum_{r=0}^{i} \frac{1}{r!(i-r)!} \sum_{\sigma \in S_i} g^r(p)[u_{\sigma(1)} \otimes \cdots \otimes u_{\sigma(r)}]\big[f^{i-r}(p)[u_{\sigma(r+1)} \otimes \cdots \otimes u_{\sigma(i)}]\big] \tag{3.66}$$

for $p \in E$ and $i = 0, \ldots, k$.

This definition is an application of the definition from [Boutaib 2016, Proposition 3.29] and is analogous to applying Faà-di-Bruno to a bilinear map applied to two $C^{k,\alpha}$ functions [Faà di Bruno 1855].

Lemma 3.23. *If $f \in \mathrm{Lip}(\gamma_f, E, V)$ and $g \in \mathrm{Lip}(\gamma_g, E, \mathbf{L}(V,W))$ then $h = g[f] \in \mathrm{Lip}(\gamma, E, W)$, where $\gamma = \min(\gamma_f, \gamma_g)$.*

Proof. By Lemma 3.19, $f \in \mathrm{Lip}(\gamma_f)$ and $g \in \mathrm{Lip}(\gamma_g)$ implies $f, g \in \mathrm{Lip}(\gamma)$ where $\|f\|_{\mathrm{Lip}(\gamma)} \leq (1+e)\|f\|_{\mathrm{Lip}(\gamma_f)}$ and $\|g\|_{\mathrm{Lip}(\gamma)} \leq (1+e)\|g\|_{\mathrm{Lip}(\gamma_g)}$, with at least one bound holding with equality. Let $B : \mathbf{L}(V,W) \times V \to W$ be defined by

$$B(A, v) = A[v], \tag{3.67}$$

for $A \in \mathbf{L}(V,W)$ and $v \in V$. By [Boutaib 2016, Proposition 3.29], the function $g[f] = B(g,f) = (B^0, \ldots, B^k) = (h^0, \ldots, h^k)$, where the h^i are defined by (3.66), satisfies

$$\|B(g,f)\|_{\mathrm{Lip}(\gamma)} \leq C_\gamma \|g\|_{\mathrm{Lip}(\gamma)} \|f\|_{\mathrm{Lip}(\gamma)} \leq C_\gamma (1+e)\|g\|_{\mathrm{Lip}(\gamma_g)} \|f\|_{\mathrm{Lip}(\gamma_f)}, \tag{3.68}$$

where C_γ is a constant depending only on γ. Therefore, $h = g[f] \in \mathrm{Lip}(\gamma, E, W)$. □

Definition 3.24 (Lip(γ) Lie Bracket)**.** *Let $f = (f^0, \ldots, f^{k_f}) \in \mathrm{Lip}(\gamma_f, E, U)$ and $g = (g^0, \ldots, g^{k_g}) \in \mathrm{Lip}(\gamma_g, E, U)$ with $\gamma_f, \gamma_g > 1$. Their Lie bracket is defined by*

$$[f, g] = Dg[f] - Df[g]. \tag{3.69}$$

Lemma 3.25. *If $f = (f^0, \ldots, f^{k_f}) \in \mathrm{Lip}(\gamma_f, E, U)$ and $g = (g^0, \ldots, g^{k_g}) \in \mathrm{Lip}(\gamma_g, E, U)$ for $\gamma_f, \gamma_g > 1$, then*

$$[f, g] \in \mathrm{Lip}(\min(\gamma_f, \gamma_g) - 1, E, U). \tag{3.70}$$

Proof. Let $\gamma = \min(\gamma_f, \gamma_g)$. By Lemma 3.19, $f, g \in \mathrm{Lip}(\gamma, E, U)$. Then Lemma 3.21 gives

$$Df, Dg \in \mathrm{Lip}(\gamma - 1, E, \mathbf{L}(U,U)). \tag{3.71}$$

Applying Lemma 3.23, we obtain

$$Dg[f], Df[g] \in \mathrm{Lip}(\gamma - 1, E, U). \tag{3.72}$$

Therefore,

$$[f, g] = Dg[f] - Df[g] \in \mathrm{Lip}(\gamma - 1, E, U). \tag{3.73}$$

□

Remark 3.26. *The proof of Lemma 3.23 relies on Lemma 3.19, the nesting property of* $\mathrm{Lip}(\gamma)$ *functions. This is the crucial difference between* $\mathrm{Lip}(\gamma)$ *functions and* $C_b^{k,\alpha}$ *functions that allows for a well defined global Lie bracket of* $\mathrm{Lip}(\gamma)$ *functions.*

The polynomial view point of $\mathrm{Lip}(\gamma)$ functions, Definition 3.16, motivates an alternative definition of the Lie bracket of two $\mathrm{Lip}(\gamma)$ functions.

Definition 3.27 ($\mathrm{Lip}(\gamma)$ Polynomial Lie Bracket)**.** *Let* $f = (f^0, \dots, f^k) \in \mathrm{Lip}(\gamma, E, U)$ *and* $g = (g^0, \dots, g^k) \in \mathrm{Lip}(\gamma, E, U)$ *with* $\gamma > 1$*. Let*

$$P_p^f(q) = \sum_{l=0}^{k} \frac{f^l(p)[(q-p)^{\otimes l}]}{l!}. \tag{3.74}$$

and

$$P_p^g(q) = \sum_{l=0}^{k} \frac{g^l(p)[(q-p)^{\otimes l}]}{l!} \tag{3.75}$$

for $p \in E$ *and* $q \in U$*. The Polynomial Lie bracket of* f *and* g *is defined pointwise by*

$$[P_p^f, P_p^g](q) = \sum_{l=0}^{k-1} \frac{g^{l+1}(p)\left[P_p^f(q) \otimes (q-p)^{\otimes l}\right] - f^{l+1}(p)\left[P_p^g(q) \otimes (q-p)^{\otimes l}\right]}{l!}. \tag{3.76}$$

for $p \in E$ *and* $q \in U$*.*

Remark 3.28. *Definition 3.27 relies on the pointwise Lie bracket of two polynomial vector fields. Since polynomial vector fields are smooth and closed under this bracket, they form a Lie subalgebra of the space of smooth vector fields.*

Initially, the two definitions seem to produce different Lie brackets. In particular, the polynomial at each point in E when using Definition 3.24 is of order $k-1$, whereas the polynomial at each point when using Definition 3.27 is of order up to $2k-1$. However, the two Lie brackets agree with each other in the $\mathrm{Lip}(\gamma - 1)$ sense.

Lemma 3.29. *Let* $f = (f^0, \dots, f^k) \in \mathrm{Lip}(\gamma, E, U)$ *and* $g = (g^0, \dots, g^k) \in \mathrm{Lip}(\gamma, E, U)$ *with* $\gamma > 1$ *and* k *the non-negative integer such that* $\gamma \in (k, k+1]$*. For each* $p \in E$*, the polynomial defined at each point by their Lie Bracket* $P_p^{[f,g]} \in \mathbf{P}^{k-1}(U, U)$ *and the polynomial Lie bracket* $[P_p^f, P_p^g]$ *agree up to the* $(k-1)^{th}$ *term.*

Proof. Let $s \leq k$. Then

$$D_q^s P_p^f(q)\Big|_{q=p} = f^s(p), \tag{3.77}$$

and

$$D_q^s P_p^g(q)\Big|_{q=p} = g^s(p). \tag{3.78}$$

Additionally,

$$D_q^s(q-p)^{\otimes l}\Big|_{q=p}[u_1\otimes\cdots\otimes u_s]=\begin{cases}\sum_{\sigma\in S_l}u_{\sigma(1)}\otimes\cdots\otimes u_{\sigma(l)}, & s=l,\\ 0, & \text{otherwise.}\end{cases}\tag{3.79}$$

Letting

$$F_g^l(q)=\frac{1}{l!}g^{l+1}(p)\left[P_p^f(q)\otimes(q-p)^{\otimes l}\right],\tag{3.80}$$

and

$$\bigotimes_{\alpha=1}^{m}u_\alpha=u_1\otimes\cdots\otimes u_m,\tag{3.81}$$

then for $l\le m\le k-1$, we first apply the product rule to obtain

$$\begin{aligned}&D_q^mF_g^l(q)\Big|_{q=p}\left[\bigotimes_{\alpha=1}^{m}u_\alpha\right]\\&=\frac{1}{l!}\sum_{i=0}^{m}\frac{1}{i!(m-i)!}\sum_{\sigma\in S_m}g^{l+1}(p)\left(D_q^iP_p^f(q)\left[\bigotimes_{\alpha=1}^{i}u_{\sigma(\alpha)}\right]\otimes D_q^{m-i}(q-p)^{\otimes l}\left[\bigotimes_{\alpha=i+1}^{m}u_{\sigma(\alpha)}\right]\right).\end{aligned}\tag{3.82}$$

At $q=p$, the term $D_q^{m-i}(q-p)^{\otimes l}$ vanishes unless $m-i=l$. Therefore only the term $i=m-l$ remains, and so

$$\begin{aligned}&D_q^mF_g^l(q)\Big|_{q=p}\left[\bigotimes_{\alpha=1}^{m}u_\alpha\right]\\&=\frac{1}{(m-l)!(l!)^2}\sum_{\sigma\in S_m}g^{l+1}(p)\left(D_q^{m-l}P_p^f(q)\left[\bigotimes_{\alpha=1}^{m-l}u_{\sigma(\alpha)}\right]\otimes D_q^l(q-p)^{\otimes l}\left[\bigotimes_{\alpha=m-l+1}^{m}u_{\sigma(\alpha)}\right]\right).\end{aligned}\tag{3.83}$$

We now evaluate both derivatives at $q=p$. Using $D_q^{m-l}P_p^f(q)|_{q=p}=f^{m-l}(p)$ and the formula above for $D_q^l(q-p)^{\otimes l}|_{q=p}$, we obtain

$$\begin{aligned}&D_q^mF_g^l(q)\Big|_{q=p}\left[\bigotimes_{\alpha=1}^{m}u_\alpha\right]\\&=\frac{1}{(m-l)!(l!)^2}\sum_{\sigma\in S_m}\sum_{\tau\in S_l}g^{l+1}(p)\left(f^{m-l}(p)\left[\bigotimes_{\alpha=1}^{m-l}u_{\sigma(\alpha)}\right]\otimes\bigotimes_{\beta=1}^{l}u_{\sigma(m-l+\tau(\beta))}\right).\end{aligned}\tag{3.84}$$

Finally, the sum over $\tau\in S_l$ contributes a factor of $l!$, and hence

$$\begin{aligned}&D_q^mF_g^l(q)\Big|_{q=p}\left[\bigotimes_{\alpha=1}^{m}u_\alpha\right]\\&=\frac{1}{(m-l)!l!}\sum_{\sigma\in S_m}g^{l+1}(p)\left(f^{m-l}(p)\left[\bigotimes_{\alpha=1}^{m-l}u_{\sigma(\alpha)}\right]\otimes\bigotimes_{\alpha=m-l+1}^{m}u_{\sigma(\alpha)}\right).\end{aligned}\tag{3.85}$$

Similarly, for

$$F_f^l(q) = \frac{1}{l!} f^{l+1}(p) \left[P_p^g(q) \otimes (q-p)^{\otimes l} \right], \tag{3.86}$$

the same argument gives

$$\begin{aligned} &D_q^m F_f^l(q)\Big|_{q=p} \left[\bigotimes_{\alpha=1}^{m} u_\alpha \right] \\ &\quad = \frac{1}{(m-l)!l!} \sum_{\sigma \in S_m} f^{l+1}(p) \left(g^{m-l}(p) \left[\bigotimes_{\alpha=1}^{m-l} u_{\sigma(\alpha)} \right] \otimes \bigotimes_{\alpha=m-l+1}^{m} u_{\sigma(\alpha)} \right). \end{aligned} \tag{3.87}$$

Since $D_q^m F_g^l(q)\Big|_{q=p} = 0$ and $D_q^m F_f^l(q)\Big|_{q=p} = 0$ for $l > m$,

$$D_q^m [P_p^f, P_p^g](q)\Big|_{q=p} = \sum_{l=0}^{m} \left(D_q^m F_g^l(q)\Big|_{q=p} - D_q^m F_f^l(q)\Big|_{q=p} \right), \tag{3.88}$$

for $0 \le m \le k-1$. Therefore,

$$D_q^m [P_p^f, P_p^g](q)\Big|_{q=p} = [f,g]^m(p), \tag{3.89}$$

for $0 \le m \le k-1$, where $[f,g]^m(p)$ is obtained by combining Definition 3.22 and Definition 3.24. By the definition of $P_p^{[f,g]}$, we also have

$$D_q^m P_p^{[f,g]}(q)\Big|_{q=p} = [f,g]^m(p), \tag{3.90}$$

for $0 \le m \le k-1$. Hence $P_p^{[f,g]}$ and $[P_p^f, P_p^g]$ agree up to the $(k-1)^{\text{th}}$ term. □

Trivially, both the $\mathrm{Lip}(\gamma)$ Lie bracket and the $\mathrm{Lip}(\gamma)$ polynomial Lie bracket satisfy bilinearity and anti-symmetry.

Theorem 3.30. *Let $\gamma > 2$. Both the $\mathrm{Lip}(\gamma)$ Lie bracket and the $\mathrm{Lip}(\gamma)$ polynomial Lie bracket satisfy the Jacobi identity in $\mathrm{Lip}(\gamma - 2)$.*

Proof. Let U be a Banach space, $E \subset U$, $\gamma > 2$, $f, g, h \in \mathrm{Lip}(\gamma, E, U)$, k be the non-negative integer such that $\gamma \in (k, k+1]$, and for $p \in E$, let $P_p^f, P_p^g, P_p^h \in \mathbf{P}^k(U,U)$ be the polynomial corresponding to f, g, and h at point p, respectively. As noted in Remark 3.28, Polynomial vector fields form a Lie subalgebra. Therefore for all $p \in E$,

$$\left[P_p^f, [P_p^g, P_p^h]\right](q) + \left[P_p^g, [P_p^h, P_p^f]\right](q) + \left[P_p^h, [P_p^f, P_p^g]\right](q) = 0 \tag{3.91}$$

and the polynomial Lie bracket satisfies the Jacobi identity. Considering $[g,h]$, Lemma 3.29 means that $P_p^{[g,h]}$ and $[P_p^g, P_p^h]$ are identical in the first $k-1$ terms for all p. Since

the first $k-2$ terms of $\left[P_p^f, [P_p^g, P_p^h]\right]$ only depend on the first $k-1$ terms of $[P_p^g, P_p^h]$, Lemma 3.29 also implies that $P_p^{[f,[g,h]]}$ and $\left[P_p^f, [P_p^g, P_p^h]\right]$ are identical in the first $k-2$ terms for all p. Therefore, (3.91) implies that the first $k-2$ terms of

$$P_p^{[f,[g,h]]}(q) + P_p^{[g,[h,f]]}(q) + P_p^{[h,[f,g]]}(q) \tag{3.92}$$

are equal to 0 for all p. Furthermore, Lemma 3.25 implies that $[f,[g,h]] \in \mathrm{Lip}(\gamma - 2, E, U)$ and similarly for the other permutations. Therefore,

$$[f,[g,h]] + [g,[h,f]] + [h,[f,g]] = 0 \tag{3.93}$$

in $\mathrm{Lip}(\gamma-2, E, U)$ and the $\mathrm{Lip}(\gamma)$ Lie bracket satisfies the Jacobi identity. □

3.3.4 The Extension Theorem

Whitney proved necessary and sufficient conditions for derivative data $(f^0, \ldots, f^k)$ on a closed set $E \subset \mathbb{R}^n$ to admit a C^k extension to $\mathbb{R}^n$ [Whitney 1934]. Stein later recast the problem using $\mathrm{Lip}(\gamma)$ functions and constructed a bounded linear extension operator with norm controlled independently of E [Stein 1970]. We refer to the general result as the Stein–Whitney extension theorem.

Theorem 3.31 (Stein-Whitney Extension Theorem [Whitney 1934; Stein 1970])**.** *Let U and V be Banach spaces, U be finite dimensional, $E \subset U$ be closed, and $f \in \mathrm{Lip}(\gamma, E, V)$. Then there exists a function $\hat{f} \in \mathrm{Lip}(\gamma, U, V)$ such that*

$$\hat{f}(p) = f(p), \quad p \in E, \tag{3.94}$$

and

$$||\hat{f}||_{\mathrm{Lip}(\gamma,U,V)} \leq C||f||_{\mathrm{Lip}(\gamma,E,V)}, \tag{3.95}$$

for some constant C independent of f and E.

Proof. Stein proves this theorem by construction for the case $U = \mathbb{R}^n$ and $V = \mathbb{R}$ [Stein 1970]. Given the equivalence of Banach norms on $\mathbb{R}^n$, U can be replaced by any finite dimensional Banach space. Furthermore, as noted in [Baldi et al. 2018, Appendix B], Stein's proof holds exactly for any Banach space V and $\mathrm{Lip}(\gamma)$ functions as given in Definition 3.15. □

Despite $\mathrm{Lip}(\gamma)$ functions being well defined for infinite dimensional U, Theorem 3.31 is in general not true, as shown by the counter example of Wells [1973]. Fefferman has extended the work of Whitney by proving necessary and sufficient conditions for the existence of a C^k extension to a function with only its value specified on E [Fefferman 2006].

3.4 Composition of $\mathrm{Lip}(\gamma)$ Functions

3.4.1 Introduction

In Chapter 4, the Log-ODE method will be applied to a CDE where the vector field is parametrised by a neural network. A non-trivial application of the Log-ODE method requires the neural network vector field to be $\mathrm{Lip}(\gamma)$ for $\gamma > 1$. A common strategy for establishing regularity of neural networks is to verify that each layer of the network satisfies an appropriate regularity condition and then invoke a composition theorem. This section develops the mathematical foundations required to apply that argument for $\mathrm{Lip}(\gamma)$ functions.

Definition 3.32 (Partition of a Set)**.** *Let $\mathcal{P}(n)$ be the set of all partitions $\pi = (\pi_1, \dots, \pi_{|\pi|})$ of the set $\{1, \dots, n\}$, where $|\pi|$ denotes the number of parts in the partition π, and $|\pi_i|$ the number of elements in the i^{th} part of the partition.*

Definition 3.33 (Composition of $\mathrm{Lip}(\gamma)$ Functions [Cass et al. 2012])**.** *Let U, V, and W be Banach spaces, $E \subset U$ and $F \subset V$. The composition of $f \in \mathrm{Lip}(\gamma, E, F)$ and $g \in \mathrm{Lip}(\gamma, F, W)$, denoted $h = (h^0, h^1, \dots, h^k) \in \mathrm{Lip}(\gamma, E, W)$, is defined by*

$$\begin{aligned} h^0(p) &= g^0\big(f^0(p)\big), \\ h^n(p)[u_1 \otimes \cdots \otimes u_n] &= \sum_{\pi \in \mathcal{P}(n)} g^{|\pi|}\big(f^0(p)\big) \left[f^{|\pi_1|}(p)[u_{\pi_1}] \otimes \cdots \otimes f^{|\pi_{|\pi|}|}(p)[u_{\pi_{|\pi|}}] \right], \end{aligned} \tag{3.96}$$

where u_{π_i} is the tensor product of the vectors indexed by π_i,

$$u_{\pi_i} = \bigotimes_{j \in \pi_i} u_j. \tag{3.97}$$

When $\gamma < 1$, the composition h in Definition 3.33 is not necessarily a $\mathrm{Lip}(\gamma)$ function. For example, $f(x) = x^\gamma$ and $g(x) = x^\gamma$ are both $\gamma-$Hölder continuous on $[0, 1]$, whereas $h(x) = (g \circ f)(x) = x^{\gamma^2}$ is not. When $\gamma = 1$, then $h \in \mathrm{Lip}(1, E, W)$ and we have the standard Lipschitz norm bound

$$||h||_{\mathrm{Lip}(1,E,W)} \leq ||g||_{\mathrm{Lip}(1,F,W)} \max\left(||f||_{\mathrm{Lip}(1,E,F)}, 1\right). \tag{3.98}$$

Lemma 3.34 (Composed $\mathrm{Lip}(\gamma)-$norm [Cass et al. 2012; Boutaib 2016])**.** *For $\gamma \geq 1$, the composition h defined in Definition 3.33 satisfies*

$$||h||_{\mathrm{Lip}(\gamma,E,W)} \leq C_\gamma ||g||_{\mathrm{Lip}(\gamma,F,W)} \max\left(||f||^\gamma_{\mathrm{Lip}(\gamma,E,F)}, 1\right), \tag{3.99}$$

where C_γ is a constant independent of f and g.

Proof. Explicit calculation can be used to verify that if f and g are $\text{Lip}(\gamma)$, Definition 3.15 implies h is $\text{Lip}(\gamma)$ with $||h||_{\text{Lip}(\gamma)}$ obeying (3.99) [Cass et al. 2012]. □

The original statement of Lemma 3.34 in [Cass et al. 2012] gives (3.99) as

$$||g \circ f||_{\text{Lip}(\gamma)} \leq C_\gamma ||g||_{\text{Lip}(\gamma)} \max\left\{||f||^k_{\text{Lip}(\gamma)}, 1\right\}. \tag{3.100}$$

We believe this is a small erratum, with the intended power being γ, as for $g : [0,1] \to [0,1]$ defined by $g(x) = x$, (3.100) implies there exists $C_1 > 0$ such that

$$||g \circ f||_{\text{Lip}(1)} = ||f||_{\text{Lip}(1)} \leq C_1 ||g||_{\text{Lip}(1)} = C_1 \tag{3.101}$$

for all bounded and Lipschitz $f : [0,1] \to [0,1]$. As a counterexample, for any $C_1 > 0$, take $f(x) = x^n$ with $n > \max\{C_1, 1\}$. Boutaib [2016] gives an alternative proof of Lemma 3.34 with the bound $||f||^\gamma_{\text{Lip}(\gamma,E,F)}$.

Accurately bounding the $\text{Lip}(\gamma)-$norm of a neural network requires bounds on C_γ in (3.99). This can be obtained via the explicit calculations mentioned in the proof of Lemma 3.34, and these have been completed for the case $\gamma \in (1,2]$.

3.4.2 The Case $1 < \gamma \leq 2$

For $1 < \gamma \leq 2$, a $\text{Lip}(\gamma)$ function has only two components, which makes it feasible to track the relevant quantities explicitly and obtain a concrete bound for C_γ.

Lemma 3.35. *Let U, V, and W be Banach spaces and $E \subset U$ and $F \subset V$. For $\gamma \in (1,2]$, let $f = (f^0, f^1) \in \text{Lip}(\gamma, E, F)$ and $g = (g^0, g^1) \in \text{Lip}(\gamma, F, W)$. Consider $h^0 : E \to W$ and $h^1 : E \to \mathbf{L}(U, W)$ defined for $p \in E$ and $u \in U$ by*

$$h^0(p) := g^0\left(f^0(p)\right) \qquad \text{and} \qquad h^1(p)[u] := g^1\left(f^0(p)\right)\left[f^1(p)[u]\right]. \tag{3.102}$$

Then $h := (h^0, h^1) \in \text{Lip}(\gamma, E, W)$ and

$$||h||_{\text{Lip}(\gamma,E,W)} \leq (1 + 2^\gamma)\, ||g||_{\text{Lip}(\gamma,F,W)} \max\left\{1, ||f||^\gamma_{\text{Lip}(\gamma,E,F)}\right\}. \tag{3.103}$$

Proof. To prove that $h = (h^0, h^1) \in \text{Lip}(\gamma, E, W)$, we must bound the pointwise terms h^0 and h^1, together with the corresponding remainder terms from Definition 3.15. We proceed in four steps. First, we record the bounds for f and g that follow directly from Definition 3.15. Next, we prove the pointwise bounds for h^0 and h^1. We then estimate the remainder terms R_0^h and R_1^h separately in the cases $\|q - p\|_U > 1$ and

$\|q-p\|_U \leq 1$. Finally, we combine the estimates to obtain the norm bound claimed in (3.103).

We begin by establishing the bounds on $f^0 : E \to F$, $f^1 : E \to \mathbf{L}(U,V)$, $g^0 : F \to W$ and $g^1 : F \to \mathbf{L}(V,W)$ that arise directly from Definition 3.15. Letting $M_f = ||f||_{\mathrm{Lip}(\gamma,E,F)}$, for all $p \in E$

$$\textbf{(I)} \quad \big|\big|f^0(p)\big|\big|_V \leq M_f \qquad \text{and} \quad \textbf{(II)} \quad \big|\big|f^1(p)\big|\big|_{\mathbf{L}(U,V)} \leq M_f. \tag{3.104}$$

Similarly, letting $M_g = ||g||_{\mathrm{Lip}(\gamma,F,W)}$, for all $x \in F$ we have that

$$\textbf{(I)} \quad \big|\big|g^0(x)\big|\big|_W \leq M_g \qquad \text{and} \quad \textbf{(II)} \quad \big|\big|g^1(x)\big|\big|_{\mathbf{L}(V,W)} \leq M_g. \tag{3.105}$$

Define $R_0^f : E \times E \to V$ and $R_1^f : E \times E \to \mathbf{L}(U,V)$ by

$$\begin{aligned} R_0^f(p,q) &:= f^0(q) - f^0(p) - f^1(p)[q-p], \\ R_1^f(p,q)[u] &:= f^1(q)[u] - f^1(p)[u], \end{aligned} \tag{3.106}$$

for any $p, q \in E$ and $u \in U$. Then

$$\begin{aligned} &\textbf{(I)} \quad \Big|\Big|R_0^f(p,q)\Big|\Big|_V \leq M_f||q-p||_U^\gamma, \\ &\textbf{(II)} \quad \Big|\Big|R_1^f(p,q)\Big|\Big|_{\mathbf{L}(U,V)} \leq M_f||q-p||_U^{\gamma-1}. \end{aligned} \tag{3.107}$$

Similarly, define $R_0^g : F \times F \to W$ and $R_1^g : F \times F \to \mathbf{L}(V,W)$ by

$$\begin{aligned} R_0^g(x,y) &:= g^0(y) - g^0(x) - g^1(x)[y-x], \\ R_1^g(x,y)[v] &:= g^1(y)[v] - g^1(x)[v], \end{aligned} \tag{3.108}$$

for $x, y \in F$ and $v \in V$. Then,

$$\begin{aligned} &\textbf{(I)} \quad ||R_0^g(x,y)||_W \leq M_g||y-x||_V^\gamma, \\ &\textbf{(II)} \quad ||R_1^g(x,y)||_{\mathbf{L}(V,W)} \leq M_g||y-x||_V^{\gamma-1}. \end{aligned} \tag{3.109}$$

Now define $h^0 : E \to W$ and $h^1 : E \to \mathbf{L}(U,W)$ as in (3.102),

$$h^0(p) := g^0\left(f^0(p)\right) \qquad \text{and} \qquad h^1(p)[u] := g^1\left(f^0(p)\right)\left[f^1(p)[u]\right], \tag{3.110}$$

for $p \in E$ and $u \in U$. Then define the corresponding remainder terms $R_0^h : E \times E \to W$ and $R_1^h : E \times E \to \mathbf{L}(U,W)$ by

$$\begin{aligned} R_0^h(p,q) &:= h^0(q) - h^0(p) - h^1(p)[q-p], \\ R_1^h(p,q)[u] &:= h^1(q)[u] - h^1(p)[u], \end{aligned} \tag{3.111}$$

for $p, q \in E$ and $u \in U$. We now establish that $h = (h^0, h^1) \in \mathrm{Lip}(\gamma, E, W)$ and that the norm estimate claimed in (3.103) is satisfied.

First we consider the bounds on h^0 and h^1. For any $p \in E$, (**I**) in (3.105) implies that

$$\left|\left|h^0(p)\right|\right|_W = \left|\left|g^0\left(f^0(p)\right)\right|\right|_W \leq M_g \tag{3.112}$$

since $f^0(p) \in F$. Further, for any $p \in E$ and any $u \in U$, (3.105) and (**II**) in (3.104) imply that

$$\begin{aligned}\left|\left|h^1(p)[u]\right|\right|_W &= \left|\left|g^1\left(f^0(p)\right)\left[f^1(p)[u]\right]\right|\right|_W \\ &\leq \left|\left|g^1\left(f^0(p)\right)\right|\right|_{\mathbf{L}(V,W)} \left|\left|f^1(p)\right|\right|_{\mathbf{L}(U,V)} ||u||_U \\ &\leq M_g M_f ||u||_U\end{aligned}$$

since $f^0(p) \in F$. Taking the supremum over $u \in U$ with unit U-norm, it follows that

$$\left|\left|h^1(p)\right|\right|_{\mathbf{L}(U,W)} \leq ||g||_{\mathrm{Lip}(\gamma,F,W)} ||f||_{\mathrm{Lip}(\gamma,E,F)}. \tag{3.113}$$

Now we consider the bounds on R_0^h and R_1^h. For this purpose we fix $p, q \in E$ and $u \in U$. We first assume that $||q-p||_U > 1$. In this case we may use (3.112) and (3.113) to compute that

$$\begin{aligned}\left|\left|R_0^h(p,q)\right|\right|_W &= \left|\left|h^0(q) - h^0(p) - h^1(p)[q-p]\right|\right|_W \\ &\leq 2||g||_{\mathrm{Lip}(\gamma,F,W)} + ||g||_{\mathrm{Lip}(\gamma,F,W)} ||f||_{\mathrm{Lip}(\gamma,E,F)} ||q-p||_U.\end{aligned}$$

Since $\gamma > 1$ means that $1 < ||q-p||_U < ||q-p||_U^\gamma$, we deduce that

$$\left|\left|R_0^h(p,q)\right|\right|_W \leq ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(2 + ||f||_{\mathrm{Lip}(\gamma,E,F)}\right) ||q-p||_U^\gamma. \tag{3.114}$$

Similarly, we may use (3.113) and $||q-p||_U^{\gamma-1} > 1$ to compute that

$$\left|\left|R_1^h(p,q)[u]\right|\right|_W = \left|\left|h^1(q)[u] - h^1(p)[u]\right|\right|_W \leq 2||g||_{\mathrm{Lip}(\gamma,F,W)} ||f||_{\mathrm{Lip}(\gamma,E,F)} ||q-p||_U^{\gamma-1} ||u||_U. \tag{3.115}$$

Taking the supremum over $u \in U$ with unit U-norm in (3.115) yields the estimate that

$$\left|\left|R_1^h(p,q)\right|\right|_{\mathbf{L}(U,W)} \leq 2||g||_{\mathrm{Lip}(\gamma,F,W)} ||f||_{\mathrm{Lip}(\gamma,E,F)} ||q-p||_U^{\gamma-1}. \tag{3.116}$$

Together, (3.114) and (3.116) establish the remainder term estimates required to conclude that $h = (h^0, h^1) \in \mathrm{Lip}(\gamma, E, W)$ in the case that $||q-p||_U > 1$.

We next establish similar remainder term estimates when $||q-p||_U < 1$. Thus we fix $p, q \in E$ and assume that $||q-p||_U < 1$. Note that $\gamma > 1$ means that $||q-p||_U^\gamma < ||q-p||_U < 1$. Additionally,

$$\begin{aligned} \left|\left|f^0(q) - f^0(p)\right|\right|_V &\overset{(3.106)}{=} \left|\left|f^1(p)[q-p] + R_0^f(p,q)\right|\right|_V, \\ &\leq ||f||_{\mathrm{Lip}(\gamma,E,F)} \left(||q-p||_U + ||q-p||_U^\gamma\right), \\ &\leq 2||f||_{\mathrm{Lip}(\gamma,E,F)}||q-p||_U, \end{aligned} \tag{3.117}$$

where (**II**) in (3.104) and (**I**) in (3.107) have been used. We now consider the term $R_0^h(p,q)$. We start by observing that

$$\begin{aligned} R_0^h(p,q) &\overset{(3.111)}{=} h^0(q) - h^0(p) - h^1(p)[q-p] \\ &\overset{(3.110)}{=} g^0\left(f^0(q)\right) - g^0\left(f^0(p)\right) - g^1\left(f^0(p)\right)\left[f^1(p)[q-p]\right] \\ &\overset{(3.108)}{=} g^1\left(f^0(p)\right)\left[f^0(q) - f^0(p) - f^1(p)[q-p]\right] + R_0^g\left(f^0(p), f^0(q)\right) \\ &\overset{(3.106)}{=} g^1\left(f^0(p)\right)\left[R_0^f(p,q)\right] + R_0^g\left(f^0(p), f^0(q)\right). \end{aligned}$$

Consequently, by using (**II**) in (3.105) to estimate the term $g^1\left(f^0(p)\right)$, (**I**) in (3.107) to estimate the term $R_0^f(p,q)$, and (**I**) in (3.109) to estimate the term $R_0^g\left(f^0(p), f^0(q)\right)$, we may deduce that

$$\left|\left|R_0^h(p,q)\right|\right|_W \leq ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(||f||_{\mathrm{Lip}(\gamma,E,F)}||q-p||_U^\gamma + \left|\left|f^0(q) - f^0(p)\right|\right|_V^\gamma\right). \tag{3.118}$$

The combination of (3.117) and (3.118) yields the estimate

$$\left|\left|R_0^h(p,q)\right|\right|_W \leq ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(||f||_{\mathrm{Lip}(\gamma,E,F)} + 2^\gamma ||f||_{\mathrm{Lip}(\gamma,E,F)}^\gamma\right) ||q-p||_U^\gamma. \tag{3.119}$$

Turning our attention to R_1^h, we fix $u \in U$ and compute that

$$\begin{aligned} R_1^h(p,q)[u] &\overset{(3.111)}{=} h^1(q)[u] - h^1(p)[u] \\ &\overset{(3.110)}{=} g^1\left(f^0(q)\right)\left[f^1(q)[u]\right] - g^1\left(f^0(p)\right)\left[f^1(p)[u]\right] \\ &\overset{(3.108)}{=} g^1\left(f^0(p)\right)\left[f^1(q)[u] - f^1(p)[u]\right] + R_1^g\left(f^0(p), f^0(q)\right)\left[f^1(q)[u]\right] \\ &\overset{(3.106)}{=} g^1\left(f^0(p)\right)\left[R_1^f(p,q)[u]\right] + R_1^g\left(f^0(p), f^0(q)\right)\left[f^1(q)[u]\right]. \end{aligned}$$

Consequently, by using (**II**) in (3.105) to estimate the term $g^1\left(f^0(p)\right)$, (**II**) in (3.104) to estimate the term $f^1(q)$, (**II**) in (3.107) to estimate the term $R_1^f(p,q)$, and (**II**) in (3.109) to estimate the term $R_1^g\left(f^0(p), f^0(q)\right)$, we may deduce that

$$\left|\left|R_1^h(p,q)[u]\right|\right|_W \leq ||g||_{\mathrm{Lip}(\gamma,F,W)}||f||_{\mathrm{Lip}(\gamma,E,F)} \left(||q-p||_U^{\gamma-1} + \left|\left|f^0(q) - f^0(p)\right|\right|_V^{\gamma-1}\right) ||u||_U. \tag{3.120}$$

The combination of (3.117) and (3.120) yields the estimate that

$$\left|\left|R_1^h(p,q)[u]\right|\right|_W \leq ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(||f||_{\mathrm{Lip}(\gamma,E,F)} + 2^{\gamma-1}||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}\right) ||q-p||_U^{\gamma-1}||u||_U. \tag{3.121}$$

Taking the supremum over $u \in U$ with unit U-norm in (3.121) yields the estimate that

$$\left|\left|R_1^h(p,q)\right|\right|_{\mathbf{L}(U,W)} \leq ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(||f||_{\mathrm{Lip}(\gamma,E,F)} + 2^{\gamma-1}||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}\right) ||q-p||_U^{\gamma-1}. \tag{3.122}$$

Finally, we complete the proof by combining the various estimates we have established for h to obtain the $\mathrm{Lip}(\gamma, E, W)$-norm bound claimed in (3.103).

We start this task by combining (3.114) and (3.119) to deduce that for every $p, q \in E$ we have

$$\left|\left|R_0^h(p,q)\right|\right|_W \leq \begin{cases} ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(2 + ||f||_{\mathrm{Lip}(\gamma,E,F)}\right) ||q-p||_U^{\gamma} & \text{if } ||q-p||_U > 1 \\ ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(||f||_{\mathrm{Lip}(\gamma,E,F)} + 2^{\gamma}||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}\right) ||q-p||_U^{\gamma} & \text{if } ||q-p||_U \leq 1. \end{cases} \tag{3.123}$$

Moreover, the combination of (3.116) and (3.122) yields the estimate that

$$\left|\left|R_1^h(p,q)\right|\right|_{\mathbf{L}(U,W)} \leq \begin{cases} 2||g||_{\mathrm{Lip}(\gamma,F,W)}||f||_{\mathrm{Lip}(\gamma,E,F)}||q-p||_U^{\gamma-1} & \text{if } ||q-p||_U > 1 \\ ||g||_{\mathrm{Lip}(\gamma,F,W)} \left(||f||_{\mathrm{Lip}(\gamma,E,F)} + 2^{\gamma-1}||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}\right) ||q-p||_U^{\gamma-1} & \text{if } ||q-p||_U \leq 1. \end{cases} \tag{3.124}$$

A consequence of (3.123) is that

$$\left|\left|R_0^h(p,q)\right|\right|_W \leq (1+2^{\gamma}) ||g||_{\mathrm{Lip}(\gamma,F,W)} \max\left\{||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}, 1\right\} ||q-p||_U^{\gamma}, \tag{3.125}$$

whilst a consequence of (3.124) is that

$$\left|\left|R_1^h(p,q)\right|\right|_{\mathbf{L}(U,W)} \leq \left(1+2^{\gamma-1}\right) ||g||_{\mathrm{Lip}(\gamma,F,W)} \max\left\{||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}, 1\right\} ||q-p||_U^{\gamma-1}. \tag{3.126}$$

Therefore, by combining (3.112), (3.113), (3.125), and (3.126), we conclude both that $h = (h^0, h^1) \in \mathrm{Lip}(\gamma, E, W)$ and that

$$||h||_{\mathrm{Lip}(\gamma,E,W)} \leq (1+2^{\gamma}) ||g||_{\mathrm{Lip}(\gamma,F,W)} \max\left\{||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}, 1\right\}. \tag{3.127}$$

□

3.4.3 Optimality

Handling the terms individually in the proof of Lemma 3.35 means it is unlikely that $C_\gamma = 1 + 2^\gamma$ is optimal. However, $||h||_{\mathrm{Lip}(\gamma,E,G)}$ being of order $||f||^{\gamma}_{\mathrm{Lip}(\gamma,E,F)}$ is optimal,

as shown by the following example.

Take $E = \{0, a\} \subset \mathbb{R}$ with $a < 1$, $F = \{0, 1\} \subset \mathbb{R}$, and $f \in \mathrm{Lip}(\gamma, E, F)$ with $\gamma \in (1, 2]$ defined by $f^0(0) = 0$, $f^0(a) = 1$, and $f^1(0) = f^1(a) = c$. Then,

$$\|f\|_{\mathrm{Lip}(\gamma,E,F)} = \max\left\{1, |c|, \frac{|1 - ca|}{a^\gamma}\right\}, \tag{3.128}$$

and $\|f\|_{\mathrm{Lip}(\gamma,E,F)}$ is minimised when

$$c = c^* = \begin{cases} 1, & a + a^\gamma > 1, \\ \frac{1}{a+a^\gamma}, & a + a^\gamma \leq 1, \end{cases} \tag{3.129}$$

with $\|f\|_{\mathrm{Lip}(\gamma,E,F)} = c^*$. Take $G = \{-1, 1\} \subset \mathbb{R}$ and $g \in \mathrm{Lip}(\gamma, F, G)$ defined by $g^0(0) = -1$, $g^0(1) = 1$, and $g^1(0) = g^1(1) = 1$. Then $\|g\|_{\mathrm{Lip}(\gamma,F,G)} = 1$. Further, $h = g \circ f$ is defined by $h^0(0) = -1$, $h^0(a) = 1$, and $h^1(0) = h^1(a) = c$, with

$$\|h\|_{\mathrm{Lip}(\gamma,E,G)} = \max\left\{1, |c|, \frac{|2 - ca|}{a^\gamma}\right\}. \tag{3.130}$$

When $a + a^\gamma \leq 1$,

$$\|h\|_{\mathrm{Lip}(\gamma,E,G)} = \frac{2 - ca}{a^\gamma} = \frac{1}{a + a^\gamma} + \frac{1}{a^\gamma}. \tag{3.131}$$

Then

$$\begin{aligned} \lim_{a\to 0} \frac{\|h\|_{\mathrm{Lip}(\gamma,E,G)}}{\|f\|^\beta_{\mathrm{Lip}(\gamma,E,F)}} &= \lim_{a\to 0} (a + a^\gamma)^{\beta-1} + \frac{(a + a^\gamma)^\beta}{a^\gamma}, \\ &= \lim_{a\to 0} a^{\beta-1}(1 + a^{\gamma-1})^{\beta-1} + a^{\beta-\gamma}(1 + a^{\gamma-1})^\beta, \\ &= \lim_{a\to 0} a^{\beta-\gamma}(1 + a^{\gamma-1})^{\beta-1}(2a^{\gamma-1} + 1), \\ &= \begin{cases} \infty & \beta < \gamma, \\ 1 & \beta = \gamma, \\ 0 & \beta > \gamma. \end{cases} \end{aligned} \tag{3.132}$$

Therefore, $||h||_{\mathrm{Lip}(\gamma,E,G)}$ being of order $||f||^\gamma_{\mathrm{Lip}(\gamma,E,F)}$ is optimal.

The same example can be used to obtain a lower bound on C_γ. Letting $a^\gamma + a = 1$, then $\|f\|_{\mathrm{Lip}(\gamma,E,F)} = 1$ and $\|g\|_{\mathrm{Lip}(\gamma,F,G)} = 1$, but

$$\|h\|_{\mathrm{Lip}(\gamma,E,G)} = \frac{2 - a}{a^\gamma} = 1 + a^{-\gamma}. \tag{3.133}$$

Figure 3.1 shows the range of C_γ when $\gamma \in (1, 2]$ given by this example and Lemma 3.35. This example implies there is a jump in C_γ at $\gamma = 1$, as standard Lipschitz composition gives $C_1 = 1$, whereas $C_\gamma > 3$ for all $\gamma \in (1, 2]$.

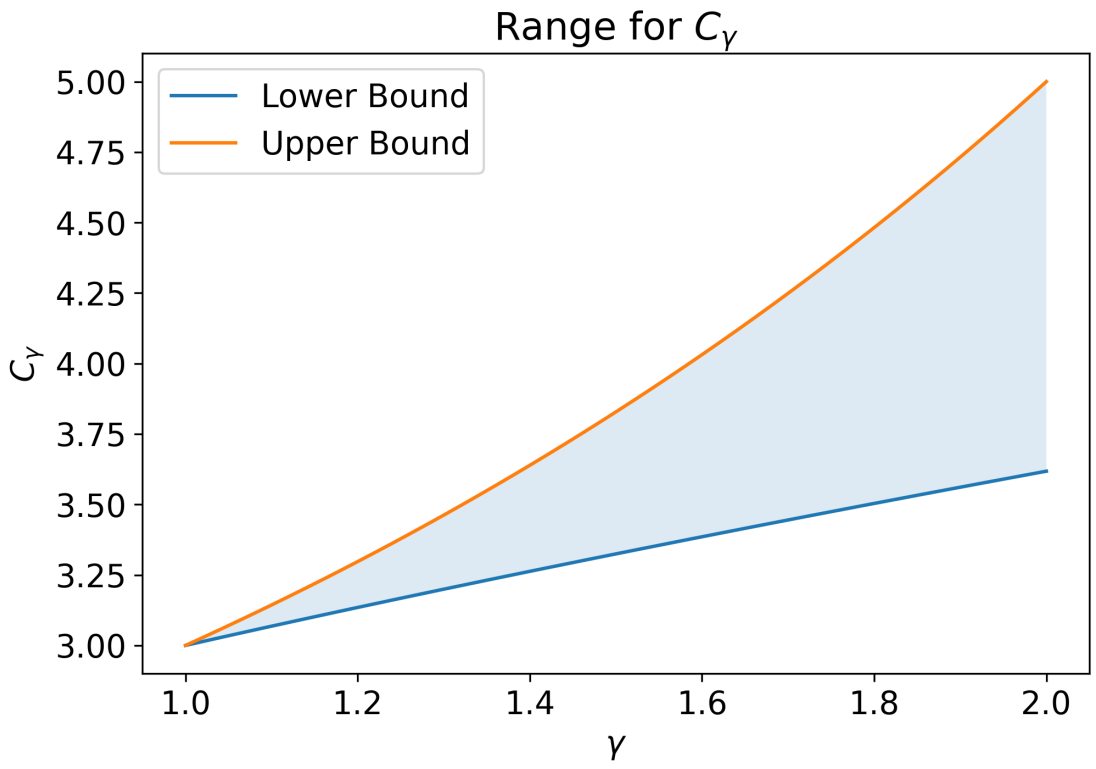


Figure 3.1: Let U, V, and W be Banach spaces, with $E \subset U$ and $F \subset V$. If $f \in \mathrm{Lip}(\gamma, E, F)$ and $g \in \mathrm{Lip}(\gamma, F, W)$, then their composition $h = g \circ f$ satisfies $\|h\|_{\mathrm{Lip}(\gamma,E,W)} \leq C_\gamma \|g\|_{\mathrm{Lip}(\gamma,F,W)} \max\left\{1, \|f\|^\gamma_{\mathrm{Lip}(\gamma,E,F)}\right\}$. This figure plots the bounds on C_γ obtained by combining Lemma 3.35 with the example of Section 3.4.3, namely $1 + a(\gamma)^{-\gamma} \leq C_\gamma \leq 1 + 2^\gamma$, where $a(\gamma)$ is the unique solution to $a^\gamma + a = 1$.

3.4.4 Future Work

At present, extending Lemma 3.35 by finding a bound on C_γ for $\gamma > 2$ remains open work. Proving such a bound would extend the explicit dimension-free control of composition to higher regularity classes. This would make it possible to give fully quantitative bounds for the $\mathrm{Lip}(\gamma)$-norms of neural architectures built by repeated composition, with constants depending explicitly on quantities such as depth and layer norms. Furthermore, such a result would support explicit error estimates for CDEs with neural network vector fields in higher-regularity regimes.

However, the proof of an explicit bound in the first non-trivial regime $\gamma \in (1, 2]$ is already involved. This motivates exploring alternative approaches for the case $\gamma > 2$. In finite-dimensional U and V one could try to extend f and g to the whole spaces via the Stein–Whitney theorem (Theorem 3.31) and then invoke the equivalence of $\mathrm{Lip}(\gamma)$ and $C_b^{k,\alpha}$ on open convex sets (Lemma 3.18) to apply classical $C_b^{k,\alpha}$ compositional bounds. However, this approach produces a constant depending on the dimension of U and V, as opposed to the dimension-free constant of Lemmas 3.34 and 3.35. Another possible approach is to use the polynomial viewpoint of $\mathrm{Lip}(\gamma)$ functions from Definition 3.16 together with standard results on the composition of polynomial functions.

3.5 Conclusion

This chapter developed the regularity theory needed for the CDE framework used throughout the thesis. We related $\mathrm{Lip}(\gamma)$ regularity to $C^{k,\alpha}$ regularity, clarifying both where these notions agree and where $\mathrm{Lip}(\gamma)$ has stronger global properties. We then introduced the derivative and Lie bracket of $\mathrm{Lip}(\gamma)$ functions on arbitrary subsets of Banach spaces. Finally, we studied the composition of $\mathrm{Lip}(\gamma)$ functions and proved an explicit bound on the composition norm in the first non-trivial regime $1 < \gamma \leq 2$. Together, these results provide the framework necessary for Chapter 4 to apply the Log-ODE method to NCDEs, where the vector fields are parametrised by neural networks.

Chapters 2 and 3 have provided the mathematical tools needed to develop the scalable and efficient continuous-time machine learning models that are the focus of the remainder of this thesis. This begins with Chapter 4 examining how to solve NCDEs

efficiently via the Log-ODE method. Chapter 5 then introduces Linear NCDEs, which enable parallel-in-time computation without sacrificing expressivity.

Chapter 4

Neural Controlled Differential Equations

> *Forty feet, down two and a half. Kicking up some dust. Thirty feet, two and a half down. Faint shadow.*

—Buzz Aldrin, *Apollo 11 Air-to-Ground Voice Transcription* (1969)

4.1 Introduction

Time series modelling is the development of mathematical, statistical, and computational methods for sequentially ordered data. It has played a central role across science and technology for nearly a century, from Yule's 1927 autoregressive models for sunspot numbers, through Kalman filtering in the Apollo guidance system for estimating the Lunar Module's altitude, to the autoregressive models that power today's large-scale language models [Yule 1927; Kalman 1960; Touvron et al. 2023].

Figure 4.1 illustrates the problem of interest for this chapter: given a sequence of irregularly spaced observations $\{X_{t_i}\}_{i=0}^n = \{(t_i, x_{t_i})\}_{i=0}^n$ from a multi-dimensional process, predict a corresponding output path y_t. This framework encompasses classification, where y_t is a single label, regression, where y_t varies over time, and autoregressive generation, where each new observation X_{t_i} is produced from the previous output $y_{t_{i-1}}$. Our focus is developing models that can generalise to unseen examples by leveraging large datasets of observation–output pairs. In particular, we explore Neural Controlled Differential Equations (NCDEs), which provide a continuous-time framework

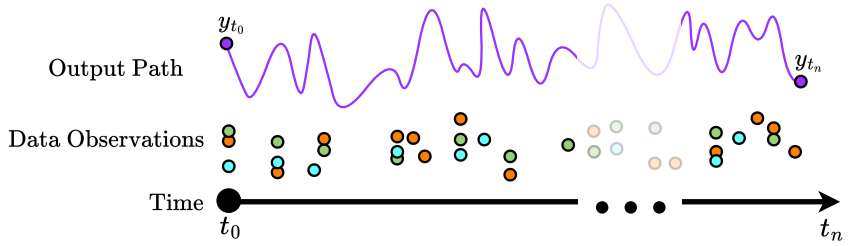


Figure 4.1: A schematic diagram of observations from a three-dimensional, irregularly sampled time series and a corresponding output path one wishes to predict. The colour of each ball represents a different channel in the time series.

naturally suited to irregularly sampled data. The development of NCDEs builds on a rich history of time series modelling, from which we note a few of the important milestones.

4.2 Historical Milestones

4.2.1 Classical Approaches

In 1927, Yule introduced autoregressive modelling, which approximates each term in a time series as a linear function of its previous values [Yule 1927]. The model is defined as

$$y_{t_i} = a_1 y_{t_{i-1}} + a_2 y_{t_{i-2}} + \cdots + a_p y_{t_{i-p}} + \epsilon_{t_i}, \tag{4.1}$$

where $a_1, a_2, \ldots, a_p$ are the autoregressive coefficients, p is the model order, and ϵ_{t_i} are independent and identically distributed random variables with mean zero and constant variance. Building on this work, the Yule–Walker equations were derived as a method for estimating the parameters of the model, laying the groundwork for statistical time series analysis [Walker 1931].

In parallel, Slutsky introduced moving average processes, which approximate each term in a time series as a linear function of previous noise values ϵ_{t_i} [Slutsky 1927; Slutsky 1937]. The model is defined as

$$y_{t_i} = \epsilon_{t_i} + b_1 \epsilon_{t_{i-1}} + b_2 \epsilon_{t_{i-2}} + \cdots + b_q \epsilon_{t_{i-q}}, \tag{4.2}$$

where $b_1, b_2, \ldots, b_q$ are the moving average coefficients. These two lines of development were unified in the autoregressive moving average model by Whittle in 1951,

and later popularised as part of the Box–Jenkins framework [Whittle 1951; Box et al. 1970].

In 1965, building on Galtieri's 1964 work on estimation in discrete-time processes, Åström and Bohlin introduced a numerical method for identifying linear dynamical systems from observed input-output data [Galtieri 1964; Åström et al. 1965]. Their model extended the autoregressive moving average framework to include an observed input series x_{t_i},

$$y_{t_i} = a_1 y_{t_{i-1}} + \cdots + a_p y_{t_{i-p}} + \epsilon_{t_i} + b_1 \epsilon_{t_{i-1}} + \cdots + b_q \epsilon_{t_{i-q}} + c_0 x_{t_i} + c_1 x_{t_{i-1}} + \cdots + c_r x_{t_{i-r}}, \quad (4.3)$$

where $c_0, c_1, \ldots, c_r$ are the input coefficients. The parameters are then estimated by maximum likelihood using a numerical optimisation procedure. This line of work was later consolidated under the name system identification [Åström et al. 1971].

A separate line of development focused on estimating signals or hidden states from noisy observations once a model had been specified. Early work in this direction includes Kolmogorov's 1941 treatment of interpolation and extrapolation for stationary random sequences [Kolmogorov 1941; Kolmogorov 1962]. In 1949, Wiener developed the Wiener filter, a method for minimising the mean-squared error when estimating a signal from noisy observations [Wiener 1949]. For a stationary process, the Wiener filter seeks to find an optimal linear filter $l(t)$ such that, when convolved with the observed signal $x(t)$, it produces an estimate $\hat{y}(t)$ of the desired signal $y(t)$,

$$\hat{y}(t) = \int_{-\infty}^{\infty} l(t-\tau) x(\tau)\, \mathrm{d}\tau. \quad (4.4)$$

Wiener provided an explicit solution for the filter in the frequency domain.

In 1960, Kalman formalised the state-space model framework for non-stationary processes and proposed the Kalman filter, an optimal recursive algorithm for estimating the hidden states of such processes [Kalman 1960]. The state-space model is defined by

$$\begin{aligned} h_{t_i} &= A h_{t_{i-1}} + B x_{t_i} + w_{t_i}, \\ y_{t_i} &= C h_{t_i} + v_{t_i}, \end{aligned} \quad (4.5)$$

where h_{t_i} is the hidden state, A is the state transition matrix, x_{t_i} are the inputs, B represents the influence of the inputs, y_{t_i} is the observed output, C is the observation matrix, and w_{t_i} and v_{t_i} are zero-mean Gaussian noise processes with known covariance

matrices. For linear Gaussian systems, the Kalman filter provides an optimal recursive solution for estimating the hidden states h_{t_i}. Subsequent advancements extended the Kalman filter to accommodate non-linear dynamics and non-Gaussian noise, leading to algorithms such as the Extended Kalman Filter, Unscented Kalman Filter, and Particle Filters [Gelb 1974; Julier et al. 1997; Gordon et al. 1993]. A further development was the extension of system identification methods from the autoregressive moving average models in (4.3) to the state-space setting of (4.5). In particular, Van Overschee and De Moor introduced N4SID in 1994, a subspace method that identifies a discrete-time state-space realisation from input-output data [Van Overschee et al. 1994].

In 1978, O'Hagan introduced a Bayesian non-parametric framework for curve fitting and prediction that laid the groundwork for what is now known as Gaussian process regression [O'Hagan 1978]. Given observations at times $T = (t_1, \dots, t_n)$, one assumes that

$$y_{t_i} = \eta(t_i) + \epsilon_{t_i}, \tag{4.6}$$

where η is an unknown regression function and $\epsilon_{t_i} \sim \mathcal{N}\big(0, \sigma^2(t_i)\big)$ independently. Rather than restricting η to a finite-dimensional parametric family, the method assumes that for any finite collection of times $S = (s_1, \dots, s_m)$, the vector

$$\big(\eta(s_1), \dots, \eta(s_m)\big)^\top \tag{4.7}$$

is jointly Gaussian with mean

$$m(S) = \big(m(s_1), \dots, m(s_m)\big)^\top \tag{4.8}$$

and covariance

$$K(S,S) = \begin{bmatrix} k(s_1,s_1) & \cdots & k(s_1,s_m) \\ \vdots & \ddots & \vdots \\ k(s_m,s_1) & \cdots & k(s_m,s_m) \end{bmatrix}, \tag{4.9}$$

where m and k specify the prior distribution of η. Let T_* denote a collection of test times, and let $\Sigma(T) = \operatorname{diag}\big(\sigma^2(t_1), \dots, \sigma^2(t_n)\big)$. Conditioning on the observed data yields the posterior distribution

$$\begin{aligned} \eta(T_*) \mid y \sim \mathcal{N}\Big(& m(T_*) + K(T_*,T)\big(K(T,T) + \Sigma(T)\big)^{-1}\big(y - m(T)\big), \\ & K(T_*,T_*) - K(T_*,T)\big(K(T,T) + \Sigma(T)\big)^{-1}K(T,T_*)\Big). \end{aligned} \tag{4.10}$$

This yields both a prediction for η at the test times and a corresponding posterior uncertainty estimate. O'Hagan's original paper developed this framework in the context

of Bayesian smoothing, curve fitting, and prediction, and it was later popularised in machine learning as a flexible kernel-based approach to non-linear regression [Williams et al. 1996; Rasmussen et al. 2006].

4.2.2 Discrete Machine Learning Approaches

In 1990, Elman introduced a recurrent neural network (RNN) architecture from which many subsequent RNNs would evolve [Elman 1990]. The Elman network is defined as

$$h_{t_i} = \sigma(W_h h_{t_{i-1}} + W_x x_{t_i} + b_h), \tag{4.11}$$

$$y_{t_i} = W_y h_{t_i} + b_y, \tag{4.12}$$

where h_{t_i} is the hidden state at time t_i, σ is an activation function, W_h, W_x, W_y are learnable weight matrices, and b_h, b_y are learnable bias vectors. The matrices W_h, W_x, and W_y play analogous roles to those of A, B, and C in (4.5), respectively. The generic architecture of an RNN can be expressed as

$$\begin{aligned} h_{t_i} &= g_\theta(h_{t_{i-1}}, x_{t_i}), \\ y_{t_i} &= l_\psi(h_{t_i}), \end{aligned} \tag{4.13}$$

where g_θ is a learnable non-linear function parametrised by θ, and l_ψ is a learnable affine transformation parametrised by ψ. Challenges such as vanishing and exploding gradients during training led to the development of more advanced architectures, including Long Short-Term Memory networks (LSTMs) in 1997 [Hochreiter et al. 1997], Gated Recurrent Units (GRUs) in 2014 [Cho et al. 2014], and Linear Recurrent Units (LRUs) in 2023 [Orvieto et al. 2023].

In 2015, Bahdanau et al. introduced the attention mechanism to RNN-based encoder-decoder models to enhance their ability to capture long-range dependencies [Bahdanau et al. 2015]. A widely used variant is the scaled dot-product attention of Vaswani et al. [2017]. Given matrices $X \in \mathbb{R}^{n \times d_X}$ and $Y \in \mathbb{R}^{m \times d_Y}$ representing time series of dimension d_X and d_Y respectively, scaled dot-product attention computes

$$\text{Attention}(Y, X) = \text{softmax}\left(\frac{(YW_Q)(XW_K)^\top}{\sqrt{d_k}}\right)(XW_V), \tag{4.14}$$

where $W_Q \in \mathbb{R}^{d_Y \times d_k}$, $W_K \in \mathbb{R}^{d_X \times d_k}$, and $W_V \in \mathbb{R}^{d_X \times d_v}$ are learnable weight matrices, and the softmax function is applied row-wise to normalise the attention scores. Here,

YW_Q, XW_K, and XW_V are referred to as the queries, keys, and values respectively. This mechanism allows the model to focus on different parts of the input sequence when generating each part of the output.

In 2017, Vaswani et al. built the Transformer architecture around scaled dot-product attention, entirely removing recurrent layers in favour of self-attention mechanisms [Vaswani et al. 2017]. In self-attention, the input sequence generates the queries, keys, and values ($X = Y$) within each layer. This design allows each position in the sequence to attend to all other positions, effectively capturing long-range dependencies. Combining the Transformer architecture with autoregressive modelling has led to significant breakthroughs in natural language processing and forms the foundation of modern large language models [Brown et al. 2020; Touvron et al. 2023]. However, for sequence length n, the computational complexity of the attention mechanism scales as $\mathcal{O}(n^2)$, compared to $\mathcal{O}(n)$ for RNNs.

4.2.3 Neural Differential Equations

In 1987, Pineda theoretically explored training a continuous-time recurrent neural network

$$\frac{\mathrm{d}h_t}{\mathrm{d}t} = -h_t + \sigma(W_h h_t) + I_t, \tag{4.15}$$

where

$$h_t = \begin{bmatrix} h_t^i \\ h_t^h \\ h_t^o \end{bmatrix}, \tag{4.16}$$

with h^i, h^h, and h^o being designated input, hidden, and output nodes, and

$$I_t = \begin{bmatrix} X_t \\ 0 \\ 0 \end{bmatrix}, \tag{4.17}$$

with X_t being a continuous input stream [Pineda 1987]. In 1989, building on earlier work which applied the adjoint sensitivity method to differential equations for optimal control problems [Bryson et al. 1962], Pearlmutter [1989] introduced an approach for computing the gradients of (4.15) by solving a backwards-in-time ODE. Theoretically, this method allows gradient calculation without storing the intermediate hidden states of the network, although in practice Pearlmutter [1989] did store the intermediate states. To the best of our knowledge, Pearlmutter [1989] also trained the first neural

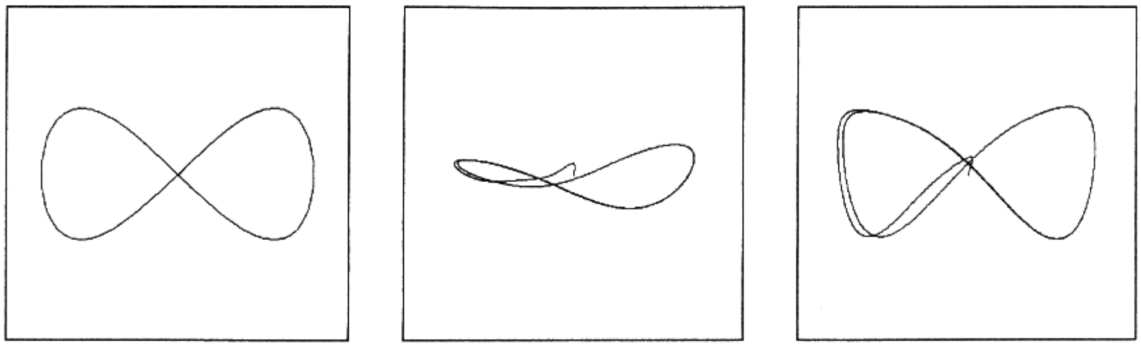


Figure 4.2: "Desired states d_1 and d_2 plotted against each other (left); actual states h_1 and h_2 plotted against each other at epoch 3,182 (centre) and 20,000 (right)". Reproduced with permission from Pearlmutter [1989].

differential equations to output desired trajectories, with Figure 4.2 showing their results on a figure eight dataset.

In 1992, Rico-Martínez et al. parametrised the vector field of a differential equation using a neural network,

$$\frac{\mathrm{d}h_t}{\mathrm{d}t} = f_\theta(t, h_t), \tag{4.18}$$

and modelled the electrodissolution of copper in phosphoric acid solution [Rico-Martínez et al. 1992]. In contrast to the adjoint sensitivity method developed by Pearlmutter [1989], Rico-Martínez et al. [1992] backpropagated directly through the steps of their differential equation solve, which requires storing the intermediate hidden states.

In 2015, He et al. introduced Residual Neural Networks (ResNets), which allowed the training of very deep networks [He et al. 2016]. This was achieved by introducing skip connections, which allow the input to bypass one or more layers, mitigating issues such as vanishing gradients. The update rule for a ResNet layer is given by

$$h_{i+1} = h_i + f^i_{\theta_i}(h_i), \tag{4.19}$$

where h_i is the hidden state at layer i, and the $f^i_{\theta_i}$ are non-linear functions parametrised by θ_i respectively. In 2018, Chen et al. observed that (4.19) resembles an Euler discretisation of (4.18), and proposed Neural ODEs as a continuous-depth analogue to ResNets [Chen et al. 2018]. Following a similar approach to Pearlmutter [1989], Chen et al. [2018] derived a method for calculating the gradients of solutions to (4.18) using the adjoint sensitivity method. Furthermore, they utilised the ability to not store

the intermediate hidden states to significantly reduce memory requirements during training.

These advancements in modelling continuous-time dynamics using neural networks set the stage for the development of NCDEs, the continuous-time analogue to recurrent neural networks.

4.3 Neural Controlled Differential Equations

4.3.1 Definition

Definition 4.1 (Neural Controlled Differential Equation [Kidger et al. 2020]). *Let $X : [t_0, t_n] \to \mathbb{R}^{d_X}$ be a continuous interpolation of $\{(t_i, x_{t_i})\}_{i=0}^n$, such that $X_{t_i} = (t_i, x_{t_i})$. Let $\xi_\phi : \mathbb{R}^{d_X} \to \mathbb{R}^{d_h}$ and $f_\theta : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h \times d_X}$ be neural networks and $l_\psi : \mathbb{R}^{d_h} \to \mathbb{R}^{d_y}$ be a linear map with learnable parameters ϕ, θ, and ψ, respectively. An NCDE is defined by*

$$\begin{aligned} h_{t_0} &= \xi_\phi(X_{t_0}), \\ h_t &= h_{t_0} + \int_{t_0}^{t} f_\theta(h_s)\mathrm{d}X_s, \\ y_t &= l_\psi(h_t). \end{aligned} \tag{4.20}$$

An NCDE consists of three learnable maps. First, ξ_ϕ maps the initial observation to the initial hidden state h_{t_0}. Second, f_θ maps each hidden state h_s to a linear map that determines how increments of the control path X update the hidden state. Finally, l_ψ maps the hidden state h_t to the output y_t.

The central innovation of NCDEs is that they interface with the data through a continuous control path X. Although Definition 4.1 is stated for irregularly sampled, fully observed data, the same framework applies more broadly whenever the observations can be encoded as a continuous path X. For example, if only a subset of the components of x_{t_i} is observed at time t_i, then one may either construct the control path channel-wise or first impute the missing values and then build X from the resulting completed observations. As another example, when the observations x_{t_i} take discrete values, one may first encode them in a Euclidean space and then choose X to be the rectilinear interpolation [Morrill et al. 2022], which between successive observations follows the path

$$(t_i, x_{t_i}) \to (t_{i+1}, x_{t_i}) \to (t_{i+1}, x_{t_{i+1}}). \tag{4.21}$$

The choice of X requires some care, since different constructions expose different information to the model, affecting both the learned dynamics and the resulting performance. In particular, some choices make X_t depend on observations from times later than t, rendering them unsuitable for online settings. Morrill et al. [2022] theoretically and empirically studied a range of choices for X, and introduced rectilinear controls for online prediction tasks. From this point onward, we assume that a suitable online control path X has already been constructed from the observations.

To make use of the techniques developed for training Neural ODEs, NCDEs are typically rewritten as an ODE,

$$\tilde{h}_t = \tilde{h}_{t_0} + \int_{t_0}^{t} g_{\theta,X}(\tilde{h}_s, s)\mathrm{d}s, \tag{4.22}$$

where $\tilde{h}$ denotes the hidden state in this ODE representation, with $\tilde{h}_{t_0} = h_{t_0}$. Originally, Kidger et al. [2020] proposed taking X to be a differentiable interpolation and

$$g_{\theta,X}(\cdot) = f_\theta(\cdot)\frac{\mathrm{d}X}{\mathrm{d}s}, \tag{4.23}$$

which gives $\tilde{h}_t = h_t$ for all t. The Log-ODE method will allow us to retain the ODE form (4.22) for a wider class of driving paths, at the expense of replacing the exact dynamics by an approximation, such that $\tilde{h}_t \approx h_t$.

4.3.2 Comparison with Alternative Approaches

NCDEs are closely related to Neural ODEs (4.18). The key difference is that the trajectory of a Neural ODE's hidden state is determined entirely by its initial condition and the learned vector field, making them unsuitable for time series data. Methods such as GRU-ODE and ODE-RNN address this by combining continuous-time evolution with discrete recurrent updates at observation times [Brouwer et al. 2019; Rubanova et al. 2019]. In contrast, NCDEs incorporate the incoming signal directly into the dynamics, and so may be viewed as a continuous-time analogue of RNNs.

This connection can be made concrete by considering the residual RNN [Yue et al. 2018]

$$h_{t_{i+1}} = h_{t_i} + g_\theta(h_{t_i}, X_{t_{i+1}} - X_{t_i}). \tag{4.24}$$

Just as a ResNet can be interpreted as a discretisation of a Neural ODE, (4.24) can be interpreted as the discretisation of a continuous process

$$\mathrm{d}h_s = g_\theta(h_s, \mathrm{d}X_s). \tag{4.25}$$

The key structural difference is that (4.25) depends non-linearly on the increment $\mathrm{d}X_s$, whereas an NCDE depends linearly on the increment. However, this linear dependence does not reduce theoretical expressivity, as discussed further in Section 4.3.3.

Alternative irregular-time methods include Gaussian processes and Neural Processes, which also naturally support predictive uncertainty quantification [O'Hagan 1978; Williams et al. 1996; Rasmussen et al. 2006; Garnelo et al. 2018]. In particular, causal Gaussian processes respect the same online temporal structure as the sequential prediction setting relevant for NCDEs [Cunningham et al. 2012]. Such approaches are appealing when quantifying predictive uncertainty is itself a central objective.

However, the focus of this thesis is on understanding parametric, causal, continuous-time models for sequential prediction. Therefore, to isolate the core questions of representation, architecture, and numerical approximation, we work in a simplified deterministic setting. Once the observations have been converted into a control path, the path X is treated as fixed, the output path y is treated as deterministic, and the maps ξ_ϕ, f_θ, and l_ψ are taken to be deterministic parametrised functions. This excludes explicit modelling of aleatoric and epistemic uncertainty, but allows this thesis to focus on the fundamental mechanisms by which NCDEs process continuous paths and propagate information through time. There are natural extensions of the NCDE framework that do incorporate uncertainty, including Neural Stochastic Differential Equations and Bayesian Neural Controlled Differential Equations [Kidger et al. 2021; Hess et al. 2024]. Furthermore, the numerical and architectural advances developed in this thesis are compatible with these uncertainty-aware settings, although we do not pursue those extensions here.

4.3.3 Expressivity

Definition 4.2 (Maximal expressivity [Walker et al. 2025])**.** *Let $\mathcal{X}$ be a topological space, and let $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathbb{R} \mid \theta \in \Theta\}$ be a class of real-valued functions on $\mathcal{X}$, parametrised by some set Θ. We say that $\mathcal{F}$ is maximally expressive (or universal) on $\mathcal{X}$ if, for every compact set $\mathcal{K} \subset \mathcal{X}$ and every real-valued continuous function*

$f : \mathcal{K} \to \mathbb{R}$, *the following property holds:*

$$\forall \epsilon > 0,\ \exists \theta \in \Theta \quad s.t. \quad \sup_{x \in \mathcal{K}} \left| f(x) - f_\theta(x) \right| < \epsilon. \tag{4.26}$$

Although maximal expressivity is not sufficient to ensure good performance, it is desirable, as it shows that at least theoretically the model class is rich enough to approximate any continuous target map on compact sets. A classical result is the Universal Approximation Theorem for neural networks.

Theorem 4.3 (Universal Approximation Theorem [Cybenko 1989; Hornik 1991])**.** *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous, bounded, and nonconstant. Define*

$$\mathcal{F} = \left\{ x \mapsto \sum_{j=1}^{m} a_j \sigma(w_j^\top x + b_j) \;\middle|\; m \in \mathbb{N},\ a_j \in \mathbb{R},\ w_j \in \mathbb{R}^d,\ b_j \in \mathbb{R} \right\}. \tag{4.27}$$

Then $\mathcal{F}$ is maximally expressive on $\mathbb{R}^d$.

Theorem 4.3 establishes that single hidden-layer neural networks are maximally expressive for continuous real-valued functions on $\mathbb{R}^d$. Theorem 4.4 shows that NCDEs satisfy the same notion of maximal expressivity for continuous functions of entire paths.

Theorem 4.4 (Maximally Expressive NCDEs [Kidger et al. 2020])**.** *Let $\mathcal{X}$ be the space of bounded variation paths on the interval $[t_0, t_n]$ that start at a common point and include time as a channel, endowed with the 1-variation topology. Let $\mathcal{F}$ be the class of real-valued maps $X \mapsto y_{t_n}$ induced by NCDEs from Definition 4.1 with $d_h \in \mathbb{N}$ and $d_y = 1$. Then $\mathcal{F}$ is maximally expressive on $\mathcal{X}$.*

Proof. The result follows from the signature being universal, Corollary 2.28, the truncated signature solving a linear CDE, (2.82), and the Universal Approximation Theorem for neural networks [Cybenko 1989; Hornik 1991]. For more details, see [Kidger 2022, Theorem C.25]. □

Theorem 4.4 can be extended from path-to-point functions to path-to-path functions by replacing the linear readout l_ψ with a neural network, as shown in [Cirone et al. 2024, Proposition D.2]. In addition to their important theoretical properties, NCDEs achieve superior performance on a range of datasets when compared to similar methods, such as GRU-ODE and ODE-RNN, which handle irregularly sampled data by combining Neural ODEs with RNNs [Brouwer et al. 2019; Rubanova et al. 2019].

Just as NCDEs can be viewed as the continuous-time analogue of RNNs, once an NCDE is discretised using a differential equation solver, it may be viewed as an RNN unrolled over the solver steps. If many solver steps are required, then training must backpropagate through a long sequence of hidden-state updates. Consequently, NCDEs can suffer from the same exploding and vanishing gradient issues as RNNs on long time series, leading to degraded performance [Morrill et al. 2021].

4.3.4 Neural Rough Differential Equations

A recurring issue in deep learning approaches to time series modelling is that large number of repeated forward passes through the neural network cause the gradient during training to either explode or vanish, as was first shown in [Hochreiter 1991] and explored further in [Hochreiter et al. 2001]. This issue was one of the motivations behind the development of LSTMs [Hochreiter et al. 1997]. The Log-ODE method introduced in Section 2.5 is an efficient and accurate method for approximating the solution to a CDE. Inspired by the Log-ODE method, Morrill et al. introduced neural rough differential equations (NRDEs), which replaced (4.23) with the piecewise in time

$$g_{\theta,X}(\cdot) = \bar{f}_\theta(\cdot)\frac{\log(S^N(X)_{[r_i,r_{i+1}]})}{r_{i+1}-r_i}, \quad s \in [r_i, r_{i+1}), \tag{4.28}$$

where $\bar{f}_\theta : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h \times \beta(d_X,N)}$ is a neural network and $\beta(d_X, N) = \mathcal{O}(d_X^N)$ is the dimension of $\mathfrak{L}^N(\mathbb{R}^{d_X})$, the space where the depth$-N$ truncated log-signature of a d_X dimensional path lives [Morrill et al. 2021].

Compared to NCDEs, NRDEs can reduce the number of forward passes through the network while evaluating the model, as the vector field is autonomous on each interval $[r_i, r_{i+1})$. This has been shown to lead to improved classification accuracy, alongside reduced time and memory-usage, on time series with up to 17,000 observations [Morrill et al. 2021]. Furthermore, as it is no longer necessary to apply a differentiable interpolation to the time series data, NRDEs are applicable to a wider range of input signals. By neglecting the Lie bracket structure of $\bar{f}_\theta$, NRDEs reduce the computational burden of evaluating the vector field, at the cost of increasing the output dimension of the neural network. In contrast, Log-NCDEs retain the Lie bracket structure of $\bar{f}_\theta$.

4.4 Log Neural Controlled Differential Equations

4.4.1 Definition

Log-NCDEs use the same underlying model as NRDEs,

$$\begin{aligned} h_t &= h_{t_0} + \int_{t_0}^{t} g_{\theta,X}(h_s)\mathrm{d}s, \\ g_{\theta,X}(\cdot) &= \bar{f}_\theta(\cdot)\frac{\log(S^N(X)_{[r_i,r_{i+1}]})}{r_{i+1}-r_i}, \quad s \in [r_i, r_{i+1}), \end{aligned} \tag{4.29}$$

but with two major changes. First, instead of parametrising $\bar{f}_\theta$ using a neural network, it is constructed using the iterated Lie brackets of an NCDE's neural network, f_θ. Second, f_θ is ensured to be a $\mathrm{Lip}(\gamma)$ function for $\gamma \in (N-1, N]$. These changes have a major benefit. For $N > 1$, Log-NCDEs are exploring a smaller output space during training than NRDEs, while maintaining the same expressivity, as NCDEs are maximally expressive. This is because the output dimension of f_θ is $d_h \times d_X$, whereas the output dimension of $\bar{f}_\theta$ is $d_h \times \beta(d_X, N)$, where $\beta(d_X, N) = \mathcal{O}(d_X^N)$. Figure 4.3 compares these values for paths of dimension d_X from 1 to 15 and truncation depths of $N = 1$ and $N = 2$. The reduced output dimension comes at the cost of needing to calculate the iterated Lie brackets when evaluating Log-NCDEs, which is quantified in Section 4.4.4 and explored empirically in Section 4.4.7. Figure 4.4 is a schematic diagram comparing the approaches of an NCDE and a Log-NCDE.

When $N = 1$, (4.29) simplifies to

$$g_{\theta,X}(\cdot) = f_\theta(\cdot)\frac{X_{r_{i+1}} - X_{r_i}}{r_{i+1}-r_i}, \quad s \in [r_i, r_{i+1}). \tag{4.30}$$

Hence, in this case the only difference between Log-NCDEs and NRDEs is the regularisation of f_θ. Furthermore, (4.30) and (4.23) are equivalent when X is a linear interpolation. Therefore, the approach of NCDEs, NRDEs, and Log-NCDEs coincide when using a depth-1 Log-ODE approximation [Morrill et al. 2021].

4.4.2 $\mathrm{Lip}(\gamma)$ Neural Networks

As discussed in Section 2.5, applying a depth N Log-ODE method requires the vector field f_θ to be $\mathrm{Lip}(\gamma)$ for $\gamma \in (N-1, N]$. There exist theoretical results linking the robustness of a learning algorithm to the algorithm's Lipschitz constant [Xu et al. 2012]. Furthermore, there are results bounding the Lipschitz constant of a fully

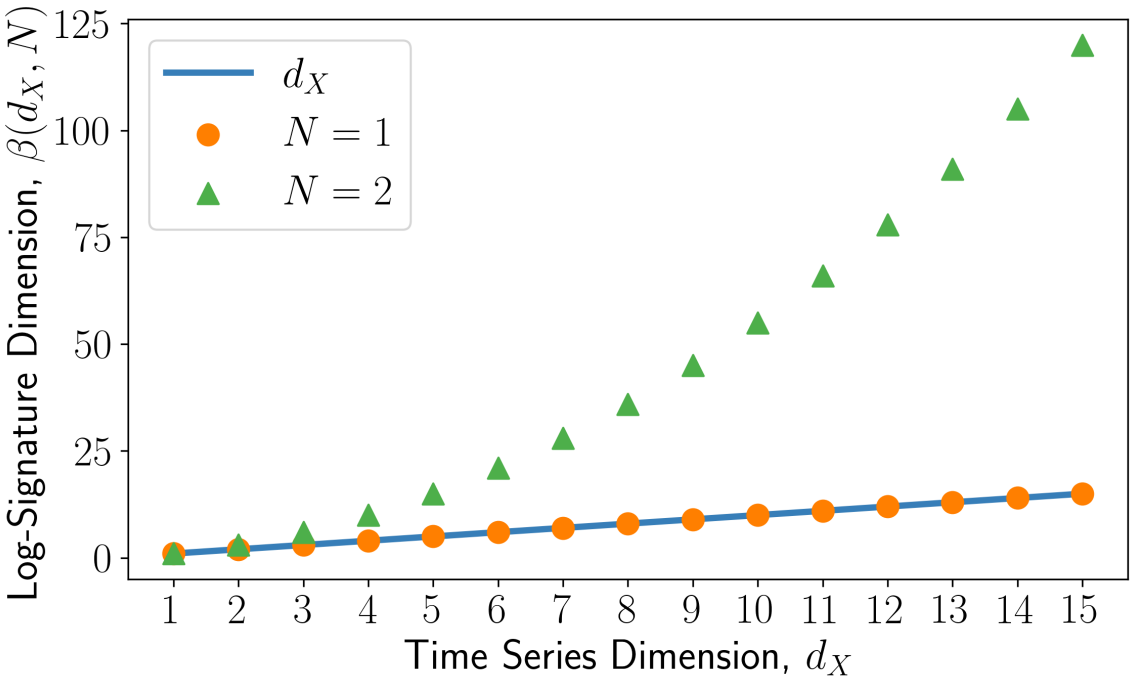


Figure 4.3: A plot of $\beta(d_X, N)$ against d_X for $N = 1, 2$. The output dimension of an NRDE's neural network is $\mathbb{R}^{d_h \times \beta(d_X, N)}$, whereas for a Log-NCDE it is $\mathbb{R}^{d_h \times d_X}$.

connected neural network (FCNN) [Szegedy et al. 2014]. Here, we extend these results to $\text{Lip}(\gamma)$ for $1 < \gamma \leq 2$.

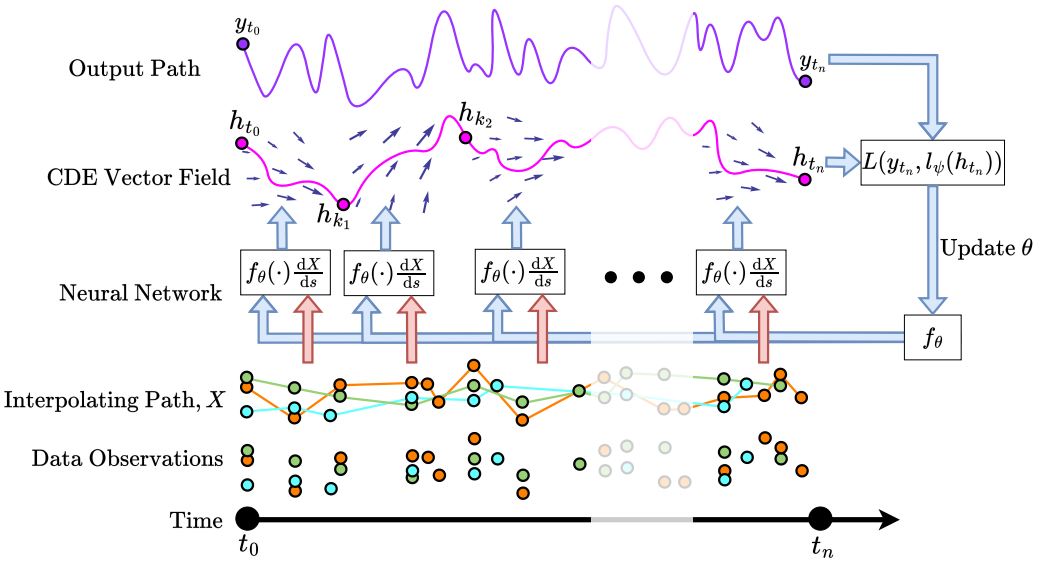
Definition 4.5 (Fully Connected Neural Network)**.** *Let $m, n_{in}, n_{out}, n_h \in \mathbb{N}$ and f_θ be a fully connected neural network (FCNN) with m layers, input dimension n_{in}, output dimension n_{out}, hidden dimension n_h, and activation function σ. Given an input $x \in \mathbb{R}^{n_{in}}$,*

$$f_\theta(x) = L^m(\cdots(L^1(x))\cdots), \tag{4.31}$$

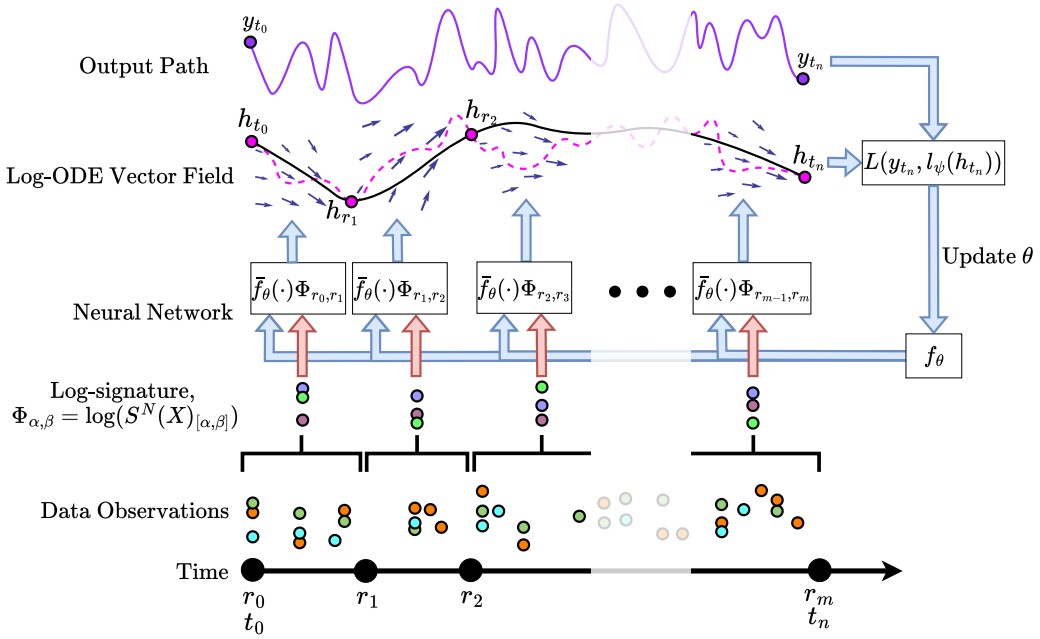
where $L^1 : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_h}$, $L^i : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ for $i = 2, \ldots, m-1$, and $L^m : \mathbb{R}^{n_h} \to \mathbb{R}^{n_{out}}$. Each layer is defined by

$$L^i(y) = \begin{bmatrix} L^i_1(y) \\ \vdots \\ L^i_\alpha(y) \end{bmatrix} = \sigma\left(\begin{bmatrix} l^i_1(y) \\ \vdots \\ l^i_\alpha(y) \end{bmatrix}\right) = \begin{bmatrix} \sigma(l^i_1(y)) \\ \vdots \\ \sigma(l^i_\alpha(y)) \end{bmatrix} = \begin{bmatrix} \sigma(W^i_1 \cdot y + b^i_1) \\ \vdots \\ \sigma(W^i_\alpha \cdot y + b^i_\alpha) \end{bmatrix}, \tag{4.32}$$

where $y \in \mathbb{R}^\beta$, $W^i = [W^i_1, \ldots, W^i_\alpha]^T \in \mathbb{R}^{\alpha \times \beta}$ and $b^i = [b^i_1, \ldots, b^i_\alpha]^T \in \mathbb{R}^\alpha$ are the



(a) A schematic NCDE. A continuous path X is constructed from the observations. Its time derivative $\frac{\mathrm{d}X}{\mathrm{d}s}$ is combined with the neural vector field f_θ to define the hidden-state dynamics, which are evolved with a differential equation solver. A linear map is then applied to h_{t_n} to produce a prediction, and the loss L against the target y_{t_n} is used to update the learnable parameters.



(b) A schematic Log-NCDE. In contrast to the NCDE, the log-signature of the driving path X is computed over intervals $\{r_i\}_{i=0}^m$ and combined with iterated Lie brackets of the neural vector field f_θ to define the hidden-state dynamics.

Figure 4.4: A schematic diagram of an NCDE and a Log-NCDE.

learnable parameters and

$$(\alpha, \beta) = \begin{cases} (n_h, n_{\text{in}}), & i = 1, \\ (n_h, n_h), & i = 2, \dots, m-1, \\ (n_{\text{out}}, n_h), & i = m. \end{cases} \tag{4.33}$$

Assumption 4.6. *Let $1 < \gamma \leq 2$. We will assume that the activation function σ satisfies the following four conditions:*

1. *σ is continuously differentiable with derivative σ',*
2. *$|\sigma(x)| \leq |x|$,*
3. *$\sup_{x \in \mathbb{R}} |\sigma'(x)| \leq M_1$, and*
4. *$[\sigma']_{\gamma-1} = \sup_{\substack{y,x \in \mathbb{R} \\ y \neq x}} \frac{|\sigma'(y) - \sigma'(x)|}{|y-x|^{\gamma-1}} \leq M_2$,*

for constants $M_1, M_2 > 0$.

Conditions 1 and 4 imply that $\sigma \in C^{1,\gamma-1}$. Additionally, by Lemma 3.17, σ satisfies all the conditions to be $\text{Lip}(\gamma)$ except $\sup_{x \in \mathbb{R}} |\sigma(x)| < \infty$, which we do not assume as it would exclude some standard activation functions which do satisfy Assumption 4.6, such as SiLU. It is worth noting that not all standard activation functions satisfy Assumption 4.6. For example, ReLU is not continuously differentiable.

Lemma 4.7. *Let $1 < \gamma \leq 2$ and f_θ be a FCNN with activation function satisfying Assumption 4.6. Take the Euclidean norm and assume the input $x \in A \subset \mathbb{R}^{n_{\text{in}}}$, where $\sup_{x \in A} \|x\|_2 = C$. Then each layer L^i satisfies*

$$\|L^i\|_{\text{Lip}(\gamma)} \leq \max\left\{\Gamma^i, M_1 \|W^i\|_{\text{op}}, M_2 \|W^i\|_{\text{op}}^\gamma\right\}, \tag{4.34}$$

where $\Gamma^i = \|W^i\|_{\text{op}} \Gamma^{i-1} + \|b^i\|_2$ for $i \geq 1$ and $\Gamma^0 = C$.

Proof. Let Y_i be the input domain to the i^{th} layer. The $\text{Lip}(\gamma)$ norm of L^i is a maximum over four terms:

1. First,
$$\sup_{y \in Y_i} \|L^i(y)\|_2 \leq \sup_{y \in Y_i} \|W^i\|_{\text{op}} \|y\|_2 + \|b^i\|_2, \tag{4.35}$$
by condition 2 in Assumption 4.6.

2. Second,

$$\sup_{y\in Y_i} \|\nabla L^i(y)\|_{\text{op}} = \sup_{y\in Y_i} \|\operatorname{diag}(\sigma'(W^i y + b^i))W^i\|_{\text{op}} \leq M_1\|W^i\|_{\text{op}}, \tag{4.36}$$

by condition 3 in Assumption 4.6.

3. Third,

$$\sup_{y\neq x} \frac{\|\nabla L^i(y) - \nabla L^i(x)\|_{\text{op}}}{\|y-x\|_2^{\gamma-1}} \leq \sup_{y\neq x} \frac{\|\operatorname{diag}(\Delta)\|_{\text{op}}}{\|y-x\|_2^{\gamma-1}} \|W^i\|_{\text{op}}, \tag{4.37}$$

where

$$\Delta_j = \sigma'(W_j^i \cdot y + b_j^i) - \sigma'(W_j^i \cdot x + b_j^i). \tag{4.38}$$

Using condition 4 in Assumption 4.6,

$$\|\operatorname{diag}(\Delta)\|_{\text{op}} \leq M_2\|y-x\|_2^{\gamma-1} \max_j \|W_j^i\|_2^{\gamma-1} \leq M_2\|y-x\|_2^{\gamma-1}\|W^i\|_{\text{op}}^{\gamma-1}. \tag{4.39}$$

Therefore,

$$\sup_{y\neq x} \frac{\|\nabla L^i(y) - \nabla L^i(x)\|_{\text{op}}}{\|y-x\|_2^{\gamma-1}} \leq M_2\|W^i\|_{\text{op}}^{\gamma}. \tag{4.40}$$

4. Fourth, since each L^i belongs to $C^{1,\gamma-1}(\mathbb{R}^\beta, \mathbb{R}^\alpha)$, Lemma 3.17 can be used to bound the final term,

$$\sup_{y\neq x} \frac{\|L^i(y) - L^i(x) - \nabla L^i(x)[y-x]\|_2}{\|y-x\|_2^{\gamma}} \leq \sup_{y\neq x} \frac{\|\nabla L^i(y) - \nabla L^i(x)\|_{\text{op}}}{\|y-x\|_2^{\gamma-1}}. \tag{4.41}$$

Therefore,

$$\|L^i\|_{\text{Lip}(\gamma)} \leq \max\left\{\sup_{y\in Y_i} \|W^i\|_{\text{op}}\|y\|_2 + \|b^i\|_2, M_1\|W^i\|_{\text{op}}, M_2\|W^i\|_{\text{op}}^{\gamma}\right\}. \tag{4.42}$$

Since $y \in Y_1$ satisfies $\|y\|_2 \leq C$,

$$\|L^i\|_{\text{Lip}(\gamma)} \leq \max\left\{\Gamma^i, M_1\|W^i\|_{\text{op}}, M_2\|W^i\|_{\text{op}}^{\gamma}\right\}, \tag{4.43}$$

where $\Gamma^i = \|W^i\|_{\text{op}}\Gamma^{i-1} + \|b^i\|_2$ for $i \geq 1$ and $\Gamma^0 = C$. □

Theorem 4.8. *Let $1 < \gamma \leq 2$ and f_θ be a FCNN with m layers and activation function σ satisfying Assumption 4.6. Assume the input $x \in A \subset \mathbb{R}^{n_{\text{in}}}$, where $\sup_{x\in A}\|x\|_2 = C$. Then $f_\theta \in \text{Lip}(\gamma)$ and*

$$||f_\theta||_{\text{Lip}(\gamma)} \leq (1+2^\gamma)^{\frac{\gamma^{m-1}-1}{\gamma-1}} \prod_{i=1}^{m} \max\left(1, \|L^i\|_{\text{Lip}(\gamma)}^{\gamma^{m-i}}\right) \tag{4.44}$$

with

$$\|L^i\|_{\text{Lip}(\gamma)} \leq \max\left\{\Gamma^i, M_1\|W^i\|_{\text{op}}, M_2\|W^i\|_{\text{op}}^{\gamma}\right\}, \tag{4.45}$$

where $\Gamma^i = \|W^i\|_{\text{op}}\Gamma^{i-1} + \|b^i\|_2$ for $i \geq 1$ and $\Gamma^0 = C$.

Proof. Lemma 3.35 states that for $f, g \in \mathrm{Lip}(\gamma)$ with $1 < \gamma \leq 2$,

$$\|g \circ f\|_{\mathrm{Lip}(\gamma)} \leq (1 + 2^\gamma)\|g\|_{\mathrm{Lip}(\gamma)} \max\{1, \|f\|^\gamma_{\mathrm{Lip}(\gamma)}\}. \tag{4.46}$$

Assume that

$$\|L^n \circ \cdots \circ L^1\|_{\mathrm{Lip}(\gamma)} \leq (1 + 2^\gamma)^{\frac{\gamma^{n-1}-1}{\gamma-1}} \prod_{i=1}^{n} \max\left(1, \|L^i\|^{\gamma^{n-i}}_{\mathrm{Lip}(\gamma)}\right), \tag{4.47}$$

which is true for $n = 1$. Then by (4.46),

$$\|L^{n+1}\circ\cdots\circ L^1\|_{\mathrm{Lip}(\gamma)} \leq (1+2^\gamma)\|L^{n+1}\|_{\mathrm{Lip}(\gamma)} \max\left(1, \left((1 + 2^\gamma)^{\frac{\gamma^{n-1}-1}{\gamma-1}} \prod_{i=1}^{n} \max\left(1, \|L^i\|^{\gamma^{n-i}}_{\mathrm{Lip}(\gamma)}\right)\right)^\gamma\right). \tag{4.48}$$

Note that for any $a, b > 0$,

$$\max(1, ab) \leq \max(1, a) \max(1, b). \tag{4.49}$$

Repeatedly applying this bound gives,

$$\begin{aligned}\|L^{n+1} \circ \cdots \circ L^1\|_{\mathrm{Lip}(\gamma)} &\leq (1 + 2^\gamma)^{\gamma\frac{\gamma^{n-1}-1}{\gamma-1}+1}\|L^{n+1}\|_{\mathrm{Lip}(\gamma)} \prod_{i=1}^{n} \max\left(1, \|L^i\|^{\gamma^{n+1-i}}_{\mathrm{Lip}(\gamma)}\right), \\ &\leq (1 + 2^\gamma)^{\frac{\gamma^{n}-1}{\gamma-1}} \prod_{i=1}^{n+1} \max\left(1, \|L^i\|^{\gamma^{n+1-i}}_{\mathrm{Lip}(\gamma)}\right).\end{aligned} \tag{4.50}$$

Therefore, (4.47) holds for all $1 \leq n \leq m$, and

$$||f_\theta||_{\mathrm{Lip}(\gamma)} = \|L^m \circ \cdots \circ L^1\|_{\mathrm{Lip}(\gamma)} \leq (1 + 2^\gamma)^{\frac{\gamma^{m-1}-1}{\gamma-1}} \prod_{i=1}^{m} \max\left(1, \|L^i\|^{\gamma^{m-i}}_{\mathrm{Lip}(\gamma)}\right). \tag{4.51}$$

Lemma 4.7 completes the proof by giving the stated bound on each layer's $\mathrm{Lip}(\gamma)$ norm. □

Although the bound in Theorem 4.8 is a worst case estimate, it reflects a genuine feature of composition in $\mathrm{Lip}(\gamma)$ spaces. In particular, the explicit example in Section 3.4.3 shows that for $\gamma \in (1, 2]$, the composition of two functions with $\mathrm{Lip}(\gamma)$ norm equal to 1 can itself have $\mathrm{Lip}(\gamma)$ norm greater than 1. Thus, even when the individual layers are uniformly controlled, composition can rapidly grow the $\mathrm{Lip}(\gamma)$ norm. Consistent with this, if each layer satisfies $\|L^i\|_{\mathrm{Lip}(\gamma)} \leq 1$, then (4.44) still gives

$$\|f_\theta\|_{\mathrm{Lip}(\gamma)} \leq (1 + 2^\gamma)^{\frac{\gamma^{m-1}-1}{\gamma-1}}, \tag{4.52}$$

which grows rapidly with the depth m.

To demonstrate this behaviour in practice, we train three neural networks to approximate $\text{sign}(x)$ for $x \in [-1, 1]$. Each neural network has a hidden dimension of 8, SiLU activation functions, and a depth of 2, 3, or 4, respectively. As can be seen in Figure 4.5, the supremum over the second derivative grows rapidly as the depth increases. This leads to the Lipschitz bound on the gradient dominating $\|f_\theta\|_{\text{Lip}(2)}$ for depths 3 and 4, which have $\|f_\theta\|_{\text{Lip}(2)} \approx 297$ and $\|f_\theta\|_{\text{Lip}(2)} \approx 16910$, respectively.

Although these empirical values are many orders of magnitude smaller than the worst case bound, they still illustrate the same qualitative phenomenon: $\|f_\theta\|_{\text{Lip}(2)}$ can grow rapidly with depth in practice. A simple way to encourage smaller parameter magnitudes during training is to introduce weight regularisation by modifying the loss function L to

$$L \mapsto L + \lambda \left(\sum_{i=1}^{m} \|W^i\|_2 + \|b^i\|_2 \right), \tag{4.53}$$

where λ is a hyperparameter controlling the weight of the penalty [Hinton 1987; Krogh et al. 1991]. This introduces only a single additional hyperparameter and directly penalises the weights and biases appearing in the layer-wise bound of Theorem 4.8. Thus, it provides a simple proxy for encouraging smaller Lip(2) norms, even though it is not a sharp estimate of $\|f_\theta\|_{\text{Lip}(2)}$.

In the experiments contained in Section 4.4.6, Log-NCDEs use a FCNN with SiLU activation functions as their vector field f_θ. The coefficient λ is treated as one component of the hyperparameter grid search, with $\lambda = 0$ included as a candidate value. Empirically, this regularisation is not uniformly beneficial: some of the best runs select $\lambda = 0$, while others select $\lambda > 0$. However, even with $\lambda = 0$, we did not observe any training divergence or solver instability.

4.4.3 Constructing the Log-ODE Vector Field

As in Section 2.5, the linear map $\bar{f}_\theta$ in (4.29) is defined recursively by

$$\bar{f}_\theta(\cdot)a = f_\theta(\cdot)a, \tag{4.54}$$

for $a \in \mathbb{R}^{d_X}$ and

$$\bar{f}_\theta(\cdot)[a, b] = [\bar{f}_\theta(\cdot)a, \bar{f}_\theta(\cdot)b], \tag{4.55}$$

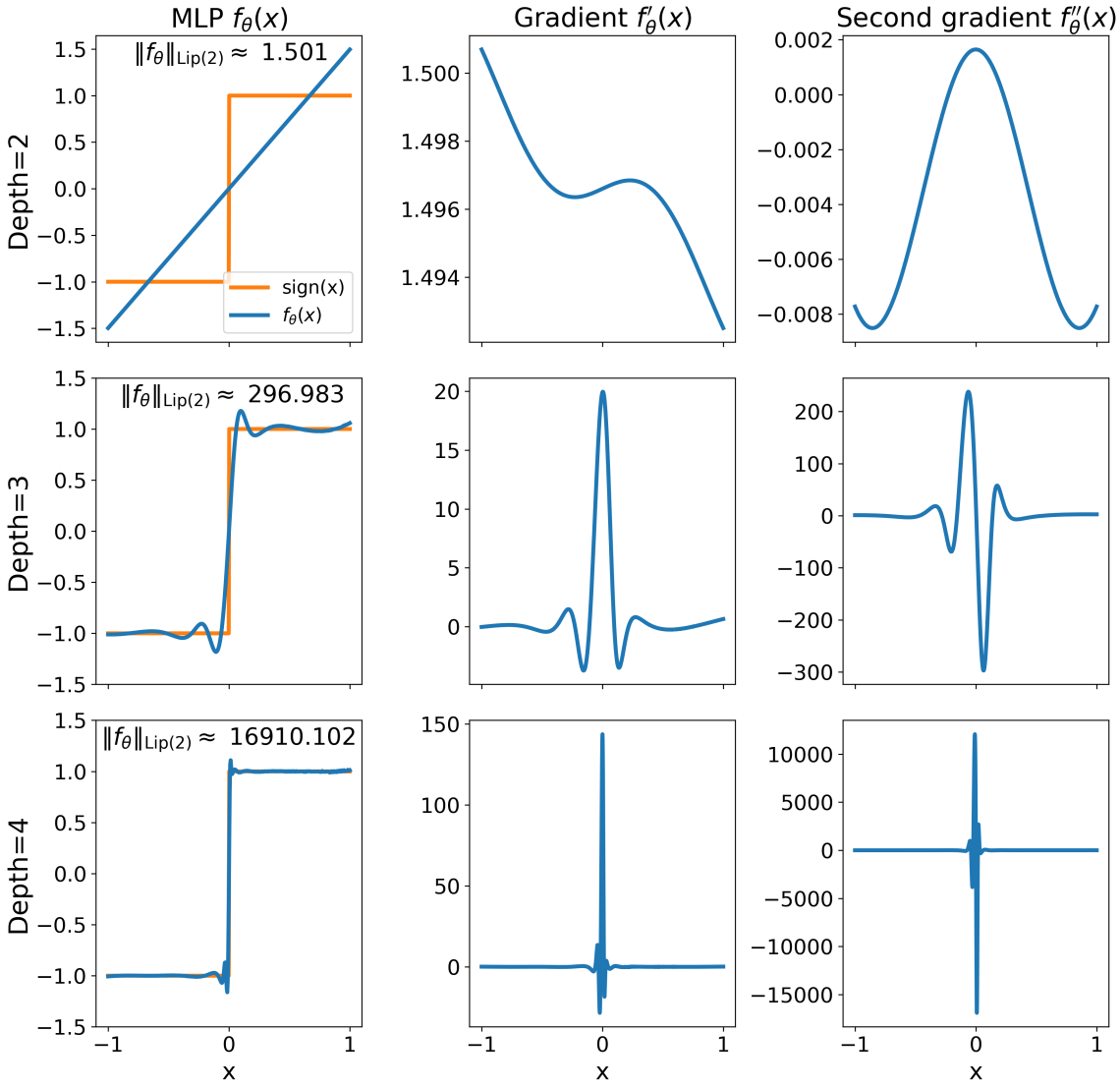


Figure 4.5: Comparing the Lip(2) norm of three fully connected neural networks trained to approximate $\text{sign}(x)$. Each neural network has a hidden dimension of 8, SiLU activation functions, and a depth of 2, 3, or 4, respectively.

where f_θ is the NCDE's vector field. Assuming $f_\theta(\cdot)a$ is infinitely differentiable, then $f_\theta(\cdot)a$ is an element of the Lie algebra $C^\infty(\mathbb{R}^{d_h}, \mathbb{R}^{d_h})$ and from Definition 3.3,

$$[f_\theta(\cdot)a, f_\theta(\cdot)b] = J_{f_\theta(\cdot)b} f_\theta(\cdot)a - J_{f_\theta(\cdot)a} f_\theta(\cdot)b. \tag{4.56}$$

Calculating (4.55) requires a basis for $\mathfrak{L}^N(\mathbb{R}^{d_X})$, the space where the depth-N truncated log-signature of the input path lives. Let $\{e_j\}_{j=1}^{d_X}$ be the usual basis of $\mathbb{R}^{d_X}$. A choice of basis for $\mathfrak{L}^N(\mathbb{R}^{d_X})$ is a Hall basis, denoted $\{\hat{e}_k\}_{k=1}^{\beta(v,N)}$, which is a specific subset of up to the $(N-1)^{\text{th}}$ iterated Lie brackets of $\{e_j\}_{j=1}^{d_X}$ [Hall 1950]. Rewriting (4.29) using a Hall basis,

$$\bar{f}_\theta(h_s) \frac{\log(S^N(X)_{[r_i, r_{i+1}]})}{r_{i+1} - r_i} = \sum_{k=1}^{\beta(v,N)} \lambda_k \bar{f}_\theta(h_s)\hat{e}_k, \tag{4.57}$$

where λ_k is the term in the scaled log-signature corresponding to the basis element $\hat{e}_k$. Since each $\hat{e}_k$ can be written as iterated Lie brackets of $\{e_j\}_{j=1}^{d_X}$, it is possible to replace $\bar{f}_\theta(\cdot)\hat{e}_k$ with the iterated Lie brackets of $f_\theta(\cdot)e_i$ using (4.54) and (4.55). Each $f_\theta(\cdot)e_i : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}$ is a vector field defined by the i^{th} column of the neural network's output. Hence, $g_{\theta,X}$ can be evaluated at a point using iterated Jacobian-vector products (JVPs) of f_θ.

4.4.4 Computational Cost

When the signature truncation depth N is greater than 1, NRDEs and Log-NCDEs incur an additional computational cost for each evaluation of the vector field, which we now quantify. Assume that an NCDE, NRDE, and Log-NCDE are all using an identical FCNN as their vector field, except for the dimension of the final layer in the NRDE. Let m and n_h be the depth and dimension of the FCNN's hidden layers, respectively, and d_h and d_X be the dimensions of h_t and X_t from (4.20). Let $\beta(d_X, N) = \dim \mathfrak{L}^N(\mathbb{R}^{d_X}) = \mathcal{O}(d_X^N)$. Letting F_{x} be the number of FLOPs required to evaluate model x's vector field,

$$\begin{aligned} F_{\text{NCDE}} &= 2d_h n_h + 2(m-2)n_h^2 + 2d_h d_X n_h, \\ F_{\text{NRDE}} &= 2d_h n_h + 2(m-2)n_h^2 + 2d_h \beta(d_X, N) n_h. \end{aligned} \tag{4.58}$$

The NRDE expression follows because the final layer outputs an element of $\mathbb{R}^{d_h \times \beta(d_X, N)}$ rather than $\mathbb{R}^{d_h \times d_X}$.

For Log-NCDEs, an exact closed-form FLOP count depends on the implementation

of the iterated Lie brackets. For fixed N, the computational cost of evaluating the Log-NCDE vector field is $\mathcal{O}(\beta(d_X, N))$ with respect to d_X, since the vector field is constructed from iterated Lie brackets corresponding to a Hall basis of $\mathfrak{L}^N(\mathbb{R}^{d_X})$. The constant hidden in this asymptotic notation grows rapidly with N, since higher-order Lie brackets require iterated JVPs of vector fields that are themselves defined recursively through lower-order JVPs. For our implementation of the $N = 2$ case, the exact expression

$$F_{\text{Log-NCDE}} = 3d_X F_{\text{NCDE}} \tag{4.59}$$

is obtained, as the number of FLOPs required to calculate a JVP is 3 times that of evaluating f_θ and d_X JVPs of f_θ are needed to evaluate (4.57) [Griewank et al. 2008, Chapter 4].

Therefore, Log-NCDEs and NRDEs have the same asymptotic computational complexity with respect to d_X for fixed N. Furthermore, each JVP is evaluated at the same point h_s. This allows the Log-NCDE vector field on high-dimensional time series to be evaluated in parallel. This computational advantage is demonstrated empirically in Section 4.4.7.

4.4.5 Limitations

In this thesis, we restrict attention to Log-NCDEs based on depth-1 and depth-2 Log-ODE approximations. This reflects two main limitations. First, there are currently no theoretical results explicitly bounding the $\text{Lip}(\gamma)$ norm of a neural network for $\gamma > 2$. Second, as discussed in Section 4.4.4, the cost of evaluating $g_{\theta,X}$ grows rapidly with the truncation depth N. In particular, at $N = 2$ this cost scales quadratically in the input dimension d_X, which in practice restricts depth-2 Log-NCDEs to moderate-dimensional inputs. The widest time series considered in our experiments has dimension $d_X = 64$. Input dimensions in the low hundreds are likely still feasible, whereas dimensions on the order of 10^3 are likely to be computationally prohibitive.

A more general limitation of NCDEs is that their hidden dynamics must be solved sequentially in time, which prevents parallelisation across time steps. This is in contrast to structured state-space models, whose linear dynamics admit explicit flows that can be composed in parallel across time [Gu et al. 2022a]. This limitation serves as the primary motivation for the work of Chapter 5.

4.4.6 Experiments

Baseline Methods

Log-NCDEs are compared against six models, which represent the state-of-the-art for a range of deep learning approaches to time series modelling. Four of these models are stacked recurrent models, whose general architecture is based on the official implementation of S5 located at https://github.com/lindermanlab/S5 [Smith et al. 2023]. A recurrent block consists of a batch or layer normalisation [Ioffe et al. 2015; Ba et al. 2016], a recurrent layer, a gated linear unit (GLU) [Dauphin et al. 2017], dropout with rate 0.1 [Srivastava et al. 2014], and a skip connection. A full model consists of a linear encoder, a number of stacked recurrent blocks, and a final linear layer. The four different recurrent layers considered are the LRU [Orvieto et al. 2023], S5 [Smith et al. 2023], S6, and Mamba, where S6 refers to the selective state-space recurrence introduced by Gu et al. [2024] and Mamba refers to the combination of a gated MLP, convolution, and S6 recurrence [Gu et al. 2024]. S5 and LRU use batch normalisation [Ioffe et al. 2015], whereas S6 and Mamba use layer normalisation [Ba et al. 2016].

The other two baseline models are continuous models; an NCDE using a Hermite cubic spline with backward differences as the interpolation and an NRDE [Kidger et al. 2020; Morrill et al. 2021]. NCDEs, NRDEs, and Log-NCDEs use a single linear layer as ξ_ϕ. NCDEs and NRDEs use FCNNs as their vector fields configured in the same way as their original papers [Kidger et al. 2020; Morrill et al. 2021]. NCDEs use ReLU activation functions for the hidden layers and a final activation function of tanh. NRDEs use the same, but move the tanh activation function to be before the final linear layer in the FCNN. Log-NCDEs use a FCNN with SiLU activation functions for the hidden layers and a final activation function of tanh. NRDEs and Log-NCDEs take their intervals $r_{i+1} - r_i$ to be a fixed number of observations, referred to as the Log-ODE step.

Toy Dataset

We construct a toy dataset of 100,000 time series with 6 channels and 100 regularly spaced samples each. For every time step, the change in each channel is sampled independently from the discrete probability distribution with density

$$p(n) = \int_{n-0.5}^{n+0.5} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \mathrm{d}x, \tag{4.60}$$

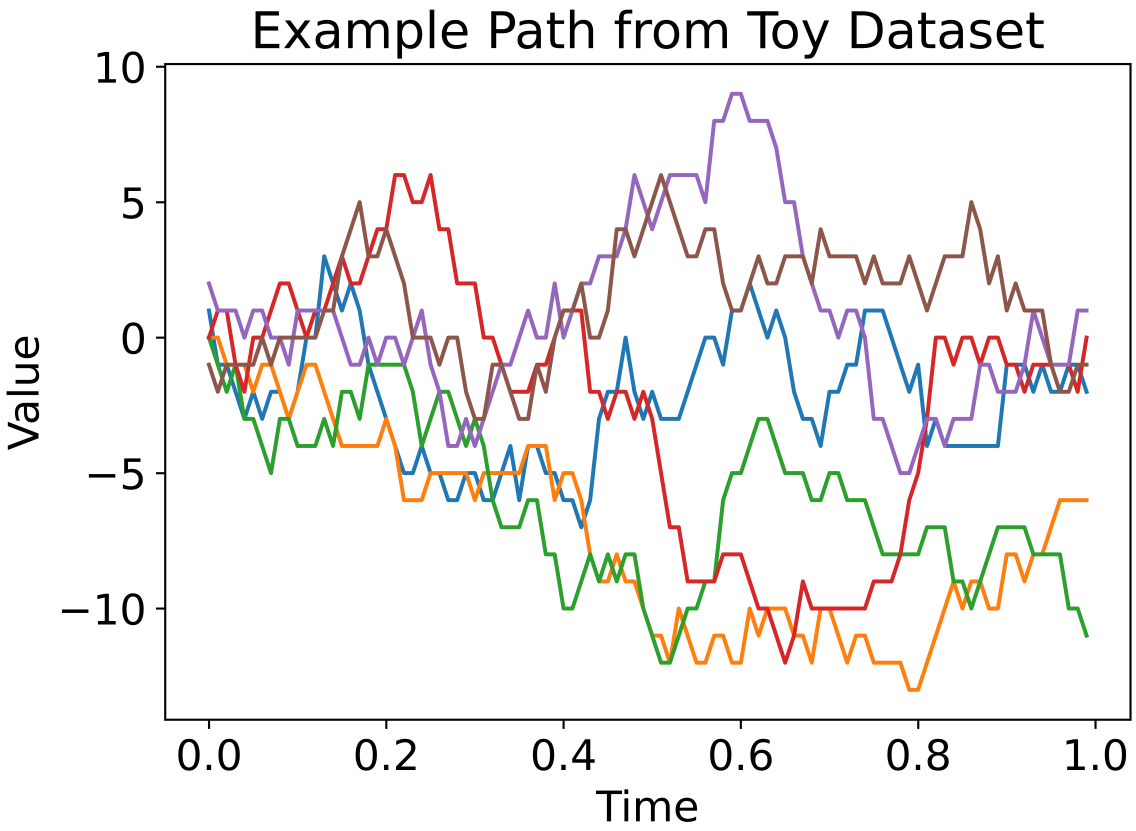


Figure 4.6: An example path from the toy dataset, where each colour represents a channel in the path.

where $n \in \mathbb{Z}$. In other words, the change in a channel at each time step is a sample from a standard normal distribution rounded to the nearest integer. Figure 4.6 is a plot of a sample path from the toy dataset.

We consider four different binary classifications on the toy dataset. Each classification is a specific term in the signature of the path which depends on a different number of channels.

1. Was the change in the third channel, $\int_0^1 \mathrm{d}X_s^3$, greater than zero?
2. Was the area integral of the third and sixth channels, $\int_0^1 \int_0^u \mathrm{d}X_s^3 \mathrm{d}X_u^6$, greater than zero?
3. Was the volume integral of the third, sixth, and first channels, $\int_0^1 \int_0^v \int_0^u \mathrm{d}X_s^3 \mathrm{d}X_u^6 \mathrm{d}X_v^1$, greater than zero?
4. Was the 4D volume integral of the third, sixth, first, and fourth channels, $\int_0^1 \int_0^w \int_0^v \int_0^u \mathrm{d}X_s^3 \mathrm{d}X_u^6 \mathrm{d}X_v^1 \mathrm{d}X_w^4$, greater than zero?

Each task is asking the model to check the sign of a specific term in the signature of the input path.

On the toy dataset, all models use a hidden dimension of 64 and Adam with a learning rate of 0.0003 [Kingma et al. 2015]. LRU, S5, S6, and Mamba use 6 blocks and S5, S6, and Mamba use a state dimension of 64. S5 uses 2 initialisation blocks and Mamba uses a convolution dimension of 4 and an expansion factor of 2. NCDEs, NRDEs, and Log-NCDEs use a FCNN with width 128 and depth 3 as their vector field. Furthermore, all NCDEs use Heun as their differential equation solver with a fixed stepsize of 0.01 [Heun 1900; Atkinson et al. 2009]. NRDEs and Log-NCDEs use a Log-ODE step of 4 and a signature truncation depth of 2. Log-NCDEs do not use any $\mathrm{Lip}(\gamma)$ regularisation, i.e. $\lambda = 0$ in (4.53).

UEA Multivariate Time Series Classification Archive

The models are evaluated on six datasets from the UEA multivariate time series classification archive (UEA-MTSCA)[1] [Bagnall et al. 2018]. These six datasets were chosen via the following two criteria. First, only datasets with more than 200 total time series were considered. Second, the six datasets with the most observations were chosen, as datasets with many observations have previously proved challenging for deep learning approaches to time series modelling. Table 4.1 provides details on the dimension, number of observations, and number of classes for the datasets chosen from the UEA-MTSCA for the experiments conducted in this thesis. Following [Morrill et al. 2021], the original train and test cases are combined and resplit into new random train, validation, and test cases using a $70 : 15 : 15$ split.

Hyperparameters for all models are found using a grid search over the validation accuracy on a fixed random split of the data. Having fixed their hyperparameters, models are compared on their average test set accuracy over five different random splits of the data. In order to compare models on their average GPU memory and runtime, 1000 steps of training are run on an NVIDIA RTX 4090. Each training step consists of a forward pass, loss computation, backward pass, and parameter update. The average runtime is estimated by combining the time for 1000 training steps with the average total number of training steps from the five runs over the random data

[1] As of November 10^{th} 2025, the EigenWorms dataset available for download at https://timeseriesclassification.com has 23 duplicated time series, which were removed for the experiments in this thesis.

Table 4.1: A summary of the subset of the UEA-MTSCA datasets used in this thesis.

Dataset	Dimension	Number of Observations	Classes
EigenWorms	6	17984	5
EthanolConcentration	3	1751	4
Heartbeat	61	405	2
MotorImagery	64	3000	2
SelfRegulationSCP1	6	896	2
SelfRegulationSCP2	7	1152	2

splits.

PPG-DaLiA

PPG-DaLiA is a multivariate time series regression dataset, where the aim is to predict a person's heart rate using data collected from a wrist-worn device [Reiss et al. 2019]. The dataset consists of fifteen individuals with around 150 minutes of recording each at a maximum sampling rate of 128 Hz. There are six channels: blood volume pulse, electrodermal activity, body temperature, and three-axis acceleration. For each individual, the data is split into training, validation, and test sets using a 70:15:15 split. After splitting, a sliding window is applied to each subset to convert the long continuous recordings into shorter overlapping time series. Specifically, each window contains 49920 consecutive samples, and successive windows are offset by 4992 samples.

Hyperparameters are found using the same method as for the UEA-MTSCA, but with validation mean squared error and slightly different hyperparameter choices given the high number of observations. Having fixed their hyperparameters, models are compared on their average mean squared error over five different runs on the same fixed data split.

Hyperparameter Optimisation

The seven models considered in the experiments fall into two groups. The first group consists of the stacked recurrent models LRU, S5, S6, and Mamba. The second group consists of the continuous-time models NCDE, NRDE, and Log-NCDE. Tables 4.2 and 4.3 list the hyperparameters optimised over for these two groups on the UEA-

MTSCA and PPG-DaLiA experiments.

For each dataset and model, hyperparameters were selected by grid search on a fixed training-validation split. For the UEA-MTSCA datasets, each configuration was trained using cross-entropy loss and compared using validation accuracy. For PPG-DaLiA, each configuration was trained using mean squared error loss and compared using validation mean squared error. Training used early stopping based on the corresponding validation metric, and the checkpoint with the best validation performance was used for the final test evaluation. After hyperparameter selection, the chosen configuration was used in the evaluation protocols described above, namely five random data splits for the UEA-MTSCA datasets and five random seeds on a fixed split for PPG-DaLiA.

All models and experiments used Adam as their optimiser [Kingma et al. 2015]. A batch size of 32 was used throughout, except for the stacked recurrent models on PPG-DaLiA, where a batch size of 4 was required due to memory constraints. NCDEs, NRDEs, and Log-NCDEs used Heun as their differential equation solver with fixed stepsize

$$\frac{1}{\max\{500, 1 + (\text{time series length}/\text{Log-ODE step})\}}, \tag{4.61}$$

with Log-ODE step equal to 1 for NCDEs. Additionally, Log-NCDEs scaled down their initial FCNN parameters by a factor of 1000 to reduce the initial Lip(2) norm of the vector field.

Table 4.2: Hyperparameters selected by the optimisation for LRU, S5, S6, and Mamba on the UEA-MTSCA datasets and PPG-DaLiA dataset. The following abbreviations are used: EigenWorms (EW), EthanolConcentration (EC), Heartbeat (HB), MotorImagery (MI), SelfRegulationSCP1 (SCP1), SelfRegulationSCP2 (SCP2), and PPG-DaLiA (PPG). A ✗ denotes that the hyperparameter is not applicable to that model.

Hyperparameters	Options	Method			
		LRU	S5	S6	Mamba
Learning Rate	10^{-3}	EW, MI, SCP1, SCP2, PPG	HB, MI, SCP1, PPG	EW, HB, MI, SCP2	EW, EC, PPG
	10^{-4}	HB	EW, SCP2	SCP1, PPG	HB, SCP2
	10^{-5}	EC	EC	EC	MI, SCP1
Include Time	True	EC, HB, SCP2	EW, EC, SCP2, PPG	EC, HB, MI, PPG	EW, EC, SCP2
	False	EW, MI, SCP1, PPG	HB, MI, SCP1	EW, SCP1, SCP2	HB, MI, SCP1, PPG
Hidden Dimension	16	MI	MI, SCP2, PPG	EW, EC, HB, MI, SCP2	EW
	64	EW, EC, SCP1, SCP2	EW	SCP1, PPG	EC, HB, SCP2
	128	HB, PPG	EC, HB, SCP1		MI, SCP1, PPG
Number of Layers	2	HB, SCP1, SCP2	EW, EC, SCP2	SCP1, SCP2, PPG	MI, SCP1
	4	EW	HB	EW, EC, HB, MI	EC, HB, PPG
	6	EC, MI, PPG	MI, SCP1, PPG		EW, SCP2
State Dimension	16	EC, SCP1, SCP2	EW, EC, HB, SCP1	EC, HB, SCP1	SCP1
	64	EW	MI, SCP2, PPG	EW, PPG	EW, MI, SCP2, PPG
	256	HB, MI, PPG		MI, SCP2	EC, HB
S5 Initialisation Blocks	2	✗	SCP2, PPG	✗	✗
	4	✗	HB, MI	✗	✗
	8	✗	EW, EC, SCP1	✗	✗
Convolution Dimension	2	✗	✗	✗	EW, HB, SCP2
	3	✗	✗	✗	MI, PPG
	4	✗	✗	✗	EC, SCP1
Expansion Factor	1	✗	✗	✗	EW, MI, SCP1
	2	✗	✗	✗	SCP2, PPG
	4	✗	✗	✗	EC, HB

Table 4.3: Hyperparameters selected by the optimisation for NCDE, NRDE, and Log-NCDE on the UEA-MTSCA datasets and PPG-DaLiA dataset. Given the length of each time series in the PPG-DaLiA dataset, different choices were considered for the Log-ODE depth and step, which are shown here in red. The following abbreviations are used: EigenWorms (EW), EthanolConcentration (EC), Heartbeat (HB), MotorImagery (MI), SelfRegulationSCP1 (SCP1), SelfRegulationSCP2 (SCP2), and PPG-DaLiA (PPG). A ✗ denotes that the hyperparameter is not applicable to that model.

Hyperparameters	Options	Method		
		NCDE	NRDE	Log-NCDE
Learning Rate	10^{-3}	EW, EC, HB, MI, SCP2, PPG	EW, EC, HB, SCP1, PPG	EW, HB, MI, PPG
	10^{-4}	SCP1	MI, SCP2	EC, SCP1, SCP2
	10^{-5}			
Include Time	True	EW, EC, HB, MI, PPG	EC, SCP1, SCP2, PPG	EW, EC, HB, PPG
	False	SCP1, SCP2	EW, HB, MI	MI, SCP1, SCP2
Hidden Dimension	16	MI, PPG		HB, MI
	64		EW, EC, HB, SCP1	EC, SCP1
	128	EW, EC, HB, SCP1, SCP2	MI, SCP2, PPG	EW, SCP2, PPG
Vector Field (Depth, Width)	(2, 32)			EW, SCP2
	(3, 64)		EW	EC, MI, PPG
	(3, 128)	EW	HB	HB, SCP1
	(4, 128)	EC, HB, MI, SCP1, SCP2, PPG	EC, MI, SCP1, SCP2, PPG	
Log-ODE (Depth, Step)	(1, 1)	✗	EC, MI, SCP1, SCP2	EC
	(2, 2)	✗	HB	HB
	(2, 4)	✗	EW	SCP2
	(2, 8)	✗		
	(2, 12)	✗		EW
	(2, 16)	✗		MI, SCP1
	(1, 10)	✗	PPG	PPG
	(2, 10)	✗		
	(2, 100)	✗		
	(2, 1000)	✗		
Regularisation λ	10^{-3}	✗	✗	EW, MI, SCP2
	10^{-6}	✗	✗	EC, HB
	0.0	✗	✗	SCP1, PPG

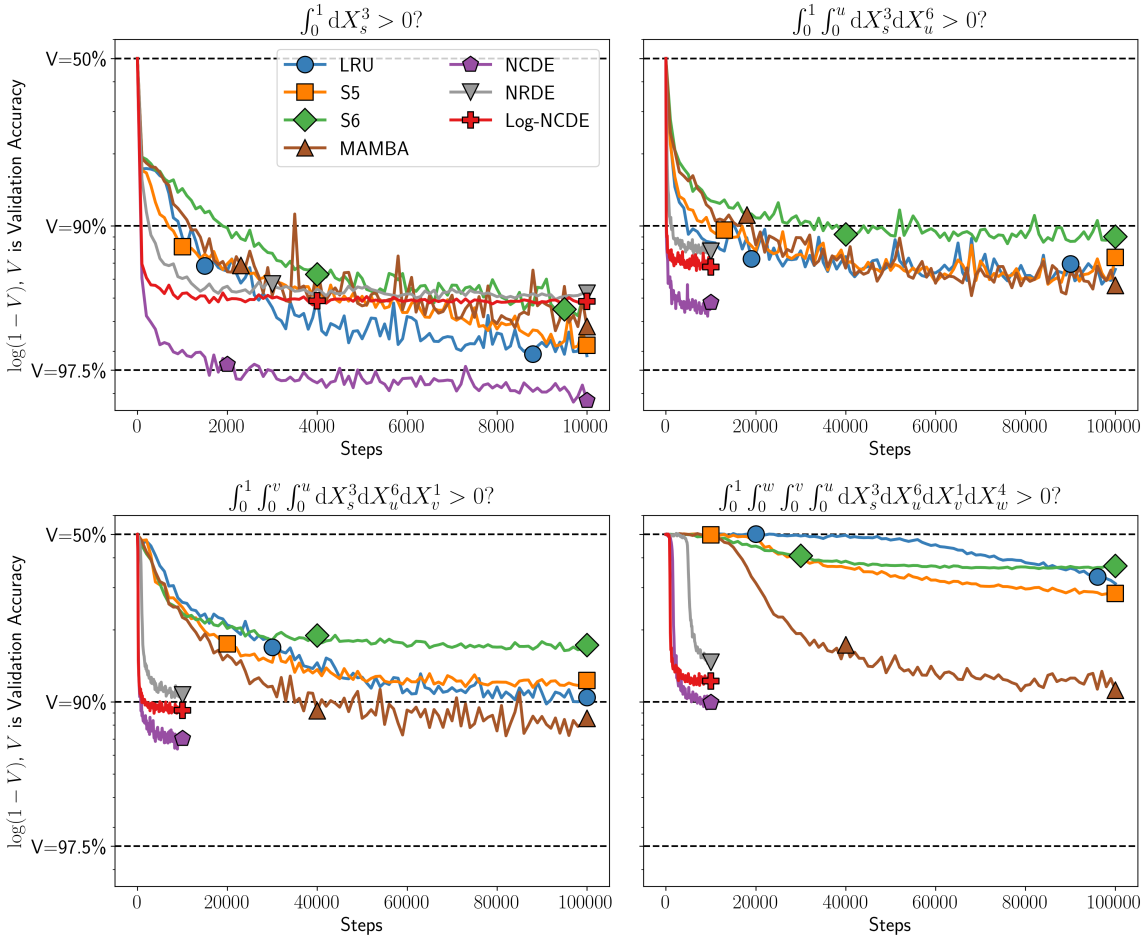


Figure 4.7: Validation accuracy against number of steps for LRU, S5, S6, Mamba, NCDE, NRDE, and Log-NCDE on the four different classifications considered for the toy dataset.

4.4.7 Results

Toy Dataset

Figure 4.7 compares the performance of the models on the four different toy dataset classifications. As expected, given that the classifications considered are solutions to CDEs, NCDEs are the best performing model. Since NRDEs and Log-NCDEs are fixed to $r_{i+1} - r_i$ being 4 observations and $N = 2$, they are both approximations of a CDE. Notably, Log-NCDEs consistently outperform NRDEs, providing empirical evidence that NRDEs do not always accurately learn the Lie bracket structure of $\bar{f}_\theta$. All of the stacked recurrent models perform well when the label depends on one or two channels. However, their performance begins to decrease for three channels, and only Mamba performs well when the label depends on four channels.

UEA-MTSCA

Table 4.4 reports the mean and standard deviation of each model's test set accuracy over five data splits. Among the stacked recurrent models, LRU, S5, and S6 achieve similar average accuracies overall. However, Mamba attains the lowest average accuracy of all seven methods. Since S6 achieves performance in line with S5, Mamba's weaker results do not appear to be caused by the selective recurrence itself. Instead, they may reflect the effect of Mamba's additional architectural components, particularly its short convolution, which places greater weight on local information and may therefore be less well suited to classification tasks that reward preserving information across the full time series.

NCDEs and NRDEs have similar average accuracies overall, although NRDEs perform notably better on EigenWorms, the dataset with the most observations. However, NRDEs are still outperformed in average accuracy by the stacked recurrent models LRU, S5, and S6. In contrast, Log-NCDEs achieve the best average accuracy and the best average rank across the six datasets. Compared to NRDEs, they attain an equal or higher average accuracy on all six datasets and a lower standard deviation on four datasets.

A Friedman test across the six datasets and seven methods did not detect a statistically significant difference in performance among the models at the 5% significance level, with $\chi^2 = 9.69$, $df = 6$, and $p = 0.138$ [Demšar 2006]. Since Log-NCDEs are a direct modification of NCDEs and NRDEs, we additionally performed one-sided Wilcoxon signed-rank tests comparing Log-NCDEs against these two baselines, with Holm correction across the two comparisons [Wilcoxon 1945]. At the 5% significance level, these tests indicated that Log-NCDEs significantly outperformed both NCDEs and NRDEs, with adjusted $p = 0.0313$ in each case. These results suggest that incorporating Lie bracket information can improve predictive performance.

PPG-DaLiA

Table 4.5 contains the average and standard deviation of each model's test set mean squared error on the PPG-DaLiA dataset. In contrast to the UEA-MTSCA experiments, Mamba is the best performing stacked recurrent model on PPG-DaLiA and clearly outperforms S6. This suggests that Mamba's additional architectural components, such as its short convolution, are beneficial for this regression task. This

Table 4.4: Mean and standard deviation of test set accuracy over five data splits on a subset of the UEA-MTSCA. The best performing model is highlighted in bold and the second best is underlined. The average accuracy and average rank are also reported.

Dataset	Method						
	LRU	S5	S6	Mamba	NCDE	NRDE	Log-NCDE
EigenWorms	**87.8 ± 2.8**	81.1 ± 3.7	85.0 ± 16.1	70.9 ± 15.8	75.0 ± 3.9	83.9 ± 7.3	85.6 ± 5.1
EthanolConcentration	21.5 ± 2.1	24.1 ± 4.3	26.4 ± 6.4	27.9 ± 4.5	29.9 ± 6.5	25.3 ± 1.8	**34.4 ± 6.4**
Heartbeat	**78.4 ± 6.7**	77.7 ± 5.5	76.5 ± 8.3	76.2 ± 3.8	73.9 ± 2.6	72.9 ± 4.8	75.2 ± 4.6
MotorImagery	48.4 ± 5.0	47.7 ± 5.5	51.3 ± 4.7	47.7 ± 4.5	49.5 ± 2.8	47.0 ± 5.7	**53.7 ± 5.3**
SelfRegulationSCP1	82.6 ± 3.4	**89.9 ± 4.6**	82.8 ± 2.7	80.7 ± 1.4	79.8 ± 5.6	80.9 ± 2.5	83.1 ± 2.8
SelfRegulationSCP2	51.2 ± 3.6	50.5 ± 2.6	49.9 ± 9.5	48.2 ± 3.9	53.0 ± 2.8	**53.7 ± 6.9**	**53.7 ± 4.1**
Av.	61.7	61.8	62.0	58.6	60.2	60.6	**64.3**
Av. Rank	3.5	4.0	3.5	5.5	4.5	4.9	**2.1**

Table 4.5: Mean and standard deviation of test set mean squared error over five runs with different random seeds on the PPG-DaLiA dataset.

Model	MSE ($\times 10^{-2}$)
LRU	12.17 ± 0.49
S5	12.63 ± 1.25
S6	12.88 ± 2.05
Mamba	10.65 ± 2.20
NCDE	13.54 ± 0.69
NRDE	9.90 ± 0.97
Log-NCDE	$\mathbf{9.56 \pm 0.59}$

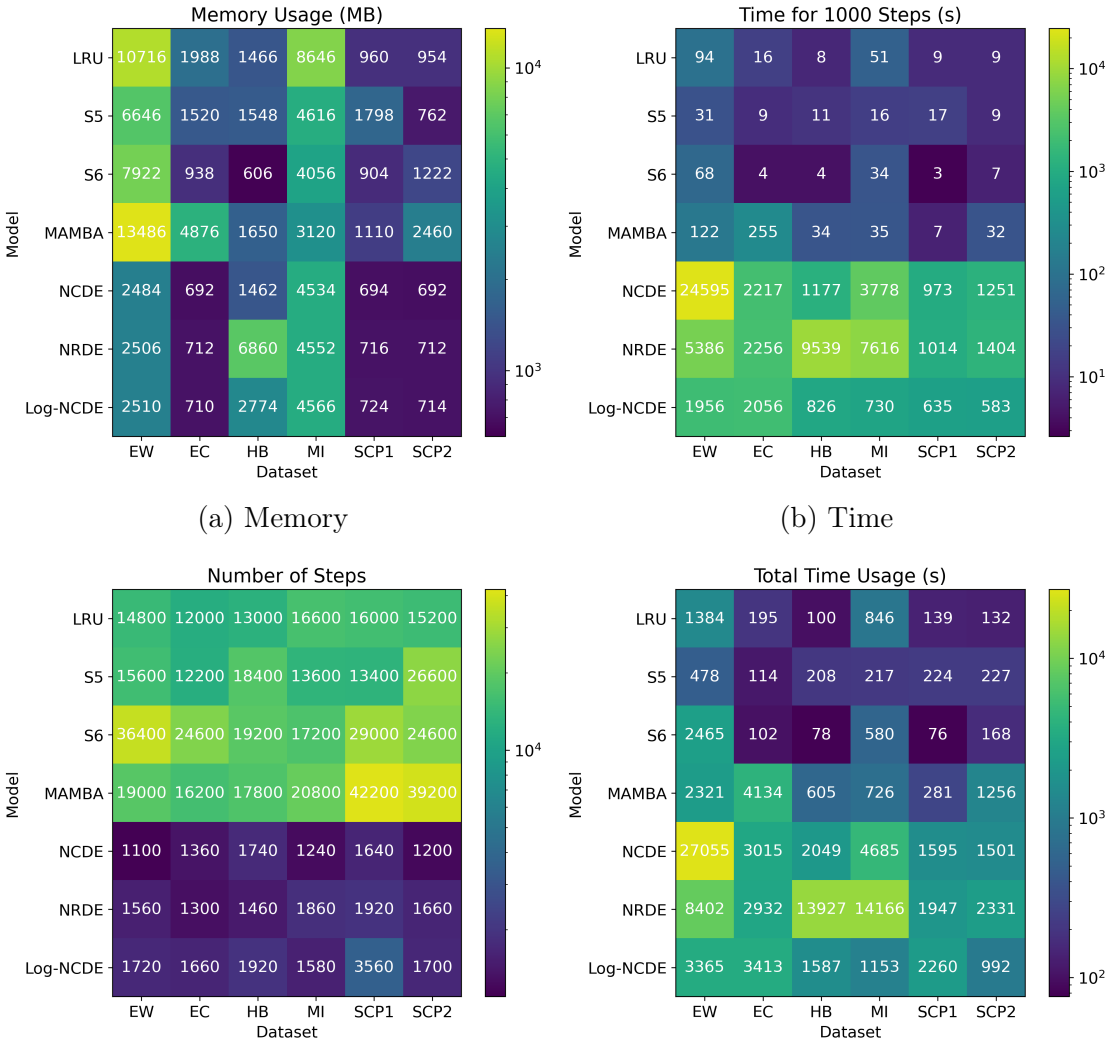
aligns with the intuition that heart-rate prediction depends heavily on recent observations. Among the neural differential equation models, both NRDEs and Log-NCDEs substantially outperform the NCDE, indicating that the Log-ODE based models are better suited to handling very long sequences. Log-NCDEs still achieve the best overall performance, obtaining the lowest average test set mean squared error and the second lowest standard deviation.

Memory and Time

Models are compared on their average GPU memory usage and runtime for the UEA-MTSCA datasets. In order to compare the models, 1000 steps of training were run on an NVIDIA RTX 4090 with each model using the hyperparameters obtained from the hyperparameter optimisation, as detailed in Tables 4.2 and 4.3. In addition to the time for 1000 steps and GPU memory usage, shown in Figures 4.8a and 4.8b, the average number of total training steps taken to produce the results in Table 4.4 is recorded in Figure 4.8c. Combining the results for time per 1000 training steps and the total number of training steps gives an approximation of the total runtime on the same hardware, and these results are shown in Figure 4.8d. The average GPU memory and runtime across the six datasets is given in Table 4.6.

Although the time per training step is lower for stacked recurrent models than NCDEs, NRDEs, or Log-NCDEs, they also require more training steps to converge. Additionally, NCDEs, NRDEs, and Log-NCDEs require less GPU memory. The largest contributors to the average runtime of NCDEs are the datasets with the most observations, EigenWorms and MotorImagery. The positive impact of the Log-ODE method on computational burden is demonstrated empirically by the decrease in runtime achieved by NRDEs and Log-NCDEs on EigenWorms when using a depth-2 Log-ODE method. When a depth-1 Log-ODE method is used, such as NRDEs on MotorImagery, the same decrease is not observed.

Section 4.4.4 demonstrated that Log-NCDEs and NRDEs have the same asymptotic computational complexity. However, when using a depth-2 Log-ODE approximation and the same stepsize, NRDEs and Log-NCDEs exhibit different runtimes on Heartbeat, a high-dimensional dataset. This difference is partly explained by the model's having different optimal hyperparameter choices, but even when using identical hyperparameters to the NRDE, Log-NCDE's time per 1000 training steps is 1673



(a) Memory

(b) Time

(c) Number of steps

(d) Total time

Figure 4.8: Memory, time per 1000 training steps, number of steps, and approximate total time for each model and dataset from the UEA-MTSCA on an NVIDIA RTX 4090. The following abbreviations are used: EigenWorms (EW), EthanolConcentration (EC), Heartbeat (HB), MotorImagery (MI), SelfRegulationSCP1 (SCP1), and SelfRegulationSCP2 (SCP2).

Table 4.6: Average GPU memory and runtime for each model over the six datasets from the UEA-MTSCA experiments.

Model	Av. GPU Mem. (MB)	Av. runtime (s)
LRU	4121.67	466.09
S5	2815.00	244.78
S6	2608.00	578.15
Mamba	4450.33	1553.83
NCDE	1759.67	6649.91
NRDE	2676.33	7284.20
Log-NCDE	1999.67	2128.32

seconds, whereas NRDE's is 9539 seconds. The remaining difference is due to being able to calculate the JVPs of f_θ in parallel, as discussed in Section 4.4.4. If instead the JVPs are calculated recurrently, then Log-NCDEs time per 1000 training steps increases to 17045 seconds.

From a practical standpoint, the Log-ODE method improves the computational viability of NCDE-style models, but it does not eliminate the underlying scalability challenge. Log-NCDEs are markedly faster than NCDEs and NRDEs on the longest datasets, use relatively little GPU memory, and still achieve strong empirical performance. They also naturally accommodate irregularly sampled and over-sampled data, unlike the stacked recurrent baselines considered here. However, they remain much slower than those recurrent models, which limits their suitability for rapid experimentation, large-scale hyperparameter tuning, and settings requiring very long training runs.

4.5 Conclusion

This chapter showed that the Log-ODE method provides a principled way to approximate NCDE dynamics during training. It built on the groundwork laid by NRDEs and used the $\mathrm{Lip}(\gamma)$ theory developed in Chapter 3 to ensure that the resulting models are well defined. Empirically, Log-NCDEs matched or exceeded the performance of NCDEs and NRDEs on all six real-world multivariate time series classification datasets considered, while reducing the average runtime by more than a factor of 3.

Log-NCDEs also achieved the best average test accuracy and the best average rank

among all seven models in Table 4.4. However, their computational cost remains substantial. The average runtime of Log-NCDEs is still nearly an order of magnitude larger than that of the fastest baseline, S5. The gap is even more striking in the time per 1000 training steps, which is 1131 seconds for Log-NCDEs and only 16 seconds for S5. Thus, although Log-NCDEs are more practical than NCDEs and NRDEs, they remain too slow for genuinely large-scale applications.

The reason for this remaining gap is the nature of the hidden-state dynamics. Log-NCDEs still require the numerical solution of a non-linear controlled differential equation during each forward pass. By contrast, S5 is based on linear dynamics whose flow can be written in closed form on each interval. This removes the need for an expensive non-linear solve and allows the hidden state over a sequence of length n to be computed in only $\mathcal{O}(\log n)$ parallel steps via an associative scan.

Motivated by this observation, Chapter 5 introduces Linear NCDEs, where the vector fields are constrained to be linear in the hidden state. This restriction yields closed-form solutions for the dynamics on each interval, making parallel-in-time computation possible while retaining the theoretical expressivity and continuous-time structure of NCDEs. Furthermore, Log-NCDEs naturally translate to this linear setting, where the Log-ODE method continues to provide substantial empirical runtime benefits.

Chapter 5

Linear Neural Controlled Differential Equations

Aren't linear CDEs expressive enough?

—Massimiliano Gubinelli (2023)

5.1 Introduction

Chapter 2 developed the continuous-time mathematical framework underlying this thesis. Paths are the fundamental objects, signatures and log-signatures provide principled summaries of path segments, CDEs describe how paths influence the state of a system, and the Log-ODE method provides an efficient and accurate approximation to the solution of a CDE. Building on the regularity theory developed in Chapter 3, Chapter 4 translated this framework into machine learning through Log-NCDEs, showing that continuous-time models can achieve performance comparable to strong discrete baselines on regularly sampled datasets, whilst naturally being able to handle irregularly sampled and over-sampled time series. However, Chapter 4 also showed that the non-linear hidden-state dynamics remain a major computational bottleneck. Even with the Log-ODE method, Log-NCDEs are still substantially slower than the strongest stacked recurrent baselines. The aim of this chapter is to retain the empirical performance and continuous-time advantages of NCDEs while replacing their non-linear dynamics with a more scalable model class.

From the perspective of CDE theory, the linear case is the natural place to look.

Linear CDEs admit explicit solutions, as discussed in Section 2.4.3, and the truncated signature is itself the solution of a linear CDE, as shown in Section 2.4.1. Moreover, when equipped with a linear readout, linear CDEs are maximally expressive on the space of time-augmented driving paths with the 1-variation topology, by Corollary 2.28 and the proof of Theorem 4.4. These facts suggest that vector fields which are linear in the hidden state may already be sufficient for time series modelling. Given this, it may seem surprising that Linear NCDEs were developed after their non-linear counterparts. The epigraph, a question posed by Prof. Massimiliano Gubinelli after a presentation on Log-NCDEs, captures this sentiment. Historically, non-linear NCDEs were explored first in order to maximise modelling capacity and to align more closely with non-linear recurrent neural networks. Chapter 4 shows that this choice comes at a significant computational cost.

In this chapter, we show that constraining an NCDE's vector field to be linear in the hidden state leads to the Log-ODE method producing closed-form flows on each interval. This removes the need for a differential equation solver, enables parallel-in-time computation, and directly addresses the scalability limitations identified in Chapter 4. We also show that Linear NCDEs retain the theoretical expressivity of NCDEs. Finally, we demonstrate empirically that, when combined with the Log-ODE method, Linear NCDEs match the performance of Log-NCDEs while reducing the time per training step by up to two orders of magnitude. This makes continuous-time machine learning practical at scales that were previously infeasible.

5.2 Linear NCDEs

5.2.1 Introduction

Definition 5.1 (Linear NCDE [Cirone et al. 2024; Walker et al. 2025])**.** *Let $\mathcal{X}(d)$ denote the space of bounded-variation d-dimensional paths on the interval $[t_0, t_n]$ which all begin at the same point and contain time as a channel. Let $\omega : \mathcal{X}(d_X) \to \mathcal{X}(d_\omega)$ be a continuous function on paths, with the shorthand $\omega^X = \omega(X)$. Let $L_\theta^1 \in \mathbb{R}^{d_h \times d_X}$, $A_\theta \in \mathbb{R}^{d_h \times d_\omega \times d_h}$, and $L_\theta^2 \in \mathbb{R}^{d_y \times d_h}$ be trainable parameters. Then a Linear NCDE is*

defined by

$$\begin{aligned} h_{t_0} &= L_\theta^1 X_{t_0}, \\ h_t &= h_{t_0} + \int_{t_0}^{t} A_\theta h_s \, \mathrm{d}\omega_s^X, \\ y_t &= L_\theta^2 h_t. \end{aligned} \tag{5.1}$$

There are two core differences between NCDEs and Linear NCDEs. First, the driving path is a function of the interpolated data, $\omega^X = \omega(X)$, rather than the interpolation X itself. Second, the vector field is linear in the hidden state,

$$h_t = h_{t_0} + \int_{t_0}^{t} A_\theta h_s \, \mathrm{d}\omega_s^X = h_{t_0} + \int_{t_0}^{t} \sum_{i=1}^{d_\omega} A_\theta^i h_s \, \mathrm{d}\omega_s^{X,i}, \tag{5.2}$$

where $A_\theta^i \in \mathbb{R}^{d_h \times d_h}$ are the parameters corresponding to the i^{th} channel of the driving path $\omega_s^{X,i}$. As discussed in Section 2.4.3, (5.2) admits an explicit solution in terms of the signature of the driving path ω^X. Although exact, this representation is not directly practical for computation. However, when ω^X is piecewise linear, the flow of (5.2) admits an explicit solution. More generally, the approximate flows generated by the Log-ODE method also admit explicit solutions. This is the key structural property that makes Linear NCDEs more scalable than NCDEs, since an explicit expression for the flow determines the evolution of any initial hidden state, rather than requiring each trajectory to be obtained by numerically solving a non-linear differential equation.

5.2.2 Computing the Flow

Let ω^X be piecewise-linear on the grid $t_0 < \cdots < t_n$. On each subinterval $[t_j, t_{j+1}]$, the dynamics are constant, so the update can be computed exactly,

$$\tilde{h}_{t_{j+1}} = \exp\left(\sum_{i=1}^{d_\omega} (\omega_{t_{j+1}}^{X,i} - \omega_{t_j}^{X,i}) A^i \right) \tilde{h}_{t_j}. \tag{5.3}$$

Equation (5.3) does more than simply remove the general-purpose ODE solver from NCDE inference, it represents a fundamental change in approach. Rather than tracking a trajectory $t \mapsto h_t$, we compute the state-transition map from t_j to t_{j+1},

$$\exp\left(\sum_{i=1}^{d_\omega} (\omega_{t_{j+1}}^{X,i} - \omega_{t_j}^{X,i}) A^i \right), \tag{5.4}$$

which is also known as the flow. This shift in perspective allows for the flow over any interval to be solved for using only $\mathcal{O}(\log(n))$ parallel steps by using a parallel

associative scan.

Given a sequence $(x_1, \ldots, x_n)$ and a binary operation $\otimes$, a prefix scan returns the running prefix products

$$y_k = x_1 \otimes \cdots \otimes x_k. \tag{5.5}$$

Given the natural recurrence $y_k = y_{k-1} \otimes x_k$, a scan can be computed in $\mathcal{O}(n)$ work and $\mathcal{O}(n)$ recurrent steps. If the operation is associative,

$$(a \otimes b) \otimes c = a \otimes (b \otimes c), \tag{5.6}$$

then it is possible to compute the prefix scan in only $\mathcal{O}(\log(n))$ parallel steps using a tree-based algorithm [Blelloch 1993]. We demonstrate the approach for $(x_1, \ldots, x_8)$. First you compute a parallel up-sweep,

$$\begin{aligned}
&\textbf{Level 1} : s_1 = x_1 \otimes x_2, \quad s_2 = x_3 \otimes x_4, \quad s_3 = x_5 \otimes x_6, \quad s_4 = x_7 \otimes x_8, \\
&\textbf{Level 2} : t_1 = s_1 \otimes s_2, \quad t_2 = s_3 \otimes s_4, \\
&\textbf{Level 3} : u = t_1 \otimes t_2.
\end{aligned} \tag{5.7}$$

This calculation created a tree, with u as the root, t_1 and t_2 as u's left and right children, and so on. Now, you perform a parallel down-sweep of this tree, with the rule that given incoming carry P, the left child receives P and the right child receives $P \otimes L$, where L is the subtree with the left child as the root. Initialising the root's carry as the identity, $P(u) = e$, and applying the carry rule to our example,

$$\begin{aligned}
\textbf{Level 3}: \quad & P(t_1) = e, && P(t_2) = e \otimes t_1 = t_1, \\
\textbf{Level 2}: \quad & P(s_1) = e, && P(s_2) = e \otimes s_1 = s_1. \\
& P(s_3) = t_1, && P(s_4) = t_1 \otimes s_3, \\
\textbf{Level 1}: \quad & P(x_1) = e, && P(x_2) = e \otimes x_1 = x_1, \\
& P(x_3) = s_1, && P(x_4) = s_1 \otimes x_3, \\
& P(x_5) = t_1, && P(x_6) = t_1 \otimes x_5, \\
& P(x_7) = t_1 \otimes s_3, && P(x_8) = t_1 \otimes s_3 \otimes x_7.
\end{aligned} \tag{5.8}$$

These carries are exactly the exclusive prefixes and the inclusive outputs are $y_k = P(x_k) \otimes x_k$.

For (5.3), the scan elements are the flows (5.4) indexed by j, and the associative binary operation is matrix multiplication. Once the flows over $[t_0, t_j]$ have been obtained for $j = 1, \ldots, n$, a batched matrix-vector multiplication with h_{t_0} yields the

hidden states h_{t_j} for $j = 1, \ldots, n$, so inference can be completed in $\mathcal{O}(\log(n))$ parallel steps. However, a scan over all n observation intervals requires materialising n matrices in $\mathbb{R}^{d_h \times d_h}$. Therefore, when d_h is large, the practical cost can be dominated by GPU memory traffic [Yang et al. 2024b]. The Log-ODE method mitigates this by replacing the fine observation grid with a coarser partition $t_0 = r_0 < \cdots < r_m = t_n$, so that the scan is performed over only $m < n$ interval flows.

An equivalent viewpoint of the approximation (5.3) is applying a depth-1 Log-ODE method to (5.1) on the grid $\{t_j\}_{j=0}^n$. Extending this approach to a depth$-N$ Log-ODE method on generic intervals $t_0 = r_0 < \cdots < r_m = t_n$ produces flows

$$\exp\left(\sum_{k=1}^{\beta(d_\omega, N)} \bar{A}_\theta^k \lambda_k^l\right), \tag{5.9}$$

where

$$\log(S^N(\omega^X)_{[r_l, r_{l+1}]}) = \sum_{k=1}^{\beta(d_\omega, N)} \lambda_k^l \hat{e}_k, \tag{5.10}$$

$\hat{e}_k$ is a Hall basis for the space where the depth-N truncated log-signature lives,

$$\bar{A}_\theta^k = A_\theta^k \tag{5.11}$$

for $1 \leq k \leq d_\omega$, and

$$\bar{A}_\theta^k = [\bar{A}_\theta^i, \bar{A}_\theta^j] = \bar{A}_\theta^i \bar{A}_\theta^j - \bar{A}_\theta^j \bar{A}_\theta^i \tag{5.12}$$

when the basis element $\hat{e}_k$ corresponds to the Lie bracket of $\hat{e}_i$ and $\hat{e}_j$ [Reutenauer 1993]. Further details are given in Sections 2.5 and 4.4. The key point is that the Log-ODE method again produces a linear flow on each interval, so the parallel associative scan remains applicable. Moreover, the Lie brackets $\bar{A}_\theta^k$ are computed from products and commutators of the matrices A_θ^i, rather than from the forward-mode auto-differentiated Jacobian-vector products required in Log-NCDEs, which substantially reduces the computational cost. It also reduces the memory and I/O cost of the scan, since only $m < n$ state-transition matrices must be materialised in GPU memory. We refer to the combination of Linear NCDEs and the Log-ODE method as Log-Linear NCDEs.

5.2.3 Expressivity

The previous subsection showed that Linear NCDEs offer substantial computational advantages. This raises the question of whether these gains come at the cost of expressivity. For a fixed hidden dimension, Linear NCDEs are less expressive than NCDEs,

since the former are a strict subclass of the latter. However, once the hidden dimension is allowed to vary, both model classes have the same theoretical expressivity.

Theorem 5.2 (Maximally Expressive Linear NCDEs [Cirone et al. 2024])**.** *Let $\mathcal{X}$ be the space of bounded variation paths on the interval $[t_0, t_n]$ that start at a common point and include time as a channel, endowed with the $1-$variation topology. Let $\mathcal{F}$ be the class of real-valued maps $X \mapsto y_{t_n}$ induced by Linear NCDEs from Definition 5.1 with $\omega_s^X = X_s$ for $X \in \mathcal{X}$, $d_h \in \mathbb{N}$, and $d_y = 1$. Then $\mathcal{F}$ is maximally expressive on $\mathcal{X}$.*

Proof. The result follows from the universality of the signature, Corollary 2.28, together with the fact that the truncated signature solves a linear CDE, as shown in (2.82). For more details, see [Cirone et al. 2024, Theorem B.13]. □

Although a Linear CDE is linear in the hidden state, its evolution is driven by the multiplicative interaction between the hidden state and the increments of the driving path. This is precisely the mechanism that generates iterated integrals, since each tensor level of the truncated signature is obtained by integrating the previous level against the path. Therefore, just as repeated application of the map $(x, y) \mapsto xy$ generates monomials, repeated multiplication of the hidden state by path increments generates the tensor levels of the truncated signature. Hence, this single multiplicative interaction is sufficient for maximal expressivity.

Similarly to NCDEs, Theorem 5.2 can be extended from path-to-point functions to path-to-path functions by replacing the linear readout L_θ^2 with a FCNN acting on h_t, as shown in [Cirone et al. 2024, Proposition D.2]. Furthermore, the linear setting allows us to give a statement about expressivity even when the entries of A_θ are sampled randomly from a prescribed distribution.

Definition 5.3 (Maximal Probabilistic Expressivity)**.** *Let $\mathcal{X}$ be a topological space, $M \in \mathbb{N}$, and $\mathcal{F}^M = \{f_\theta^M : \mathcal{X} \to \mathbb{R} \mid \theta \in \Theta^M\}$ be a class of real-valued functions on $\mathcal{X}$ defined by*

$$f_\theta^M(\omega) = l_{\theta_2}(\tilde{f}_{\theta_1}^M(\omega)), \tag{5.13}$$

for $\omega \in \mathcal{X}$, where $\tilde{f}_{\theta_1}^M : \mathcal{X} \to \mathbb{R}^M$, $l_{\theta_2} \in \mathbb{R}^M$ is a linear readout, $\theta_1 \in \Theta_1^M$, $\theta_2 \in \Theta_2^M$, and $\Theta^M = \Theta_1^M \cup \Theta_2^M$. Given a sequence of probability measures $\mathbb{P}_M$ on Θ_1^M with $\theta_1 \sim \mathbb{P}_M$, $\mathcal{F}$ has maximal probabilistic expressivity if, for every compact set $\mathcal{K} \subset \mathcal{X}$

and every real-valued continuous function $f : \mathcal{K} \to \mathbb{R}$, the following property holds:

$$\forall \epsilon > 0, \lim_{M \to \infty} \mathbb{P}_M \left\{ \exists l_{\theta_2} \ s.t. \sup_{\omega \in \mathcal{K}} \left| f(\omega) - f_\theta^M(\omega) \right| < \epsilon \right\} = 1. \tag{5.14}$$

Theorem 5.4 (Maximal Probabilistic Expressivity for Linear NCDEs [Cirone et al. 2024])**.** *Let $\mathcal{X}$ be the space of bounded variation paths on the interval $[t_0, t_n]$ that start at a common point and include time as a channel, endowed with the $1-$variation topology. Let $\mathcal{F}$ be the class of real-valued maps $X \mapsto y_{t_n}$ induced by Linear NCDEs from Definition 5.1 with $\omega_s^X = X_s$ for $X \in \mathcal{X}$, $d_h \in \mathbb{N}$, $d_y = 1$, and A_θ^i with i.i.d entries from $\mathcal{N}\left(0, \frac{1}{d_h}\right)$. Then $\mathcal{F}$ has maximal probabilistic expressivity on $\mathcal{X}$.*

Proof. For a detailed proof see [Cirone et al. 2024, Theorem B.13]. Here we present the core idea. In the proof of Theorem 5.2 the depth$-N$ truncated tensor algebra of $\mathbb{R}^{d_X}$ is recreated as a Euclidean space, requiring $\mathcal{O}(d_X^N)$ orthogonal vectors. At the cost of losing exactness, we can leverage results of the Johnson-Lindenstrauss type to find $O(e^{\epsilon^2 N})$ vectors in $\mathbb{R}^N$ orthogonal up to an ϵ error, using random projections [Dasgupta et al. 2003]. This relaxation allows us to use random matrices A_θ^i and still achieve maximal expressivity. □

In the context of machine learning, maximal probabilistic expressivity may be viewed as a more promising property than maximal expressivity. For sufficiently large hidden dimension d_h, it implies there exists a significant abundance of parameters θ_1 that are capable of achieving uniformly bounded and arbitrarily low error rates with a linear readout layer. This in turn suggests that, when d_h is large, random initialisations of the matrices A_θ^i can already encode informative features of the input path, and may therefore provide a useful starting point for optimisation.

Linear NCDEs with random A_θ^i have also been studied under the name randomised signatures. There is a growing body of work establishing their expressivity and empirical effectiveness [Cuchiero et al. 2021; Cuchiero et al. 2022; Compagnoni et al. 2023]. In particular, the concept behind the proof of Theorem 5.4 goes back to Cuchiero et al. 2021, who used a Johnson-Lindenstrauss type result to show that randomised signatures can match the expressivity of classical signatures with substantially fewer features.

5.2.4 Related Approaches

Linear NCDEs are the continuous-time analogue of a tensor RNN,

$$h_{t_{i+1}} = \sigma\left(\left(\sum_{j=1}^{d_x} A_\theta^j \, x_{t_i}^j\right) h_{t_i}\right), \tag{5.15}$$

with $\sigma(x) = x$. Tensor RNNs were introduced by Giles et al. [1989], who conjectured they could learn any regular language. Architectures of the form (5.15) are also known as 2-RNNs [Lizaire et al. 2024]. Building on tensor RNNs, Sutskever et al. 2011 introduced multiplicative RNNs, which take the generic form

$$h_{t_{i+1}} = \sigma\left(A_\theta(x_{t_i})h_{t_i} + B_\theta(x_{t_i})x_{t_i}\right). \tag{5.16}$$

When $\sigma(x) = x$, (5.16) is an RNN whose update is linear in the hidden state,

$$h_{t_{i+1}} = A_\theta(x_{t_i})h_{t_i} + B_\theta(x_{t_i})x_{t_i}. \tag{5.17}$$

Structured state-space models (SSMs), so named for the similarity between (5.17) and classical state-space models (4.5), are a subclass of (5.17) that utilise a structured state-transition matrix A_θ [Gu et al. 2022a].

Section 5.3.2 will show that two prominent SSMs, S4 and Mamba, can be recast as affine Linear NCDEs,

$$h_t = h_{t_0} + \int_{t_0}^{t} A_\theta h_s \mathrm{d}\omega_s^X + \int_{t_0}^{t} B_\theta \mathrm{d}\xi_s^X, \tag{5.18}$$

where $B_\theta \in \mathbb{R}^{d_h \times d_\xi}$ is a trainable matrix and $\xi : \mathcal{X}(d_X) \to \mathcal{X}(d_\xi)$ is another function of the interpolated data [Gu et al. 2022a; Gu et al. 2024]. Concurrently with Linear NCDEs, Merrill et al. [2024] introduced IDS4, an SSM corresponding to a discretised (5.18) with

$$\frac{\omega_{t_{j+1}}^X - \omega_{t_j}^X}{t_{j+1} - t_j} = X_{t_j}, \qquad \xi_s^X = \omega_s^X. \tag{5.19}$$

Motivated by the inability of earlier SSM architectures to learn regular languages, Merrill et al. [2024] proved that IDS4 can learn any regular language, thereby answering the conjecture of Giles et al. [1989] in the case of an affine tensor RNN. In Section 5.3.3, we use the Linear NCDE framework to fully characterise the expressivity of IDS4, S4, and Mamba.

Table 5.1: Mean and standard deviation of test accuracy over five different splits, together with the time per 1000 training steps and GPU memory usage, for NCDE, Log-NCDE, Linear NCDE (LNCDE), and Log-Linear NCDE (Log-LNCDE) on EigenWorms.

	NCDE	Log-NCDE	LNCDE	Log-LNCDE
Test Accuracy	75.0 ± 3.9	85.6 ± 5.1	87.2 ± 5.2	87.8 ± 5.7
Recurrent time per 1000 training steps (s)	26020	2263	299.4	29.9
Recurrent GPU Memory (MB)	3484	3494	9624	3486
Parallel time per 1000 training steps (s)	–	–	100.4	17.5
Parallel GPU Memory (MB)	–	–	13730	3492

5.2.5 Experiments

We compare an NCDE, Log-NCDE, Linear NCDE, and Log-Linear NCDE on EigenWorms, the time series classification dataset with the most observations from those considered in Section 4.4.6. This makes it a natural stress test for practical scalability, since both the cost of repeatedly solving a non-linear differential equation and the memory cost of materialising linear flows are amplified on long sequences. Taking

$$\frac{\omega^X_{t_{j+1}} - \omega^X_{t_j}}{t_{j+1} - t_j} = (1, X_{t_j}), \tag{5.20}$$

and setting $\xi^X_t = 0$ gives an input dimension of $d_\omega = 7$. All four models use a hidden dimension of $d_h = 128$, and each time series contains 17984 observations. We keep the hyperparameters for the NCDE and its linear variant, and for the Log-NCDE and its linear variant, identical to those selected by the hyperparameter optimisation in Section 4.4.6, except for replacing the non-linear vector field f_θ with a linear vector field A_θ. See Table 4.3 for full details. The models are compared on average test set accuracy, as well as time per training step and GPU memory on an NVIDIA RTX 4090 with a batch size of 1. The parallel associative scan is applied to chunks of 128 steps, with each chunk processed recurrently.

As shown in Table 5.1, replacing the non-linear vector field of an NCDE with a linear vector field increases the average test accuracy from 75.0% to 87.2%, bringing performance in line with Log-NCDEs, and therefore with the other state-of-the-art time series models considered in Section 4.4.7. A possible cause is that removing

the differential equation solver improves training stability, since the Linear NCDE is solved exactly over each interval. This has the added benefit of reducing the recurrent time per training step by a factor of over 80. Equivalently, 10000 training steps would take around 3 days for the NCDE, but only around 50 minutes for the Linear NCDE. When a parallel associative scan is applied, the reduction in time per training step increases to a factor of over 250, so that 10000 training steps now take under 17 minutes. However, this change also significantly increases GPU memory usage.

Applying the Log-ODE method brings GPU memory usage back to a level comparable to that of the models with non-linear vector fields, while further reducing the time per training step and maintaining a high average test set accuracy. Overall, a Log-Linear NCDE combined with a parallel associative scan reduces the time per training step by a factor of almost 1500 relative to the NCDE, so that 10000 training steps take under 3 minutes rather than around 3 days. This is achieved while increasing the average test accuracy by 12.8 percentage points and using only 8MB more GPU memory than the NCDE.

Before broadening our empirical study of Linear NCDEs to additional tasks and baselines in Sections 5.3.4 and 5.4.6, we establish their theoretical relationship to SSMs. This connection provides the foundation for Section 5.4, which introduces Structured Linear NCDEs, where A_θ^i is replaced by a structured variant to further improve model efficiency.

5.3 Structured State-Space Models

5.3.1 Definition

In 2021, drawing inspiration from traditional state-space models such as (4.5), Gu et al. [2022a] introduced S4, the first SSM. The model is based on a continuous differential equation,

$$\begin{aligned} \mathrm{d}h_s^j &= C_\theta^j h_s^j + D_\theta X_s^j \mathrm{d}s, \\ y_s^j &= E_\theta \cdot h_s^j \end{aligned} \tag{5.21}$$

where each channel of the input path X_s^j produces a complex-valued hidden state $h_s^j \in \mathbb{C}^{d_h}$ and the trainable parameters are $D_\theta \in \mathbb{C}^{d_h}$, $E_\theta \in \mathbb{C}^{d_h}$, and the channel specific $C_\theta^j \in \mathbb{C}^{d_h \times d_h}$. S4 is a stable discretisation of (5.21),

$$h_{t_{i+1}}^j = \bar{C}_\theta^j h_{t_i}^j + \bar{D}_\theta^j x_{t_i}^j \tag{5.22}$$

where $\bar{C}_\theta^j$ and $\bar{D}_\theta^j$ are determined by the method of discretisation and the channel-dependent step size Δ^j. A common choice is the zero-order hold discretisation,

$$\begin{aligned}\bar{C}_\theta^j &= \exp(\Delta^j C_\theta^j),\\ \bar{D}_\theta^j &= (\Delta^j C_\theta^j)^{-1}(\exp(\Delta^j C_\theta^j) - I)\Delta^j D_\theta \approx \Delta^j D_\theta.\end{aligned} \tag{5.23}$$

During inference, S4 is equivalent to a linear RNN. However, the training via gradient descent is performed on the continuous-time variables, which helps manage vanishing and exploding gradients [Orvieto et al. 2023; Zucchet et al. 2024]. The "structured" aspect refers to specific parametrisations and initialisations of the state-to-state transition matrices C_θ^j to ensure stability and efficiency, particularly for processing long sequences. For example, S4D uses diagonal C_θ^j, which has become the dominant choice [Gu et al. 2022b].

SSMs have achieved state-of-the-art results on long-range reasoning benchmarks [Tay et al. 2021] and demonstrated strong performance in various domains including vision [Nguyen et al. 2022], audio [Goel et al. 2022], biological signals [Gu et al. 2022a], and reinforcement learning [Lu et al. 2023]. SSMs have garnered significant interest, as their computational complexity scales linearly in sequence length, while attention scales quadratically. Moreover, unlike non-linear RNNs such as LSTMs [Hochreiter et al. 1997] and GRUs [Cho et al. 2014], they can be efficiently parallelised on GPUs during training using the approach outlined in Section 5.2.2 [Smith et al. 2023]. While standard SSMs perform well on signal processing tasks, their computational power is limited: the core sequential mechanism of S4 is equivalent to a convolution [Li et al. 2023]. This represents a drawback in challenging domains such as text and genetics, where the ability to select data efficiently in an input-dependent manner is crucial [Wang et al. 2023a; Fu et al. 2023; Arora et al. 2024].

In 2023, Gu et al. proposed Mamba, which uses a recurrent layer based on a real-valued discretised model,

$$h_{t_{i+1}}^j = \bar{C}_\theta^j(x_{t_i}) h_{t_i}^j + \bar{D}_\theta^j(x_{t_i}) x_{t_i}^j, \tag{5.24}$$

where

$$\begin{aligned}\bar{C}_\theta^j(x_{t_i}) &= \exp(\Delta^j(x_{t_i})C_\theta),\\ \bar{D}_\theta^j(x_{t_i}) &= (\Delta^j(x_{t_i})C_\theta)^{-1}(\exp(\Delta^j(x_{t_i})C_\theta) - I)\Delta^j(x_{t_i})D_\theta x_{t_i},\\ &\approx \Delta^j(x_{t_i})D_\theta x_{t_i},\end{aligned} \tag{5.25}$$

and the trainable parameters are $D_\theta \in \mathbb{R}^{d_h \times d_x}$, a shared diagonal state-transition matrix $C_\theta \in \mathbb{R}^{d_h \times d_h}$, and

$$\Delta^j(x_{t_i}) = \text{softplus}(\alpha_\theta^j \cdot x_{t_i} + \beta_\theta^j), \tag{5.26}$$

with trainable parameters $\alpha_\theta^j \in \mathbb{R}^{d_x}$ and $\beta_\theta^j \in \mathbb{R}$ [Gu et al. 2024]. This recurrent layer is known as S6. Compared to S4, the evolution of each channel's hidden state is now dependent on all channels of the current input x_{t_i} through $\Delta^j(x_{t_i})$. This dependence is intended to gate the flow of information by controlling the balance between the previous hidden state and the new update. When $\Delta^j(x_{t_i}) \ll 1$, the state-transition matrix is close to the identity and $\bar{D}_\theta^j(x_{t_i})x_{t_i}^j$ is small, so the hidden state is largely preserved. For larger values of $\Delta^j(x_{t_i})$, the new update has greater influence, leading to stronger state changes and greater forgetting. This modification allows Mamba to achieve state-of-the-art performance on a range of language modelling tasks. Similar ideas appear in recent attention-inspired architectures such as RWKV, Gated Linear Attention, and HGRN2 [Peng et al. 2023; Yang et al. 2024a; Qin et al. 2024].

The expressiveness of non-linear RNNs, such as (4.13), has been extensively studied since the seminal work of Siegelmann et al. [1992]. In particular, Hanson et al. [2020] proved that wide enough non-linear RNNs can approximate non-linear time-homogeneous systems of differential equations driven by input paths to arbitrary precision. However, SSMs have state-to-state transitions which are linear in the hidden state. Although this allows for parallel-in-time computation, it also reduces the recurrence's expressivity. In 2022, Li et al. showed that linear RNNs, a generic term for S4 like recurrences, can approximate arbitrary convolution filters in the width limit [Li et al. 2022]. It has also been shown that single layer linear recurrences are universal approximators, when equipped with a fixed point-wise FCNN acting across the recurrence output [Orvieto et al. 2024; Wang et al. 2023b].

Mamba's recurrence falls neither in the linear RNN nor the non-linear RNN setting: it is linear in the hidden state, but unlike S4 it is not linear time-invariant, since the input controls the recurrence's eigenvalues. This input dependence increases Mamba's expressivity relative to S4 and improves language modelling performance. In the remainder of this section, we investigate the approximation capabilities of Mamba by recasting the model as an affine Linear NCDE. Existing work on Mamba's expressiveness has focused on specific toy tasks [Jelassi et al. 2024] or the framework of formal language theory [Merrill et al. 2024]. Here, we seek a generic result.

5.3.2 SSMs are Linear NCDEs

First, we show that the real-valued continuous version of S4, (5.21), can be rewritten as an affine Linear NCDE, (5.18). Let $h_t \in \mathbb{R}^{d_h d_X}$,

$$\begin{aligned} \omega_t^{X,k} &= t, \\ \xi_t^X &= \int_{t_0}^t X_s ds, \end{aligned} \tag{5.27}$$

and

$$\begin{aligned} A_\theta^k &= \mathrm{diag}(0, \ldots, 0, C_\theta^k, 0, \ldots, 0) \in \mathbb{R}^{d_h d_X \times d_h d_X}, \\ B_\theta &= \mathrm{diag}(D_\theta, \ldots, D_\theta) \in \mathbb{R}^{d_h d_X \times d_X}, \end{aligned} \tag{5.28}$$

where the non-zero diagonal element of A^k is in the k^{th} position. Then the affine Linear NCDE (5.18) corresponds to a stacked version of (5.21) with real-valued parameters.

The discrete version of Mamba's recurrence, (5.24), can be considered a zero-order hold discretisation of

$$\mathrm{d}h_s^j = C_\theta \Delta^j(X_s) h_s^j + D_\theta X_s \Delta^j(X_s) X_s^j \mathrm{d}s \tag{5.29}$$

with a step size of 1, where $C_\theta \in \mathbb{R}^{d_h \times d_h}$ and $D_\theta \in \mathbb{R}^{d_h \times d_X}$. These equations can be stacked and rewritten as an affine Linear NCDE by taking

$$\begin{aligned} \omega_t^{X,k} &= \int_{t_0}^t \mathrm{softplus}(\alpha^k \cdot X_s + \beta^k) \mathrm{d}s, \\ \xi_t^X &= \int_{t_0}^t \begin{bmatrix} X_t \mathrm{softplus}(\alpha^1 \cdot X_s + \beta^1) X_s^1 \\ \vdots \\ X_t \mathrm{softplus}(\alpha^{d_X} \cdot X_s + \beta^{d_X}) X_s^{d_X} \end{bmatrix} ds, \end{aligned} \tag{5.30}$$

and

$$\begin{aligned} A_\theta^k &= \mathrm{diag}(0, \ldots, 0, C_\theta, 0, \ldots, 0) \in \mathbb{R}^{d_h d_X \times d_h d_X}, \\ B_\theta &= \mathrm{diag}(D_\theta, \ldots, D_\theta) \in \mathbb{R}^{d_h d_X \times d_X^2}, \end{aligned} \tag{5.31}$$

where the non-zero diagonal element of A_θ^k is in the k^{th} position.

In this framework, the major difference between S4 and Mamba's recurrence is the choice of ω and ξ. Gu et al. [2024] argue that this difference allows Mamba to gate the hidden state based on the input stream, and therefore perform in-context learning. For this reason, we refer to ω and ξ as the gating functions. A notable difference

between SSMs and the general Linear NCDE is that SSMs process the hidden state for each channel individually, whereas a Linear NCDE mixes the hidden state and the channels of the transformed input path ω^X in the recurrent step. As shown in the next section, this has a significant impact on the expressivity of SSMs.

5.3.3 Expressivity of SSMs

For certain choices of ω_t^X and ξ_t^X, the affine Linear NCDE is maximally expressive. For example, $\omega_t^{X,k} = X_t^k$ and $\xi_t^X = 0$ puts you in the setting of Theorem 5.2. However, both S4 and Mamba use alternative choices for ω_t^X and ξ_t^X, so we now characterise the expressiveness of generic affine Linear NCDEs.

Theorem 5.5. *Let $\mathcal{X}(d)$ denote the space of bounded-variation d-dimensional paths on the interval $[t_0, t_n]$ which all begin at the same point and contain time as a channel, endowed with the $1-$variation topology. For continuous gates $\omega^X : \mathcal{X}(d_X) \to \mathcal{X}(d_\omega)$ and $\xi^X : \mathcal{X}(d_X) \to \mathcal{X}(d_\xi)$, let*

$$\mathcal{F} = \left\{ (X, t) \mapsto \Psi(\omega^X_{[t_0,t]}) \cdot X_0 + \int_{t_0}^{t} \Phi(\omega^X_{[s,t]}) \cdot \mathrm{d}\xi^X_s \right\}, \tag{5.32}$$

where $\Psi : \mathcal{X}(d_\omega) \to \mathbb{R}^{d_X}$ and $\Phi : \mathcal{X}(d_\omega) \to \mathbb{R}^{d_\xi}$ are continuous functions and $\omega^X_{[s,t]}$ is ω^X restricted to the interval $[s, t]$. Then for any compact set $\mathcal{K} \subseteq \mathcal{X}(d_X)$, any continuous paths ω^X and ξ^X with $\omega_t^{X,1} \equiv t$ and $\omega_t^{X,2} \equiv t^2$, any $\epsilon > 0$, and any $F \in \mathcal{F}$, there exists a choice of hidden dimension $d_h \geq 1$ and parameters for the affine Linear NCDE such that

$$\sup_{(X,t) \in \mathcal{K} \times [t_0, t_n]} |F(X, t) - y_t| \leq \epsilon. \tag{5.33}$$

A complete proof of Theorem 5.5 can be found in [Cirone et al. 2024, Appendix B]. Here, we give an overview of the argument. We begin by deriving an explicit formula for the solution of an affine Linear NCDE, following the same Picard iteration argument used to prove Theorem 2.37. This formula shows that the solution is represented in terms of the signature of the transformed path ω^X, with ξ^X entering through weighted integrals against those features. In particular, it makes clear that the choice of ω^X is the main factor governing the model's expressivity.

Lemma 5.6. *Let ω_t and ξ_t be bounded variation paths of dimension d_ω and d_ξ, respectively. For any choice of $A^i \in \mathbb{R}^{d_h \times d_h}$, and $B \in \mathbb{R}^{d_h \times d_\xi}$, the unique solution to*

$$dh_t = \sum_{i=1}^{d_\omega} A^i h_t \mathrm{d}\omega_t^i + B\mathrm{d}\xi_t, \tag{5.34}$$

is

$$h_t = \sum_{I\in\mathcal{I}} S^I_{[t_0,t]}(\omega)A^I h_{t_0} + \sum_{I\in\mathcal{I}} A^I B \int_{t_0}^t S^I_{[s,t]}(\omega)\mathrm{d}\xi_s, \tag{5.35}$$

where $\mathcal{I}$ is the set of multi-indices

$$\mathcal{I} = \{\emptyset\} \cup \{I | I = (i_1, \dots, i_k), k \in \mathbb{N}, 1 \le i_j \le d_\omega\} \tag{5.36}$$

with $|I| = |(i_1, \dots, i_k)| = k$, $A^I = A^{i_k} \cdots A^{i_1}$ with $A^\emptyset$ being the identity, and $S^I_{[s,t]}(\omega)$ is the term in the signature of ω over $[s,t]$ corresponding to the multi-index I,

$$S^I_{[s,t]}(\omega) = \underbrace{\int\cdots\int}_{s\le u_1<\cdots<u_k\le t} \mathrm{d}\omega^{i_1}_{u_1} \cdots \mathrm{d}\omega^{i_k}_{u_k}, \tag{5.37}$$

with $S^\emptyset = 1$.

Proof. The proof uses Picard iteration, following the same approach as the proof of Theorem 2.37. Let $h^{(0)}_t = h_{t_0}$ and for $n \ge 0$ define

$$h^{(n+1)}_t = h_{t_0} + \sum_{i=1}^{d_\omega} \int_{t_0}^t A^i h^{(n)}_s \,\mathrm{d}\omega^i_s + B\int_{t_0}^t \mathrm{d}\xi_s. \tag{5.38}$$

Assume for some $n \ge 0$ that,

$$h^{(n)}_t = \sum_{|I|\le n} S^I_{[t_0,t]}(\omega)A^I h_{t_0} + \sum_{|I|\le n-1} A^I B \int_{t_0}^t S^I_{[s,t]}(\omega)\,\mathrm{d}\xi_s. \tag{5.39}$$

Then,

$$\begin{aligned} h^{(n+1)}_t = h_{t_0} &+ \sum_{i=1}^{d_\omega} A^i \int_{t_0}^t \left(\sum_{|I|\le n} S^I_{[t_0,s]}(\omega)A^I h_{t_0}\right) \mathrm{d}\omega^i_s \\ &+ \sum_{i=1}^{d_\omega} A^i \int_{t_0}^t \left(\sum_{|I|\le n-1} A^I B \int_{t_0}^s S^I_{[u,s]}(\omega)\,\mathrm{d}\xi_u\right) \mathrm{d}\omega^i_s + B\int_{t_0}^t \mathrm{d}\xi_s. \end{aligned} \tag{5.40}$$

For the first term,

$$\sum_{i=1}^{d_\omega} A^i \int_{t_0}^t \left(\sum_{|I|\le n} S^I_{[t_0,s]}(\omega)A^I h_{t_0}\right) \mathrm{d}\omega^i_s = \sum_{i=1}^{d_\omega}\sum_{|I|\le n} A^i A^I h_{t_0} \int_{t_0}^t S^I_{[t_0,s]}(\omega)\mathrm{d}\omega^i_s. \tag{5.41}$$

By the recursive definitions of A^I and S^I,

$$\sum_{i=1}^{d_\omega}\sum_{|I|\le n} A^i A^I h_{t_0} \int_{t_0}^t S^I_{[t_0,s]}(\omega)\mathrm{d}\omega^i_s = \sum_{1\le|I|\le n+1} S^I_{[t_0,t]}(\omega)A^I h_{t_0}. \tag{5.42}$$

For the second term,

$$\sum_{i=1}^{d_\omega} A^i \int_{t_0}^{t} \left(\sum_{|I|\leq n-1} A^I B \int_{t_0}^{s} S^I_{[u,s]}(\omega)\,\mathrm{d}\xi_u \right) \mathrm{d}\omega_s^i = \sum_{i=1}^{d_\omega} \sum_{|I|\leq n-1} A^i A^I B \int_{t_0}^{t}\int_{t_0}^{s} S^I_{[u,s]}(\omega)\,\mathrm{d}\xi_u\,\mathrm{d}\omega_s^i. \tag{5.43}$$

By Fubini's theorem,

$$\sum_{i=1}^{d_\omega} \sum_{|I|\leq n-1} A^i A^I B \int_{t_0}^{t}\int_{t_0}^{s} S^I_{[u,s]}(\omega)\,\mathrm{d}\xi_u\,\mathrm{d}\omega_s^i = \sum_{i=1}^{d_\omega} \sum_{|I|\leq n-1} A^i A^I B \int_{t_0}^{t}\int_{s}^{t} S^I_{[s,u]}(\omega)\,\mathrm{d}\omega_u^i\,\mathrm{d}\xi_s. \tag{5.44}$$

Again using the recursive definitions of A^I and S^I,

$$\sum_{i=1}^{d_\omega} \sum_{|I|\leq n-1} A^i A^I B \int_{t_0}^{t}\int_{s}^{t} S^I_{[s,u]}(\omega)\,\mathrm{d}\omega_u^i\,\mathrm{d}\xi_s = \sum_{1\leq|I|\leq n} A^I B \int_{t_0}^{t} S^I_{[s,t]}(\omega)\,\mathrm{d}\xi_s. \tag{5.45}$$

Therefore,

$$h_t^{(n+1)} = \sum_{|I|\leq n+1} S^I_{[t_0,t]}(\omega) A^I h_{t_0} + \sum_{|I|\leq n} A^I B \int_{t_0}^{t} S^I_{[s,t]}(\omega)\,\mathrm{d}\xi_s. \tag{5.46}$$

Since (5.39) is true for $n = 0$, it holds for all $n \geq 0$. Let $M = \max_i\{\|A^i\|_{\mathrm{op}}\}$. By [Lyons 1994, Theorem 2.2.1], there exists finite $C > 1$ such that

$$\left\|S^I_{[t_0,t]}(\omega)\right\| \leq C\frac{\|\omega\|^{|I|}_{1\text{-var};[t_0,t]}}{|I|!}. \tag{5.47}$$

Hence,

$$\|S^I_{[t_0,t]}(\omega)A^I h_{t_0}\| \leq CM^{|I|}\frac{\|\omega\|^{|I|}_{1\text{-var};[t_0,t]}}{|I|!}\|h_{t_0}\| \tag{5.48}$$

and

$$\left\|A^I B \int_{t_0}^{t} S^I_{[s,t]}(\omega)\,\mathrm{d}\xi_s\right\| \leq CM^{|I|}\|B\|_{\mathrm{op}}\frac{\|\omega\|^{|I|}_{1\text{-var};[t_0,t]}}{|I|!}\|\xi\|_{1\text{-var};[t_0,t]}, \tag{5.49}$$

The factorial decay means $\lim_{n\to\infty} h_t^{(n)}$ converges uniformly to h_t where

$$h_t = \sum_{I\in\mathcal{I}} S^I_{[t_0,t]}(\omega) A^I h_{t_0} + \sum_{I\in\mathcal{I}} A^I B \int_{t_0}^{t} S^I_{[s,t]}(\omega)\mathrm{d}\xi_s. \tag{5.50}$$

Continuity of the Riemann-Stieltjes integral for bounded variation paths allows the limit to be passed through the integral, so h_t is a solution to (5.34). For uniqueness, suppose there are two solutions h_t and $\tilde{h}_t$ to the CDE. Then by linearity, $g_t = h_t - \tilde{h}_t$ satisfies

$$g_t = \int_{t_0}^{t} \sum_{i=1}^{d_\omega} A^i g_s \mathrm{d}\omega_s^i \tag{5.51}$$

with $g_{t_0} = 0$. Therefore, $g_t = 0$ and the solution is unique. □

Lemma 5.6 shows that an affine Linear NCDE is not a simple linear recurrence. Rather, its solution is a linear map on features built from iterated integrals of the transformed path ω^X, together with additional features built from integrating those terms against $d\xi^X$. This helps explain why Linear NCDEs are maximally expressive. Although the dynamics are linear in the hidden state, the hidden state itself is built from highly non-linear path features generated recursively through repeated interactions with the driving path. Compared to classical signature methods, a Linear NCDE learns to construct the weighted-signature features most relevant to the task, rather than relying on a fixed truncated collection of signature terms. We now package these weighted-signature features into a single feature map.

Definition 5.7. *Let $\mathbb{W}_{d_X,d_\omega,d_\xi}$ be the set of words in the alphabet*

$$\mathcal{A}_{d_X,d_\omega,d_\xi} = \{\boldsymbol{e}_i\}_{i=1}^{d_X} \cup \{\boldsymbol{\epsilon}_j^\xi\}_{j=1}^{d_\xi} \cup \{\boldsymbol{\epsilon}_k^\omega\}_{k=1}^{d_\omega} \tag{5.52}$$

For fixed ω and ξ, define $T(X) : [t_0, t_n]^2 \to l^2(\mathbb{W}_{d_X,d_\omega,d_\xi}) \subseteq T((\mathcal{A}_{d_X,d_\omega,d_\xi}))$ as the unique solution to:

$$T_{[s,t]}(X) = \sum_{i=1}^{d_X} X_s^i \boldsymbol{e}_i + \sum_{j=1}^{d_\xi} \xi_t^{X,j} \boldsymbol{\epsilon}_j^\xi + \sum_{k=1}^{d_\omega} \int_s^t T_{[s,u]}(X)\, \mathrm{d}\omega_u^{X,k} \otimes \boldsymbol{\epsilon}_k^\omega \tag{5.53}$$

This is similar to the tensor-valued CDE representation of the signature seen in Section 2.4.1,

$$\mathrm{d}S_{[t_0,s]}(\omega^X) = S_{[t_0,s]}(\omega^X) \otimes \mathrm{d}\omega_s^X, \tag{5.54}$$

with the addition of two terms to track X_s and ξ^X [Salvi et al. 2021]. We could also understand $T(X)_{[s,t]}$ as a sub-tensor of

$$X_s \otimes S_{[s,t]}([\omega^X, \xi^X]), \tag{5.55}$$

but in doing this we would have to explicitly ignore most of the terms in this tensor. The CDE (5.53) does exactly this, but implicitly. In any case, the subtensor view shows that $T : \mathcal{X}(d_X) \times [t_0, t_n]^2 \to l^2(\mathbb{W}_{v,v_\omega,v_\xi})$ is well defined and continuous.

Having defined a feature map $T(\cdot)_{[s,t]}$ with values in the Hilbert space $l^2(\mathbb{W}_{d_X,d_\omega,d_\xi})$, it is possible to associate to it a Reproducing Kernel Hilbert Space (RKHS) [Berlinet et al. 2004], where the kernel is induced by the l^2 product. We denote the RKHS by $\mathcal{H}_t^{\omega,\eta}$. Proposition B.10 in Cirone et al. 2024 demonstrates that linear maps of h_t are in the uniform closure of $\mathcal{H}_t^{\omega,\eta}$, and Proposition B.11 allows us to characterise the closure as

$$F \in \left\{ (X,t) \mapsto \Psi(\omega_{[t_0,t]}^X) \cdot X_0 + \int_{t_0}^t \Phi(\omega_{[s,t]}^X) \cdot \mathrm{d}\xi_s^X \right\}. \tag{5.56}$$

The proof of Proposition B.11 relies on the signature being able to separate points in the image of the map $(X, s, t) \mapsto \omega^X_{[s,t]}$. By Lemma 2.25, the signature separates the points $\omega^X_{[s,t]}$ from $\tilde{\omega}^X_{[s,t]}$ as ω^X is augmented to include time. In order to separate points of the form $\omega^X_{[s,t]}$ from $\tilde{\omega}^X_{[s',t']}$, we include t^2 as a channel in ω^X, as then $S_{[s,t]}(\omega^X) = S_{[s,t]}(\tilde{\omega}^X)$ implies that

$$\begin{aligned} \int_s^t d(r^2) = t^2 - s^2 = (t')^2 - (s')^2 = \int_{s'}^{t'} d(r^2), \\ \int_s^t d(r) = t - s = t' - s' = \int_{s'}^{t'} d(r). \end{aligned} \tag{5.57}$$

Therefore, $s' = s$ and $t' = t$.

The proof of Theorem 5.5 concludes by showing that linear maps on h_t are dense in the uniform closure of $\mathcal{H}^{\omega,\eta}_{[t_0,t_n]}$, using the same strategy as the proof of Theorem 4.4, that NCDEs are maximally expressive [Kidger 2022].

Theorem 5.5 can be seen as a generalisation to generic functions ω and ξ of [Li et al. 2022, Theorem 7]. That result considered the case $\omega^X_t = t$ and $\xi^X_t = \int_{t_0}^t X_s \mathrm{d}s$, which is the setting of S4, S5, and the LRU [Gu et al. 2022a; Smith et al. 2023; Orvieto et al. 2023]. In this setting, the only information contained in $\omega_{[s,t]}$ is the increment $t - s$. Therefore, (5.32) reduces to

$$\left\{ (X, t) \mapsto \psi(t - t_0) + \int_{t_0}^t \phi(t - s) \cdot X_s \mathrm{d}s \right\}, \tag{5.58}$$

which is the set of linear filters on the input.

Now consider $\omega^X_t = X_t$. The first term in the function class

$$\left\{ \Psi(X_{[t_0,t]}) + \int_{t_0}^t \Phi(X_{[s,t]}) \cdot \mathrm{d}\xi^X_s \right\} \tag{5.59}$$

is already enough to establish that the output $L^2_\theta h_t$ is a non-linear function of all previously seen inputs $X_{[t_0,t]}$. However, the term $\Psi(X_{[t_0,t]})$ is only non-trivial when $h_{t_0} \neq 0$. A case similar to Mamba is $h_{t_0} = 0$ and $\xi^X_t = \int_{t_0}^t X_s \mathrm{d}s$. Here, we can approximate arbitrarily well outputs of the form

$$\left\{ (X, t) \mapsto \int_{t_0}^t \Phi(X_{[s,t]}) \cdot X_s \mathrm{d}s \right\} \tag{5.60}$$

where Φ is any continuous function of the input path, restricted to the portion $[s, t]$. This clearly shows that dense Linear NCDEs are capable of context-dependent filtering: the output is again a linear combination of previously seen inputs, but weights are not predetermined as in linear RNNs like S4. However, in S4D and Mamba, the A^i are constrained to be diagonal, and this severely restricts the expressivity.

Theorem 5.8. *If the A^i are diagonal, then the requirements $\omega_t^{X,1} \equiv t$, $\omega_t^{X,2} \equiv t^2$ can be dropped and the existence result only holds with*

$$F \in \left\{ (X, t) \mapsto \psi(\omega_t^X) \cdot X_{t_0} + \int_{t_0}^{t} \phi(\omega_t^X - \omega_s^X) \cdot \mathrm{d}\xi_s^X \right\} \tag{5.61}$$

for continuous functions $\psi : \mathbb{R}^{d_\omega} \to \mathbb{R}^d$ and $\phi : \mathbb{R}^{d_\omega} \to \mathbb{R}^{d_\xi}$.

Proof. See [Cirone et al. 2024, Appendix B]. □

Taking $\omega_t^X = X_t$ and $\xi_t^X = \int_{t_0}^t X_s \, \mathrm{d}s$, dense matrices filter based on the entire trajectory $X_{[s,t]}$,

$$\int_{t_0}^{t} \Phi(X_{[s,t]}) \cdot X_s \, \mathrm{d}s \tag{5.62}$$

where diagonal matrices restrict you to comparing two elements of the input sequence,

$$\int_{t_0}^{t} \phi(X_t - X_s) \cdot X_s \, \mathrm{d}s \tag{5.63}$$

The key difference is that dense matrices allow hidden coordinates to interact, which lets the recurrence recursively build higher-order features of the path. When the A^i are diagonal, each hidden coordinate evolves in isolation, so these recursive interactions are lost. As a result, diagonal models cannot generate the same class of higher-order path features as dense Linear NCDEs. A smart choice of gating functions ω and ξ can still improve the resulting non-linear filtering strategy, but it cannot remove this fundamental processing discrepancy relative to the dense setting.

To give a simplistic example of the difference in expressivity between diagonal and dense state-transition matrices, consider a stream of bits

$$x_1, x_2, \ldots, \; x_n \in \{0, 1\}, \tag{5.64}$$

where we want to predict the parity label defined by

$$p_n = S_n \bmod 2 \in \{0, 1\}, \quad S_n = \sum_{k=1}^{n} x_k. \tag{5.65}$$

Whenever a new bit is 1 the label flips; if the bit is 0 the label stays the same. Taking a diagonal Linear NCDE with a hidden dimension of 2 and $\omega^x_{k+1} - \omega^x_k = x_{k+1}$, then

$$h_{n+1} = \exp\left(\begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix} x_{n+1}\right) h_n, \tag{5.66}$$

and

$$h^i_n = h^i_0 \exp(a_i S_n), \qquad i = 1, 2. \tag{5.67}$$

With a linear read-out $r = (r_1, r_2)^\top$ followed by a monotone activation ϕ (such as tanh, ReLU, sigmoid):

$$\hat{p}_n = \phi\Big(r^\top h_n\Big) = \phi\Big(r_1 h^1_0 e^{a_1 S_n} + r_2 h^2_0 e^{a_2 S_n}\Big) = \phi\Big(f(S_n)\Big). \tag{5.68}$$

Since $f(S)$ has at most one turning point, and ϕ is monotone, $\hat{p}_n$ can cross any chosen threshold at most twice. However, the true label p_n flips every time $S_n \mapsto S_n + 1$. Hence, no diagonal 2×2 Linear NCDE can realise parity on arbitrarily long input. Similarly, for a hidden dimension of n, $f(S)$ can have at most $n - 1$ turning points, so no diagonal Linear NCDE with a fixed hidden dimension can realise parity on arbitrarily long input. If you replace A with

$$A = \begin{pmatrix} 0 & \pi \\ -\pi & 0 \end{pmatrix}, \tag{5.69}$$

then

$$\exp(A x_{n+1}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{5.70}$$

when $x_{n+1} = 0$ and

$$\exp(A x_{n+1}) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{5.71}$$

when $x_{n+1} = 1$. Thus

$$h_{n+1} = (-1)^{x_{n+1}} h_n \tag{5.72}$$

and

$$h_n = (-1)^{S_n} h_0. \tag{5.73}$$

Taking $r = (1, 0)^\top$, $h^{(1)}_0 = 1$, and $\phi(s) = \big(1 - \mathrm{sign}(s)\big)/2$,

$$\hat{p}_n = \frac{1 - \mathrm{sign}\big((-1)^{S_n}\big)}{2} = S_n \bmod 2 = p_n. \tag{5.74}$$

Therefore, a dense Linear NCDE can solve parity exactly with a hidden dimension of 2.

In practice, it is possible to regain expressivity without sacrificing the computational advantages of diagonal matrices through stacking, where a new Linear NCDE is driven by the solution of a previous Linear NCDE. A formal statement and proof of the recovery of expressivity is given in Appendix C of Cirone et al. 2024. An important corollary of the results on stacking is that linear RNNs, such as S4, require non-linearities in-between layers to recover expressivity, whereas selective SSMs like Mamba only need linear mixing layers. Intuitively, stacking recovers the mixing between the hidden dimensions which is crucial for the expressiveness of dense Linear NCDEs.

5.3.4 Experiments

The first task considered is based on the toy dataset from Section 4.4.6, where the aim is to predict terms in the input path's signature. The dataset's objective aligns with the proofs of Theorems 5.4 and 5.5, which characterise the expressivity using the path's signature. We use two datasets with dimensions 2 and 3, respectively. The increment in each channel at each step is an integer-rounded sample from a standard Normal distribution,

$$p(n) = \int_{n-0.5}^{n+0.5} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \mathrm{d}x, \tag{5.75}$$

where $n \in \mathbb{Z}$. The 2D dataset's target is the area integral

$$\int_0^1 \int_0^v dX_u^1 dX_v^2, \tag{5.76}$$

and the 3D dataset's target is a volume integral

$$\int_0^1 \int_0^w \int_0^v dX_u^1 dX_v^2 dX_w^3. \tag{5.77}$$

We consider seven models on this dataset:

- $(1, 2)$: A single S4D or S6 recurrence with a linear readout,
- $(3, 4)$: Two stacked S4D or S6 recurrences with a linear mixing layer in-between and a linear readout,
- $(5, 6)$: Two stacked S4D or S6 recurrences with a linear mixing layer and ReLU in-between and a linear readout,
- 7: A Linear NCDE with gates $\omega_t^X = \xi_t^X = (t, X_t)$ and a linear readout.

All of the SSMs have trainable matrices in their recurrences, whereas the Linear NCDE is using fixed random matrices. All models use a hidden dimension of 256, with the SSMs using a state dimension of 256. The SSMs are trained using gradient descent with a batch size of 32 and Adam with a learning rate of 10^{-4} [Kingma et al. 2015]. The output from the Linear NCDE's recurrence is obtained using the Tsit5 adaptive ODE solver, with an absolute and relative tolerance of 10^{-2} [Tsitouras 2011]. The Linear NCDE's linear readout is obtained via ordinary least squares.

The second task we consider is the A_5 benchmark from Merrill et al. [2024]. It tests models on state-tracking, a crucial ability for tasks involving permutation composition, such as chess. The dataset comprises sequences from the group of even permutations on five elements, A_5, where the target is the cumulative composition of all preceding permutations. Datasets vary by sequence length, ranging from 3 to 20. The models are evaluated on the number of stacked layers required to achieve greater than 90% validation accuracy.

We consider six models on the A_5 benchmark: a Linear NCDE, a diagonal Linear NCDE, an LSTM, a Transformer, S4D, and Mamba. The LSTM, Transformer, S4D, and Mamba use a hidden dimension of 1024. LSTM uses direct stacking whereas the other baseline models use stacked blocks consisting of a sequence model, a GLU layer [Dauphin et al. 2017], and layer normalisation [Ba et al. 2016]. The Linear NCDEs take

$$\frac{\omega^X_{t_{j+1}} - \omega^X_{t_j}}{t_{j+1} - t_j} = (1, X_{t_j}) \tag{5.78}$$

and $\xi^X_t = 0$. Furthermore, they use 1024 trainable parameters per recurrence, corresponding to a hidden dimension of 1024 for the diagonal Linear NCDE and 32 for the Linear NCDE. Models are trained using a token-tagging loss for 1,000,000 steps with a batch size of 256. For all sequence lengths, a small batch of sequences of length 2 are included at each training step to aid convergence. All models have trainable matrices in their recurrences and use Adam with weight decay as the optimiser [Kingma et al. 2015], alongside a linear warm-up followed by cosine annealing with a minimum learning rate of 10^{-5} and a maximum learning rate of 10^{-3}. Additionally, all models use dropout with a rate of 0.1 and a trainable embedding layer [Srivastava et al. 2014].

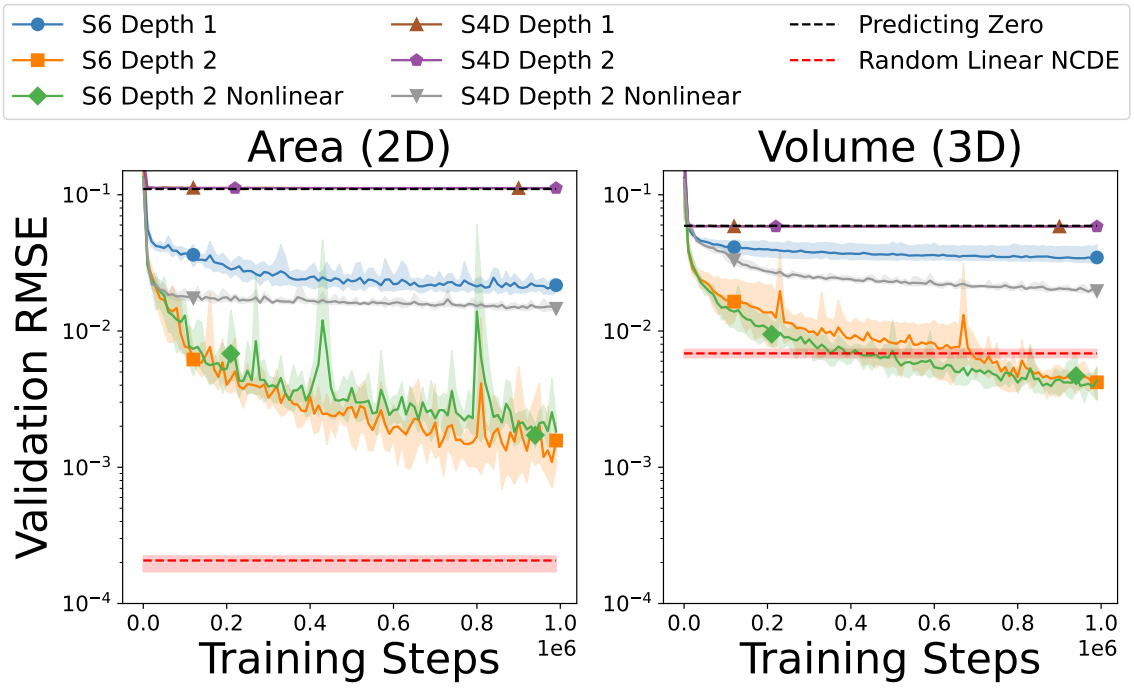


Figure 5.1: Comparison of the Linear NCDE, S4D, and S6 on the two signature prediction tasks, (5.76) and (5.77). For each model, we plotted the mean and range of the validation root mean square error over 5 independent runs.

5.3.5 Results

Figure 5.1 presents the results on the signature prediction task. These results empirically demonstrate a number of the theoretical results presented in the previous sections. Firstly, as discussed in Section 5.3.3, recurrences which are linear in the input, such as S4D, require a non-linearity in-between the recurrent layers to perform well. Furthermore, as stated in Theorem 5.8, even if the recurrence is non-linear in the input, such as Mamba's recurrence S6, the expressivity of models with diagonal matrices is improved by stacking. Additionally, the inclusion of the non-linearity in-between the S6 layers does not improve performance, as the recurrences themselves are expressive enough. Finally, as stated in Theorem 5.4, dense matrices can achieve strong expressivity with random initialisation, no stacking, and only a trainable linear readout.

Figure 5.2 is a plot of the results on the A_5 benchmark. The figure shows that the number of blocks S4D, Mamba, and the Linear NCDE with diagonal state-transition matrices require to achieve greater than 90% validation accuracy grows with the se-

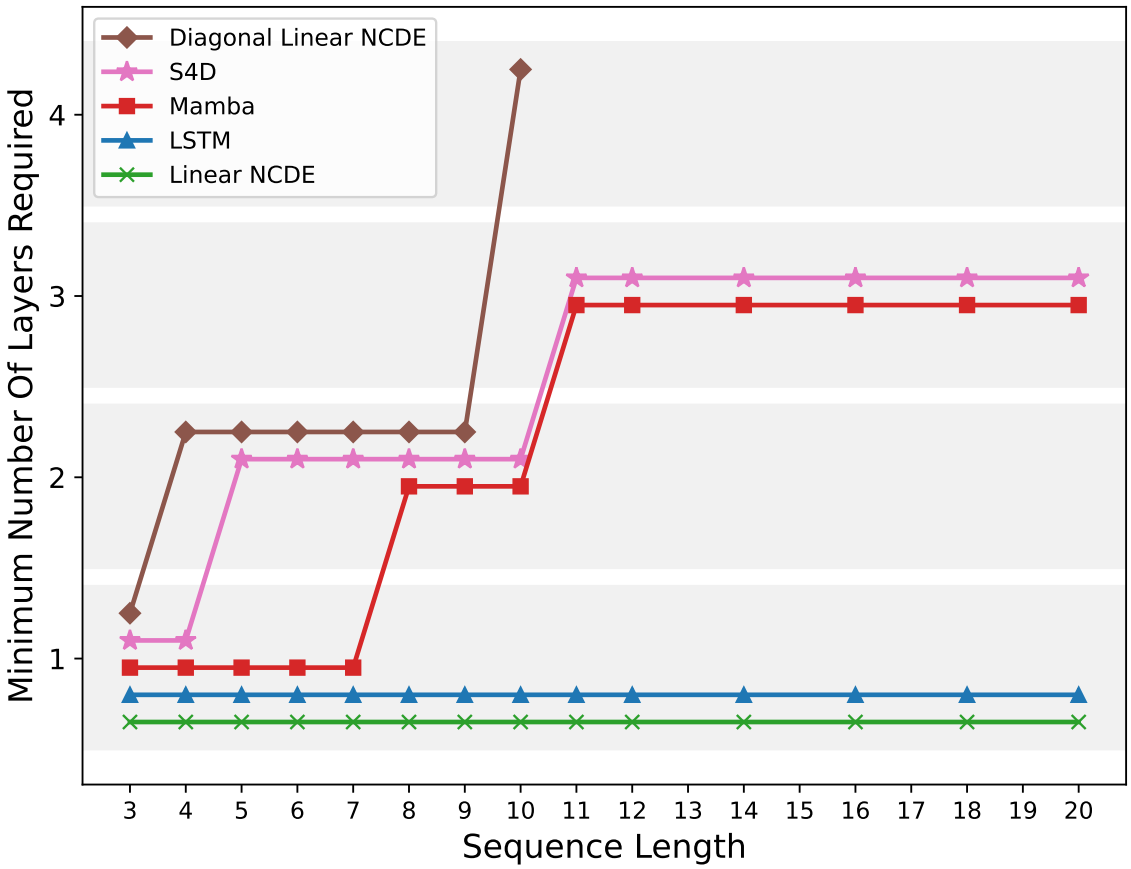


Figure 5.2: Comparison of a diagonal Linear NCDE, S4D, Mamba, LSTM, and Linear NCDE on the A_5 benchmark. For each sequence length, the plot shows the minimum number of blocks required to achieve at least 90% validation accuracy, with each shaded band corresponding to a number of blocks. Missing points mean the model did not achieve at least 90% validation accuracy with 4 blocks or less.

quence length. On the other hand, the LSTM and Linear NCDE are able to achieve greater than 90% validation accuracy for all lengths considered using only one block. These empirical results further validate Theorem 5.8: there exists a gap in expressivity between diagonal and dense state-transition matrices, with stacking required to recover expressivity. Furthermore, they provide empirical evidence that even for simple state-tracking problems, the number of stacked blocks required to recover expressivity can grow quickly with sequence length.

5.3.6 Limitations

As shown in Section 5.2.5, replacing the non-linear vector field of an NCDE with a linear vector field leads to substantial improvements in time per training step. Furthermore, as shown theoretically in Section 5.3.3 and empirically in Section 5.3.5, using dense state-transition matrices A_θ^i provides more expressivity than the diagonal state-transition matrices of S4D and Mamba. However, there remains a core limitation of Linear NCDEs. The number of parameters and computational cost scales as $\mathcal{O}(d_h^3)$, where d_h is the hidden dimension. This is in contrast to diagonal state-transition matrices, where they scale as $\mathcal{O}(d_h^2)$. This makes Linear NCDEs infeasible in large models, where hidden dimensions can reach 8192 [Touvron et al. 2023].

This limitation motivates the following question: Do there exist structured matrices A_θ^i which are computationally more efficient than dense matrices, whilst having the same theoretical expressivity and comparable empirical performance? It is this question we aim to answer in Section 5.4.

5.4 Structured Linear NCDEs

5.4.1 Introduction

Structured Linear Neural Controlled Differential Equations (SLiCEs) are Linear NCDEs

$$h_t = h_{t_0} + \int_{t_0}^{t} \sum_{i=1}^{d_\omega} A_\theta^i h_s \mathrm{d}\omega_s^X, \tag{5.79}$$

where each A_θ^i is constrained to have a particular structure. While using diagonal matrices is computationally efficient, Section 5.3 showed that this choice theoretically limits expressivity and empirically hurts performance on state-tracking tasks. This

section explores four more powerful alternatives, two inspired by prior work and two that are novel.

- Block-diagonal: The matrix is composed of smaller, dense blocks along the diagonal,

$$A_\theta^i = \text{BlockDiag}\big(B_{\theta,1}^i, B_{\theta,2}^i, \ldots, B_{\theta,k}^i\big), \tag{5.80}$$

  where each $B_{\theta,j}^i \in \mathbb{R}^{b_j \times b_j}$ is a trainable dense block, k is the number of blocks, and $d_h = \sum_{j=1}^k b_j$. This structure is inspired by the input-dependent block-diagonal linear RNN [Fan et al. 2024]

- Diagonal-plus-low-rank (DPLR): The matrix is the sum of a diagonal matrix and a low-rank matrix,

$$A_\theta^i = D_\theta^i + u_\theta^i \big(v_\theta^i\big)^\top, \tag{5.81}$$

  where D_θ^i is diagonal and $u_\theta^i, v_\theta^i \in \mathbb{R}^{d_h \times r}$ with rank $r < d_h$. DPLR structures are used by DeltaNet, DeltaProduct, and Gated DeltaNet [Schlag et al. 2021; Yang et al. 2024b; Siems et al. 2025; Yang et al. 2025].

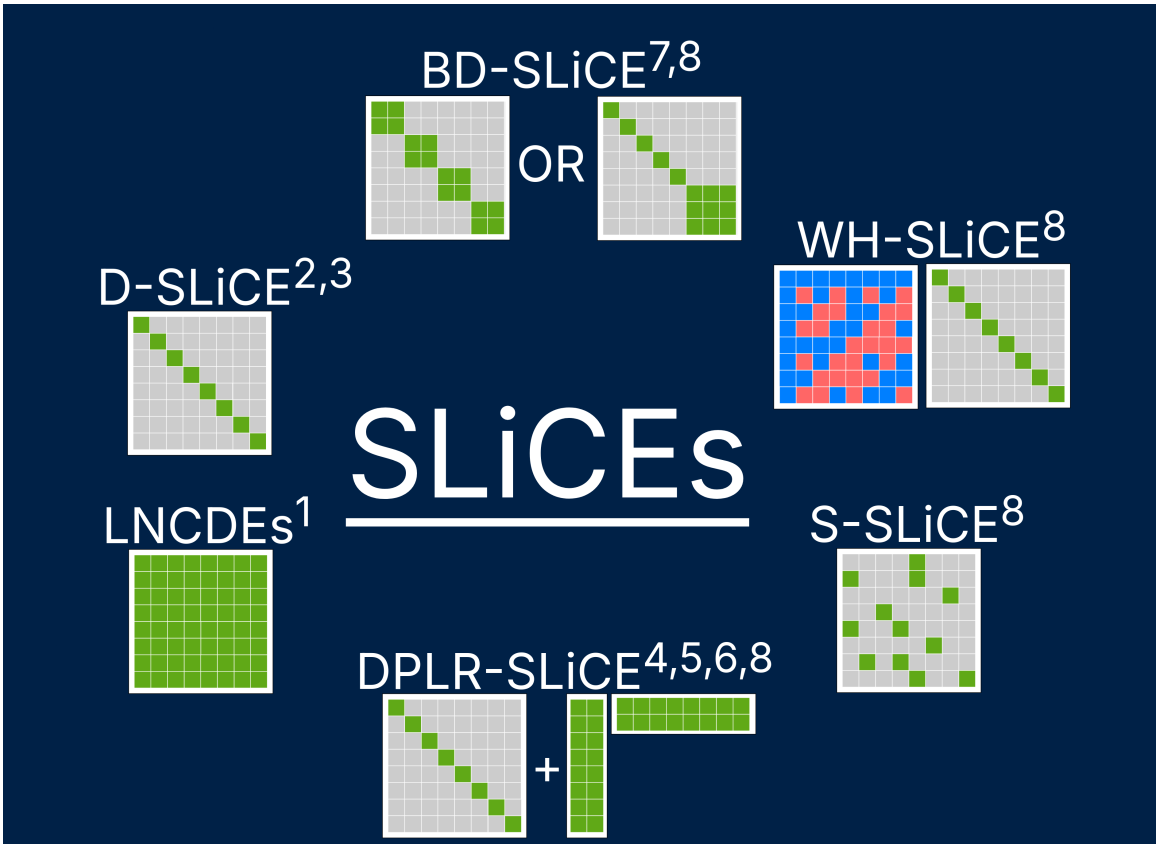
- Sparse: Each A_θ^i is a sparse matrix with $\mathcal{O}(d_h^{1+\epsilon})$ non-zero entries for some $0 < \epsilon < 1$, sampled at random via a Bernoulli mask.

- Walsh–Hadamard: The matrix is the product of a fixed matrix H and a trainable diagonal matrix D_θ^i,

$$A_\theta^i = H D_\theta^i. \tag{5.82}$$

  where H is a Hadamard matrix of order d_h (entries ± 1 with mutually orthogonal columns and rows).

Figure 5.3 is a visual comparison of the structures. Compared to dense matrices, all four proposed structures significantly reduce both the parameter count and the computational cost of a recurrent update. Additionally, the Linear NCDE framework can be used to prove that unlike diagonal matrices, all four structures possess maximal probabilistic expressivity, as will be discussed further in Section 5.4.3. Empirical results in Section 5.4.6 demonstrate that all four structures successfully perform state-tracking on the A_5 benchmark used in Section 5.3.4. Furthermore, the block-diagonal and DPLR SLiCE establish a new state-of-the-art among parallel-in-time models for length generalisation on regular language tasks [Delétang et al. 2023].



1. Cirone, N. M. et al. (2024). Theoretical foundations of deep selective state-space models. NeurIPS.
2. Gu, A. et al. (2022). On the Parameterization and Initialization of Diagonal State Space Models. NeurIPS.
3. Gu, A. et al. (2024). *Mamba: Linear-time sequence modeling with selective state spaces. COLM.*
4. Schlag, I. et al. (2021). Linear transformers are secretly fast weight programmers. ICML.
5. Yang, S. et al. (2024). Parallelizing linear transformers with the delta rule over sequence length. NeurIPS.
6. Siems, J. et al. (2025). DeltaProduct: Improving state-tracking in linear RNNs via Householder products. ICLR.
7. Fan, T. et al. (2024). Advancing regular language reasoning in linear recurrent neural networks. NAACL.
8. Walker, B. et al. (2025). Structured Linear CDEs: Maximally Expressive and Parallel-in-Time Models. NeurIPS.

Figure 5.3: A visualisation of the structure for one A_θ^i when $d_h = 8$ in a diagonal (D), block-diagonal (BD), Walsh–Hadamard (WH), sparse (S), and diagonal-plus-low-rank (DPLR) SLiCE. Green entries represent trainable parameters, grey represent parameters fixed at zero, blue represent parameters fixed at 1, and red represent parameters fixed at -1.

5.4.2 Related Work

The block-diagonal and DPLR SLiCEs take inspiration from block-diagonal input-dependent linear RNN and DeltaNet, respectively [Fan et al. 2024; Schlag et al. 2021; Yang et al. 2024b]. Both models are linear RNNs, in the sense that their recurrent update is linear in the previous hidden state. For example, block-diagonal input dependent linear RNN is defined by

$$h_{t_{i+1}} = A_\theta(x_{t_i})h_{t_i} + B_\theta x_{t_i}, \tag{5.83}$$

where A_θ is a block-diagonal matrix that depends non-linearly on x_{t_i}. In contrast, the state-transition matrix of block-diagonal SLiCE depends linearly on x_{t_i}. As shown in Section 5.4.3 and 5.4.6, a linear dependence is theoretically sufficient for expressivity and empirically effective for state tracking.

DeltaNet belongs to a subset of linear RNNs whose hidden states are matrix-valued. This perspective originated as an alternative viewpoint on linear Transformers, where softmax is replaced with a kernel admitting a finite-dimensional feature map, $\kappa(q,k) = \phi(q)^\top\phi(k)$ for $\phi : \mathbb{R}^{d_q} \to \mathbb{R}^{d_\phi}$. Letting $W_\theta^q \in \mathbb{R}^{d_q \times d_x}$, $W_\theta^k \in \mathbb{R}^{d_q \times d_x}$, and $W_\theta^v \in \mathbb{R}^{d_v \times d_x}$ be learnable weights, then causal linear attention is defined by

$$o_{t_i} = \frac{\sum_{j=1}^{i} \phi\big(W_\theta^q x_{t_i}\big)^\top \phi\big(W_\theta^k x_{t_j}\big)\, W_\theta^v x_{t_j}}{\sum_{j=1}^{i} \phi\big(W_\theta^q x_{t_i}\big)^\top \phi\big(W_\theta^k x_{t_j}\big)}. \tag{5.84}$$

As shown by Katharopoulos et al. [2020], this admits an RNN formulation,

$$\begin{aligned} S_{t_i} &= S_{t_{i-1}} + \phi\big(W_\theta^k x_{t_i}\big)\big(W_\theta^v x_{t_i}\big)^\top, \\ y_{t_i} &= y_{t_{i-1}} + \phi\big(W_\theta^k x_{t_i}\big), \\ o_{t_i} &= \frac{\phi\big(W_\theta^q x_{t_i}\big)^\top S_{t_i}}{\phi\big(W_\theta^q x_{t_i}\big)^\top y_{t_i}}, \end{aligned} \tag{5.85}$$

where $S_i \in \mathbb{R}^{d_\phi \times d_v}$ with $S_0 = 0$, $y_i \in \mathbb{R}^{d_\phi}$ with $y_0 = 0$, and $o_i \in \mathbb{R}^{d_v}$. Ignoring the normalisation, the core recurrence is a matrix-valued linear RNN.

Matrix-valued linear RNNs are a useful framework for understanding several recent sequence models, as highlighted by Yang et al. [2024b]. Many of these models share the general form:

$$S_{t_{i+1}} = S_{t_i} \bullet M_{t_i} + v(x_{t_i})k(x_{t_i})^\top, \tag{5.86}$$

where $\bullet$ denotes an associative operator and v and k are arbitrary functions of the current input with compatible output dimensions. In practice, they are often linear projections, with possible additions such as feature maps, element-wise non-linearities, or normalisation. In Section 5.3.2, the hidden state for each channel of Mamba's input was stacked vertically to view the entire model as a Linear NCDE. If instead you view the hidden state for each channel of the input as columns of a matrix, then Mamba's recurrence (5.24) can be rewritten as

$$S_{t_{i+1}} = S_{t_i} \odot M_{t_i} + v(x_{t_i})k(x_{t_i})^\top, \tag{5.87}$$

where $\odot$ refers to the Hadamard (element-wise) product and $S_{t_i} \in \mathbb{R}^{d_h \times d_x}$. The Hadamard product is a direct consequence of Mamba's diagonal state-transition matrix, which inherently prevents interaction between individual elements of the hidden state. This framework gives a clear interpretation for one of the key modifications introduced by Mamba-2,

$$S_{t_{i+1}} = \gamma(x_{t_i})S_{t_i} + v(x_{t_i})k(x_{t_i})^\top, \tag{5.88}$$

where γ is a real-valued function [Dao et al. 2024]. Additionally, it highlights the structural similarity between Mamba and linear Transformers.

Replacing the Hadamard product by a matrix product allows richer interactions between the hidden state,

$$S_{t_{i+1}} = S_{t_i}M_{t_i} + v(x_{t_i})k(x_{t_i})^\top. \tag{5.89}$$

However, similarly to using dense matrices in a Linear NCDE, the cost makes layers of this type intractable in larger models. DeltaNet uses a diagonal-plus-rank-one structure,

$$S_{t_{i+1}} = S_{t_i}(I - \beta(x_{t_i})k(x_{t_i})k(x_{t_i})^\top) + \beta(x_{t_i})v(x_{t_i})k(x_{t_i})^\top, \tag{5.90}$$

where β is a real-valued function [Schlag et al. 2021; Yang et al. 2024b]. DeltaProduct later generalised DeltaNet to DPLR matrices [Siems et al. 2025]. Beyond reducing parameter count and recurrent computational cost, this structure also facilitates an efficient chunk-wise algorithm that can outperform parallel associative scans for large hidden dimensions, as outlined in [Yang et al. 2024b, Section 3.2]. As will be seen in Section 5.3.4, it also significantly improves state-tracking performance.

Many other recent sequence models can also be viewed as matrix-valued linear RNNs. These include, but are not limited to, Gated DeltaNet [Yang et al. 2025], RWKV-7 [Peng et al. 2025], HGRN-2 [Qin et al. 2024], mLSTM [Beck et al. 2024], Gated Linear Attention [Yang et al. 2024a], Gated Random Feature Attention [Peng et al. 2021], Gated Slot Attention [Zhang et al. 2024], TTT-Linear [Sun et al. 2025], and Titans [Behrouz et al. 2025]. A detailed comparison of the specific form of (5.86) for many of these models is provided in Table 2 of [Yang et al. 2024b]. Beck et al. [2024] introduced mLSTM alongside a non-linear recurrent model sLSTM, which together form their sequence model xLSTM. These components are designed to play different roles, with mLSTM acting as the memory and sLSTM performing the reasoning. Section 5.4.6 will use mLSTM, sLSTM, and xLSTM as baseline methods to highlight the different roles the components play.

Matrix-valued linear RNNs can be converted into vector-valued linear RNNs by returning to the stacking approach of Section 5.3.2. Let $\mathrm{vec} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ be the column-major vectorisation operator, which transforms a matrix into a vector by stacking its columns. Letting $\otimes$ denote the Kronecker product, the Hadamard product linear RNN (5.87) can be rewritten as

$$\mathrm{vec}(S_{t_{i+1}}) = \mathrm{diag}(\mathrm{vec}(M_{t_i}))\,\mathrm{vec}(S_{t_i}) + k(x_{t_i}) \otimes v(x_{t_i}), \tag{5.91}$$

where the state-transition matrix is diagonal, consistent with Mamba's design. Noting that

$$\mathrm{vec}(AXB) = (B^\top \otimes A)\,\mathrm{vec}(X), \tag{5.92}$$

then the linear RNN with a matrix product (5.89) can be written as

$$\mathrm{vec}(S_{t_{i+1}}) = (M_{t_i}^\top \otimes I_{d_h})\,\mathrm{vec}(S_{t_i}) + k(x_{t_i}) \otimes v(x_{t_i}). \tag{5.93}$$

In both cases, a matrix-valued recurrence is equivalent to a vector-valued recurrence on a flattened hidden state, where the state-transition matrix is constrained to have a specific structure. For the Hadamard product, this gives a diagonal matrix, while the matrix product leads to a Kronecker product structure.

For the remainder of this work, we will focus our analysis on the vector-valued case with the structured matrices introduced in Section 5.4.1 (block-diagonal, DPLR, sparse, and Walsh-Hadamard). The insights gained from this analysis naturally extend to the matrix-valued setting. For instance, the limitations in expressivity of

diagonal matrices demonstrated in Section 5.3 directly translate to the Hadamard product formulation. Similarly, the Kronecker product structure arising from vectorising a matrix-valued recurrence clearly illustrates how the properties of the constituent matrices determine the properties of the overall state transition.

SLiCEs can be generalised to matrix-valued hidden states by selecting a suitable structure and reversing the above vectorisation. Alternatively, m copies of the same SLiCE can be run via

$$H_t = H_{t_0} + \int_{t_0}^{t} \left(\sum_{i=1}^{d_\omega} A_\theta^i \, \mathrm{d}\omega_s^{X,i} \right) H_s, \tag{5.94}$$

where $H_s \in \mathbb{R}^{d_h \times m}$. Clearly, m copies of a vector-valued SLiCE match the expressivity of a single copy, so all of our theoretical results naturally carry over to this setting. However, the columns only differ due to their initial conditions. To introduce meaningful column-specific dynamics, you can include a column-specific bias term in the vector field. A natural way to incorporate such biases into a matrix-valued SLiCE is to consider an affine SLiCE,

$$H_t = H_{t_0} + \int_{t_0}^{t} \left(\sum_{i=1}^{d_\omega} A_\theta^i \, \mathrm{d}\omega_s^{X,i} \right) H_s + B_\theta \, \mathrm{diag}(\mathrm{d}\xi_s^X), \tag{5.95}$$

where $B_\theta \in \mathbb{R}^{d_h \times m}$. Letting h_t^k for $1 \leq k \leq m$ denote the columns of H_t, then

$$h_t^k = h_{t_0}^k + \int_{t_0}^{t} \left(\sum_{i=1}^{d_\omega} A_\theta^i \, \mathrm{d}\omega_s^{X,i} \right) h_s^k + B_\theta^k \mathrm{d}\xi_s^{X,k}. \tag{5.96}$$

Approximating ω_s and ξ_s with linear interpolation on the grid $t_0 < \cdots < t_n$ yields

$$\tilde{h}_{t_{j+1}}^k \approx \exp\left(\sum_{i=1}^{d_\omega} A_\theta^i \left(\omega_{t_{j+1}}^{X,i} - \omega_{t_j}^{X,i} \right) \right) \tilde{h}_{t_j}^k + B_\theta^k (\xi_{t_{j+1}}^{X,k} - \xi_{t_j}^{X,k}), \tag{5.97}$$

where we have used the approximation

$$\int_0^{t_{j+1}-t_j} \exp\left(\tau \sum_{i=1}^{d_\omega} A_\theta^i \frac{\omega_{t_{j+1}}^{X,i} - \omega_{t_j}^{X,i}}{t_{j+1} - t_j} \right) \mathrm{d}\tau \approx (t_{j+1} - t_j) I, \tag{5.98}$$

for small increments. The outputs $\tilde{h}_{t_j}^k$ remain computable in $\mathcal{O}(\log(n))$ parallel steps using a parallel associative scan [Blelloch 1993].

This construction is closely related to the path development layer, whose hidden state evolves on a matrix Lie group via the recurrence

$$H_{t_{j+1}} = \exp\big(M_\theta(x_{t_{j+1}} - x_{t_j})\big)\, H_{t_j}, \tag{5.99}$$

where M_θ is a linear map into a matrix Lie algebra $\mathfrak{g}$ [Lou et al. 2024; Jiang et al. 2024]. Any such linear map can be written coordinate-wise as

$$M_\theta(v) = \sum_{i=1}^{d_\omega} A_\theta^i v^i, \tag{5.100}$$

where the choice of $\mathfrak{g}$ determines the structure of the matrices $A_\theta^i \in \mathfrak{g}$. Hence, this is a special case of (5.95) with $\xi_s^X = 0$, $H_{t_0} = I$, and $\omega_s^X = X_s$, where X_s is the linear interpolation of $\{x_{t_i}\}_{i=0}^n$. Therefore, the scan-based methods developed here can also be applied to parallelise the computation of the path development layer.

5.4.3 Expressiveness

As demonstrated in Section 5.3.3, a core difference between diagonal and dense state-transition matrices A_θ^i is their theoretical expressivity. Here, we show that all four of the proposed SLiCE architectures retain maximal probabilistic expressivity.

Applying Lemma 5.6 with $B = 0$, the solution to (5.79) is

$$h_t = \sum_{I \in \mathcal{I}} S^I_{[t_0,t]}(\omega) A^I h_0, \tag{5.101}$$

where $\mathcal{I}$ is the set of multi-indices

$$\mathcal{I} = \{\emptyset\} \cup \{I | I = (i_1, \ldots, i_k), k \in \mathbb{N}, 1 \le i_j \le d_\omega\} \tag{5.102}$$

$A^I = A^{i_k} \cdots A^{i_1}$ with $A^\emptyset$ being the identity, and $S^I_{[s,t]}(\omega)$ is the term in the signature of ω over $[s,t]$ corresponding to the multi-index I,

$$S^I_{[s,t]}(\omega) = \underbrace{\int \cdots \int}_{s \le u_1 < \cdots < u_k \le t} \mathrm{d}\omega^{i_1}_{u_1} \cdots \mathrm{d}\omega^{i_k}_{u_k}, \tag{5.103}$$

with $S^\emptyset = 1$. The proof of maximal probabilistic expressivity relies on showing that the feature vectors $A^I h_0$ and $A^J h_0$ become approximately orthogonal as the hidden dimension increases. This property is guaranteed if the following bound holds:

$$\left\| \frac{1}{d_h} \langle A^I h_0, A^J h_0 \rangle_{\mathbb{R}^{d_h}} - \delta_{I,J} \right\|_{L^2(\mathbb{P}_{d_h})} \le (\kappa(|I| + |J|))!!\, \mathcal{O}\left(\frac{1}{\sqrt{d_h}}\right), \tag{5.104}$$

where κ is a constant independent of d_h. Cirone et al. [2023] showed that (5.104) holds when A^i_θ have i.i.d entries from $\mathcal{N}\left(0, \frac{1}{d_h}\right)$ and this fact was used by Cirone et al. [2024] to prove Theorem 5.4. The proof that each of the four SLiCE structures considered possesses maximal probabilistic expressivity proceeds by demonstrating that (5.104) holds for a distribution appropriate to that structure.

The detailed proofs of these results were primarily developed by a collaborator, Nicola Muça Cirone. The proofs build on their work developing a novel graphical framework for neural networks that linearises the effect of activation functions, allowing for the application of Wick's principle and the genus expansion technique when proving convergence results [Cirone et al. 2025a]. This framework facilitates simple proofs that (5.104) holds for all four SLiCE structures, but introducing the required machinery falls outside the scope of this thesis. Here, we present the expressivity results for the four structures with references to the locations of detailed proofs.

Theorem 5.9. *Let $\mathcal{X}$ be the space of bounded variation paths on the interval $[t_0, t_n]$ that start at a common point and include time as a channel, endowed with the $1-$variation topology. Let $\mathcal{F}$ be the space of SLiCEs with $d_h \in \mathbb{N}$. Then $\mathcal{F}$ has maximal probabilistic expressivity when $h_0 \sim \mathcal{N}(0,1)$ and A^i_θ satisfies one of the following conditions:*

1. *$A^i_\theta = \text{BlockDiag}\big(B^i_{\theta,1}, B^i_{\theta,2}, \ldots, B^i_{\theta,k}\big)$, where $B^i_{\theta,j} \in \mathbb{R}^{b\times b}$, $b = \lceil \log(d_h) \rceil$, and $B^i_{\theta,j}$ has i.i.d entries from $\mathcal{N}\left(0, \frac{1}{b}\right)$.*
2. *$A^i_\theta = u^i_\theta \big(v^i_\theta\big)^\top$, where $u^i_\theta, v^i_\theta \in \mathbb{R}^{d_h \times r}$ with $r = \lceil \log(d_h) \rceil$ and u^i_θ and v^i_θ have i.i.d entries from $\mathcal{N}(0,1)$.*
3. *A^i_θ has entries obtained by pointwise multiplying W with i.i.d entries from $\mathcal{N}(0, \frac{1}{d_h p(d_h)})$ and B with i.i.d entries from a Bernoulli distribution with probability $p(d_h)$ of being 1, where $p(d_h)$ satisfies $d_h p(d_h) \to \infty$ as $d_h \to \infty$.*
4. *$A^i_\theta = HD^i_\theta$, where $H \in \mathbb{R}^{d_h \times d_h}$ is a fixed matrix with entries bounded uniformly in d_h by a constant C satisfying $HH^\top = d_h I_{d_h}$ and D^i_θ is a diagonal matrix with i.i.d entries from $\mathcal{N}(0,1)$.*

Proof. The proof of (1), (3), and (4) can be found as Proposition B.6, B.2, B.4 in [Walker et al. 2025], respectively. The proof of (2) can be found as Proposition F.2 in [Movahedi et al. 2025]. □

The expressivity result for the low-rank structure (2) naturally extends to a DPLR structure, by simply taking $D_\theta^i = 0$. Additionally, condition (3) shows that maximal probabilistic expressivity in the sparse case is retained only when the matrices do not become too sparse, in the sense that the expected number of non-zero entries per row must diverge as $d_h \to \infty$. For instance, sparse matrices with $\mathcal{O}(d_h^{1+\epsilon})$ non-zero entries for some $\epsilon > 0$ satisfy this condition. Furthermore, a Hadamard matrix matches the conditions of (4). Therefore, Theorem 5.9 confirms that, unlike diagonal matrices, SLiCEs with block-diagonal, DPLR, sparse, and Walsh–Hadamard matrices have maximal probabilistic expressivity.

5.4.4 Comparison

Table 5.2 summarises the differences in parameter count, computational cost, existence of an efficient implementation, and expressivity of the proposed SLiCE structures, where for simplicity we have taken $d_\omega = d_h$. As can be seen, the four proposed structures lead to a reduction in parameter count and recurrent cost of inference, whilst still maintaining maximal expressivity. However, current deep-learning frameworks, such as JAX [Bradbury et al. 2018] and PyTorch [Paszke et al. 2019], are not optimised for unstructured sparsity, so the sparse structure does not lead to practical speed-ups in our implementations. Furthermore, not all the structures reduce the theoretical computational cost of applying a parallel associative scan. This is because the parallel associative scan is repeatedly composing

$$\exp\left(\sum_{i=1}^{d_\omega}(\omega_{t_{j+1}}^i - \omega_{t_j}^i)A_\theta^i\right). \tag{5.105}$$

When the A_θ^i are diagonal or block-diagonal, the composition of (5.105) preserves the structure, as these classes of matrices are closed under multiplication. Therefore, using a parallel associative scan reduces the scan depth from n to $\log(n)$, whilst having a computational cost per composition of $\mathcal{O}(d_h^2)$ or $\mathcal{O}(d_h \sum_j b_j^2)$, respectively. However, for DPLR, sparse, and Walsh–Hadamard SLiCEs, the structured matrices are not closed under multiplication, which means that the limiting computational cost per composition is the same as a dense Linear NCDE, $\mathcal{O}(d_h^3)$.

As discussed in Section 5.2.2, parallel associative scans result in high I/O costs for large models, as each state-transition matrix must be materialised in GPU memory [Yang et al. 2024b]. Yang et al. [2024b] introduced an alternative approach

Table 5.2: Comparison of SLiCEs with dense, diagonal, diagonal-plus-low-rank (DPLR), sparse, Walsh–Hadamard (WH), and block-diagonal (BD) structure on parameter count, computational cost, the existence of an efficient implementation, and expressivity. Here, d_h is the hidden dimension, n is the sequence length, b_j are BD-SLiCE's block-sizes, r is DPLR-SLiCE's rank, ϵ is S-SLiCE's sparsity, and for simplicity we have taken $d_\omega = d_h$, as is common in practice when stacking layers. Parallel cost is measured as $\mathcal{O}$(scan depth, cost per composition) when applying a parallel associative scan.

Structure	**Parameters**	**Recurrent Cost**	**Parallel Cost**	**Efficient Impl.**	**Maximally Expressive**
Dense	$\mathcal{O}(d_h^3)$	$\mathcal{O}(nd_h^3)$	$\mathcal{O}(\log(n), d_h^3)$	Yes	Yes
Diagonal	$\mathcal{O}(d_h^2)$	$\mathcal{O}(nd_h^2)$	$\mathcal{O}(\log(n), d_h^2)$	Yes	No
DPLR	$\mathcal{O}(rd_h^2)$	$\mathcal{O}(nrd_h^2)$	$\mathcal{O}(\log(n), d_h^3)$	Yes	Yes
Sparse	$\mathcal{O}(d_h^{2+\epsilon})$	$\mathcal{O}(nd_h^{2+\epsilon})$	$\mathcal{O}(\log(n), d_h^3)$	No	Yes
WH	$\mathcal{O}(d_h^2)$	$\mathcal{O}(nd_h^2)$	$\mathcal{O}(\log(n), d_h^3)$	Yes	Yes
BD	$\mathcal{O}\left(d_h \sum_j b_j^2\right)$	$\mathcal{O}\left(nd_h \sum_j b_j^2\right)$	$\mathcal{O}\left(\log(n), d_h \sum_j b_j^2\right)$	Yes	Yes

for DeltaNet, where a chunk-wise algorithm specifically tailored for diagonal-plus-rank-one state-transition matrices is used to bypass the need to materialise every intermediate matrix, significantly cutting down I/O costs [Yang et al. 2024b]. These approaches can also be applied to diagonal state-transition matrices. Therefore, a block-diagonal SLiCE with a large diagonal portion ($b_i = 1$ for $i = 1, \ldots, k-1$) followed by a small dense block emerges as an attractive solution. The large diagonal section can efficiently utilise the chunk-wise algorithm and the smaller dense section can be processed using parallel associative scans without incurring significant I/O costs. We refer to this structure as diagonal-dense SLiCE (D-DE-SLiCE).

5.4.5 Implementation Details

Algorithm 1 provides a pseudo-code implementation for the forward pass of a SLiCE. The approach is demonstrated for dense state-transition matrices A_θ^i, with comments highlighting where a SLiCE's structure can be used to speed up computation or reduce memory footprint. In this thesis, each SLiCE recurrence is embedded in a simple block structure, combining a linear layer to mix the channels, tanh activation function, layer normalisation [Ba et al. 2016], and a skip connection. Inspired by the

Lip(γ) regularisation introduced in Section 4.4.2, all SLiCEs use weight regularisation on their state-transition matrices A_θ^i. Additionally, due to instability during training, Walsh–Hadamard's diagonal matrix D_θ^i is parametrised to take values between -1 and 1. Finally, to improve training speed the following approximation is made,

$$\exp\left(\sum_{i=1}^{d_\omega}(\omega_{t_{j+1}}^i - \omega_{t_j}^i)A_\theta^i\right) \approx I + \sum_{i=1}^{d_\omega}(\omega_{t_{j+1}}^i - \omega_{t_j}^i)A_\theta^i. \tag{5.106}$$

This is equivalent to applying an Euler discretisation with step-size 1, and hence aligns with the approach of recurrent models. There is ongoing work to develop an efficient GPU kernel to reduce the computational burden of repeated matrix exponentials, as will be discussed further in Section 5.5.

Algorithm 1 Structured Linear NCDE: The algorithm is presented for a dense state-transition matrix A_θ. Comments indicate where the structure of A_θ can be used to reduce memory footprint and speed-up computation.

Input: $\boldsymbol{\omega} : (B,\, L,\, d_\omega)$

Output: $\mathbf{h} : (B,\, L,\, d_h)$

$\mathbf{h}_0 : (B,\, d_h) \leftarrow \xi_\phi(\boldsymbol{\omega}_0)$

$\boldsymbol{\omega}^{\text{inc}} : (B,\, L-1,\, d_\omega) \leftarrow \text{diff}(\boldsymbol{\omega})$

$A_\theta : (d_\omega,\, d_h,\, d_h) \leftarrow \textit{Parameter}$ ▷ Exploit structure of A_θ

$\mathbf{I} : (d_h,\, d_h) \leftarrow d_h \times d_h$ identity matrix.

if $\textit{mode} = \text{parallel}$ **then**

  $\mathbf{F} : (B,\, L-1,\, d_h,\, d_h) \leftarrow \mathbf{I} + \text{einsum}(bli, ijk \to bljk, \boldsymbol{\omega}^{\text{inc}}, A_\theta)$ ▷ Broadcast $\mathbf{I}$, exploit structure of A_θ

  $\mathbf{F}^{\text{comp}} \leftarrow \text{pscan}(\mathbf{F})$ ▷ Parallel associative scan, exploit structure of $\mathbf{F}$

  $\mathbf{h}_{1:L-1} \leftarrow \text{einsum}(bljk, bk \to blj,\, \mathbf{F}^{\text{comp}},\, \mathbf{h}_0)$

  $\mathbf{h} \leftarrow \left[\mathbf{h}_0, \mathbf{h}_{1:L-1}\right]$

else ▷ Recurrent pass

  for $t \leftarrow 0$ **to** $L-2$ **do**

    $\mathbf{F}_t : (B,\, d_h,\, d_h) \leftarrow \mathbf{I} + \text{einsum}(bi, ijk \to bjk, \boldsymbol{\omega}_t^{\text{inc}}, A_\theta)$ ▷ Exploit structure of A_θ

    $\mathbf{h}_{t+1} \leftarrow \text{einsum}(bjk, bk \to bj,\, \mathbf{F}_t,\, \mathbf{h}_t)$ ▷ Exploit structure of $\mathbf{F}$

  end for

  $\mathbf{h} \leftarrow \left[\mathbf{h}_0, \ldots, \mathbf{h}_{L-1}\right]$ ▷ Stack along length axis

end if

return h

5.4.6 Experiments

First, all the SLiCE variants and a wide variety of sequence model baselines are evaluated on the A_5 benchmark used in Section 5.3.4. Our experiments follow the approach of Merrill et al. [2024]. The models are trained on sequences ranging from length 3 to 20 and compared on the number of stacked layers required to achieve greater than 90% validation accuracy. Models are trained using a token-tagging loss for 1,000,000 steps with a batch size of 256. For all sequence lengths, a small batch of sequences of length 2 are included at each training step to aid convergence. All models use Adam with weight decay as the optimiser, and linear warm-up followed by cosine annealing with a minimum learning rate of 10^{-5} and a maximum learning rate of 10^{-3} [Kingma et al. 2015]. Additionally, all models use dropout with a rate of 0.1 and a trainable embedding layer [Srivastava et al. 2014].

The five baseline models, Mamba, LSTM, mLSTM, sLSTM, and DeltaProduct, all use a hidden dimension of 1024. LSTM uses direct stacking whereas the other baseline models use stacked blocks consisting of a sequence model, a GLU layer [Dauphin et al. 2017], and layer normalisation [Ba et al. 2016]. DeltaProduct uses both gating and negative eigenvalues, maximising the potential expressivity [Yang et al. 2025; Grazzi et al. 2025]. All of the SLiCEs use

$$\frac{\omega^X_{t_{j+1}} - \omega^X_{t_j}}{t_{j+1} - t_j} = (1, X_{t_j}) \tag{5.107}$$

and 1024 non-zero parameters for each A^i_θ. For the diagonal and Walsh–Hadamard SLiCE this corresponds to a hidden dimension of 1024 and for the dense SLiCE this corresponds to a hidden dimension of 32. The DPLR SLiCE uses a rank of 2, giving a hidden dimension of 205, the block-diagonal SLiCE uses $b_i = 4$, giving a hidden dimension of 256, and the diagonal-dense SLiCE uses a dense block size of 23, giving a hidden dimension of 518. The sparse SLiCE uses a hidden dimension of 128 and a sparsity of $\epsilon = \frac{3}{7}$. All SLiCEs use stacked blocks consisting of a SLiCE, a linear layer followed by a tanh activation function, layer normalisation [Ba et al. 2016], and weight regularisation.

The second experiment evaluates length generalisation on sequences from the A_5 benchmark. Models which achieve at least 90% validation accuracy on sequences of length 20 in the previous experiment are retrained on sequences from length 3 to 40 with early stopping on a validation set of sequences from length 40 to 128. The

models are then evaluated on test sequences from length 20 to 5120. The mLSTM is excluded as it operates on fixed-length inputs.

The third experiment consists of the four regular tasks from the formal language benchmark, a collection of language style tasks split into categories using the Chomsky Hierarchy [Delétang et al. 2023]. They are:

1. Cycle navigation: Infer the final position of a walk on a cycle starting at the origin. Actions are randomly sampled from "go forward one step", "stay in the same place", and "go backward one step". We use a cycle of length 5. Therefore, a random guesser will achieve an accuracy of 20%.

2. Even pairs: There are two states in the system and the goal is to determine if there is an equal number of transitions between the two states. A random guesser will achieve an accuracy of 50%.

3. Modular arithmetic no brackets: Performs modular arithmetic consisting of only addition and multiplication. We use mod 5 and hence a random guesser will achieve 20%.

4. Parity: There are two elements in the system. To determine the parity, the number of the second element is counted to determine if it is even or odd. This can be viewed as modular summation with mod 2. A random guesser will achieve an accuracy of 50%.

All four of these tasks can be solved by processing inputs sequentially with a fixed set of internal states and no external memory, i.e. state-tracking. The models are challenged to generalise to longer sequences, by training on sequences from length 3 to 40 and evaluating on sequences from length 40 to 256. Following Beck et al. [2024], all models use two stacked layers and a trainable embedding layer. In addition to the hidden dimension of 512 used by Beck et al. [2024], we also train the baseline models with a hidden dimension of 128, selecting the value that yields the highest average validation accuracy for each model on each task. For Mamba, which does not support a hidden dimension of 128, we instead choose between 256 and 512 based on validation performance. All models are trained using a token-tagging loss for 100,000 steps with a batch size of 256. Additionally, all models use dropout with a rate of 0.01 [Srivastava et al. 2014], Adam with weight decay as the optimiser [Kingma et al. 2015], and a linear warm-up followed by cosine annealing with a minimum learning

rate of 10^{-5} and a maximum learning rate of 2×10^{-3}.

Similarly to the A_5 benchmark, LSTM uses direct stacking whereas the other baseline models use stacked blocks consisting of a sequence model, a GLU layer [Dauphin et al. 2017], and layer normalisation [Ba et al. 2016]. The baseline models considered are vanilla DeltaNet [Schlag et al. 2021; Yang et al. 2024a], DeltaNet with negative eigenvalues (DeltaNet[-1,1]) [Grazzi et al. 2025], Gated DeltaNet [Yang et al. 2025], Gated DeltaProduct with negative eigenvalues and a rank of 2 [Siems et al. 2025], sLSTM [Beck et al. 2024], mLSTM [Beck et al. 2024], xLSTM [Beck et al. 2024], RWKV-7 [Peng et al. 2025], a Transformer [Vaswani et al. 2017], S4D [Gu et al. 2022b], and Mamba [Gu et al. 2024].

We consider all SLiCEs on this benchmark except sparse due to the lack of an efficient implementation. All SLiCEs use

$$\frac{\omega^X_{t_{j+1}} - \omega^X_{t_j}}{t_{j+1} - t_j} = (1, X_{t_j}), \tag{5.108}$$

and two stacked blocks consisting of the sequence layer, a linear layer followed by a tanh activation function, and layer normalisation [Ba et al. 2016]. For the diagonal and Walsh–Hadamard SLiCE variants, we consider hidden dimensions of 128 and 512, corresponding to 128 and 512 non-zero parameters per state-transition matrix, respectively. For all other SLiCE variants, the number of non-zero parameters in the state-transition matrix is fixed at 512. For DPLR we consider ranks of $r = 1, 2, 4, 8$, and for block-diagonal we consider two variants, $b_i = b$ for all i with $b = 2, 4, 8, 16$, and $b_i = 1$ for $i = 1, \dots, k-1$, and then a final dense block $b_k = b$ for $b = 2, 4, 8, 16$, referred to as diagonal-dense SLiCE.

The final experiment expands the comparison of Section 5.2.5 to consider all SLiCE structures and all six datasets from the UEA-MTSCA used in Section 4.4.6. In particular, the vector field of a Log-NCDE is replaced by a structured A^i_θ, with all other hyperparameters kept identical to those in Table 4.3. The Log-SLiCEs are then retrained on each of the six UEA-MTSCA datasets from 4.4.6 and compared on average test set accuracy, time per training step, and GPU memory. All timing and GPU memory results were performed on an NVIDIA H100 GPU.

5.4.7 Results

Figure 5.4 presents the results on the A_5 benchmark. As expected from Theorem 5.9, the DPLR, sparse (S), Walsh–Hadamard (WH), and block-diagonal (BD) structures allow a SLiCE to achieve greater than 90% validation accuracy on all sequence lengths considered with only one layer, matching the performance of the dense Linear NCDE (DE-LNCDE). The only other models to achieve this are the non-linear recurrent models, sLSTM and LSTM. All other parallelisable models require a growing number of layers. This includes the diagonal-dense (D-DE) SLiCE and Gated DeltaProduct with negative eigenvalues, despite their structured state-transition matrices. However, DPLR and BD SLiCE both need one layer for all sequence lengths, suggesting this is not an inherent limitation of these structures.

Figure 5.5 presents the results on the A_5 length generalisation experiment. The non-linear recurrent LSTM and sLSTM generalise well, maintaining high test accuracy beyond both the training and validation ranges. Among the parallel-in-time models, three patterns emerge: (i) WH-SLiCE and Mamba do not attain high accuracy even at training lengths; (ii) DeltaProduct and D-DE-SLiCE generalise to approximately twice the training length but not beyond the validation range; and (iii) DE-LNCDE, DPLR-SLiCE, S-SLiCE, and BD-SLiCE sustain high accuracy on sequences at least 8 times the training length, exceeding the maximum validation length.

The results presented in Table 5.3 show that expressive structures like BD, D-DE, and DPLR significantly outperform the diagonal baseline on the regular language tasks. The top-performing D-DE and DPLR model, both with an average accuracy of 85.1%, demonstrate a substantial improvement over the best diagonal model's 70.2%. Conversely, the WH structure performs worse than the diagonal model, despite its greater theoretical expressivity. This provides an empirical demonstration that, although maximal probabilistic expressivity is a desirable property, it is an asymptotic notion and does not guarantee favourable optimisation, sample efficiency, or inductive bias at finite width. We hypothesise that the Walsh–Hadamard structure's fixed global mixing is poorly matched to the algorithmic state-tracking required by regular language tasks, as also suggested by its poor performance on the A_5 length generalisation task.

An analysis of the successful hyperparameter choices reveals a fundamental trade-

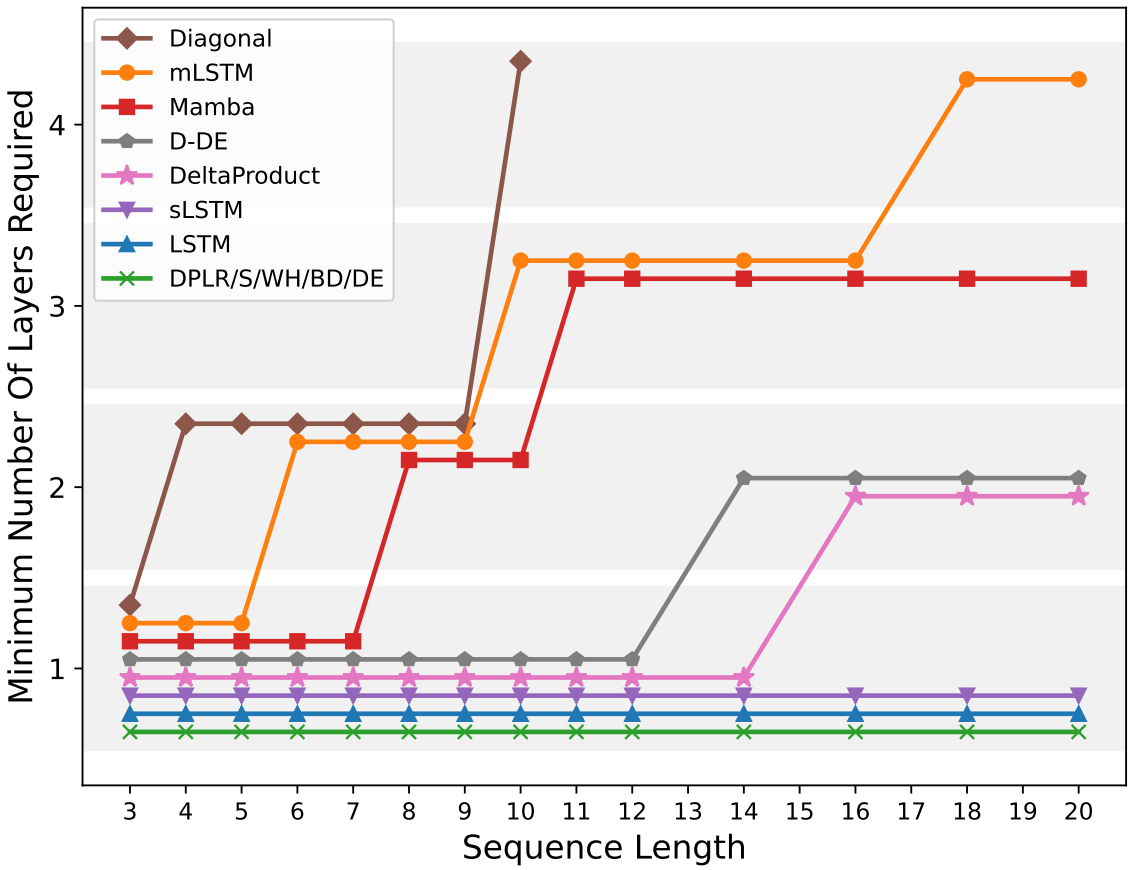


Figure 5.4: Comparison of an mLSTM, Mamba, DeltaProduct, sLSTM, LSTM, nd SLiCEs with diagonal, dense (DE), sparse (S), Walsh–Hadamard (WH), block-diagonal (BD), and diagonal-dense (D-DE) matrices on the A_5 benchmark. For each sequence length, the plot shows the minimum number of blocks required to achieve at least 90% validation accuracy, with each shaded band corresponding to a number of blocks. Missing points mean the model did not achieve at least 90% validation accuracy with 4 blocks or less.

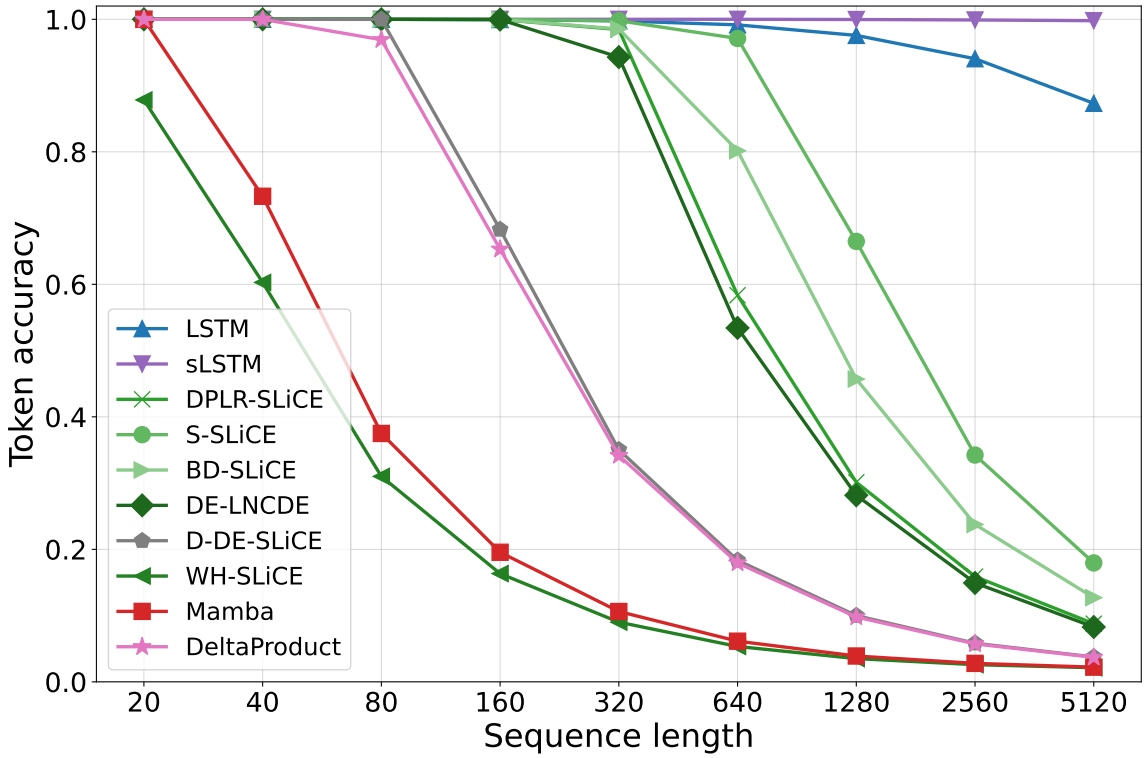


Figure 5.5: Comparison of an Mamba, DeltaProduct, sLSTM, LSTM, and SLiCEs with diagonal, dense (DE), sparse (S), Walsh–Hadamard (WH), block-diagonal (BD), and diagonal-dense (D-DE) matrices on A_5 length generalisation. Models are trained on sequences from length 3 to 40 with early stopping on a validation set of sequences from length 40 to 128. The plot shows test set token accuracy against sequence length.

off between model expressivity and hidden state dimension. For a fixed parameter budget, increasing one of these factors necessitates a decrease in the other. For instance, BD-SLiCE achieves its best performance not at an extreme, but with a block size of $b = 4$ and hidden dimension $d_h = 128$, yielding an average accuracy of 83.8%. Similarly, DPLR-SLiCE achieves its best performance with a rank of 4, yielding the joint highest average accuracy of 85.1%. This suggests that the strongest results emerge from maintaining a sufficient degree of structure without overly compressing the hidden state. The D-DE structure operates in a notably different regime, with even its largest dense block configuration retaining a larger hidden dimension than any BD or DPLR variant. As a result, D-DE-SLiCE achieves its best performance at the largest block size, matching the top average accuracy of DPLR-SLiCE.

Table 5.4 compares SLiCEs to a range of state-of-the-art sequence model baselines. As

Table 5.3: Average and standard deviation of validation accuracy over five runs on the regular language tasks for SLiCEs with diagonal, Walsh–Hadamard (WH), block-diagonal (BD), diagonal-dense (D-DE), and DPLR structures.

Model	**Cycle Nav.**	**Even Pairs**	**Mod Arith. No Brack.**	**Parity**	**Average**
$\text{Diagonal}_{d_h=128}$	69.5 ± 6.3	100.0 ± 0.0	20.8 ± 0.2	89.12 ± 18.6	69.9
$\text{Diagonal}_{d_h=512}$	59.7 ± 5.1	100.0 ± 0.0	20.9 ± 0.1	100.0 ± 0.0	70.2
$\text{WH}_{d_h=128}$	69.7 ± 8.8	93.1 ± 13.9	20.5 ± 0.3	50.7 ± 0.3	58.5
$\text{WH}_{d_h=512}$	35.5 ± 1.8	58.5 ± 2.5	23.8 ± 1.1	71.4 ± 12.9	47.3
$\text{BD}_{d_h=256,\, b=2}$	92.5 ± 14.0	72.7 ± 3.0	37.7 ± 2.6	99.6 ± 0.8	75.6
$\text{BD}_{d_h=128,\, b=4}$	99.8 ± 0.2	85.9 ± 11.3	54.0 ± 12.5	95.3 ± 3.9	83.8
$\text{BD}_{d_h=64,\, b=8}$	99.9 ± 0.1	91.3 ± 6.3	70.6 ± 21.4	54.1 ± 4.9	79.0
$\text{BD}_{d_h=32,\, b=16}$	97.6 ± 3.5	94.7 ± 6.5	76.6 ± 22.1	50.8 ± 0.3	79.9
$\text{D–DE}_{d_h=510,\, b=2}$	61.9 ± 20.4	91.3 ± 11.5	20.8 ± 0.2	97.8 ± 2.9	67.9
$\text{D–DE}_{d_h=500,\, b=4}$	81.6 ± 15.0	85.3 ± 18.0	29.4 ± 15.4	83.7 ± 9.4	70.0
$\text{D–DE}_{d_h=456,\, b=8}$	90.6 ± 9.4	90.7 ± 3.0	31.0 ± 4.2	79.9 ± 3.5	73.1
$\text{D–DE}_{d_h=272,\, b=16}$	73.3 ± 29.4	84.8 ± 8.5	98.4 ± 0.7	83.8 ± 11.3	85.1
$\text{DPLR}_{d_h=171,r=1}$	46.5 ± 26.3	91.1 ± 4.4	25.8 ± 10.2	87.8 ± 9.5	62.8
$\text{DPLR}_{d_h=102,r=2}$	53.1 ± 14.7	96.8 ± 5.1	43.9 ± 9.0	79.7 ± 14.4	68.4
$\text{DPLR}_{d_h=57,r=4}$	81.1 ± 16.6	100.0 ± 0.0	68.3 ± 19.3	91.0 ± 18.0	85.1
$\text{DPLR}_{d_h=30,r=8}$	90.1 ± 10.2	100.0 ± 0.0	60.3 ± 19.7	50.7 ± 0.3	75.3
Random	20.0	50.0	20.0	50.0	35.0

expected, the recurrent LSTM generalises near-perfectly on all four tasks, establishing a clear performance benchmark. This highlights the inherent strength of non-linear recurrent architectures for problems that require precise, step-by-step state tracking. Interestingly, sLSTM and xLSTM do not replicate this success. Among the parallel models, the strongest baselines are DeltaNet with negative eigenvalues and Gated DeltaProduct with negative eigenvalues, achieving 77.7% and 80.7%, respectively. This aligns with the expectation that increased complexity in the state-transition matrix improves state-tracking performance.

This principle is further reinforced by the observation that a diagonal SLiCE outperforms Mamba. Mamba is restricted to a state-transition matrix with eigenvalues in the range $[0, 1]$, whereas diagonal SLiCE is unrestricted. As shown by Grazzi et al. [2025], expanding the eigenvalue range of the state-transition matrix to $[-1, 1]$ is crucial for solving regular language tasks. On average validation accuracy, the diagonal-dense and DPLR SLiCE are the two strongest performing parallelisable models. They outperform not only widespread modern architectures like Mamba and the Transformer, but also the strongest DeltaNet baselines specifically designed for increased expressivity.

A Friedman test detected a statistically significant difference in performance among the parallel-in-time models across the four tasks at the 5% significance level, with $\chi^2 = 25.41$, $df = 13$, and $p = 0.0204$. To assess our architectural hypothesis more directly, we partitioned the parallel models into those with expressive recurrences and those without. The expressive class consists of DPLR-SLiCE, D-DE-SLiCE, BD-SLiCE, WH-SLiCE, DeltaNet$[-1, 1]$, Gated DeltaProduct$[-1, 1]$, DeltaNet, and Gated DeltaNet, while the non-expressive class consists of RWKV-7, mLSTM, Transformer, Mamba, S4D, and D-SLiCE. We then compared the mean rank of each class on each task, with lower rank indicating better performance. The expressive class attained mean ranks of 5.62, 6.75, 5.25, and 6.12 on Cycle Navigation, Even Pairs, Modular Arithmetic without Brackets, and Parity, respectively, compared to 10.00, 8.50, 10.50, and 9.33 for the non-expressive class. Overall, this corresponds to average ranks of 5.94 for the expressive class and 9.58 for the non-expressive class. We do not extend this descriptive summary to a statistical one, since four paired datasets are not sufficient for a Wilcoxon signed-rank test to assign significance [Wilcoxon 1945].

Table 5.5 and 5.6 present the impact of replacing the Log-NCDE's non-linear vector

Table 5.4: Average and standard deviation of validation accuracy over five runs for a range of recurrent and parallel models on the formal language tasks.

Model	**Cycle Nav.**	**Even Pairs**	**Mod Arith. No Brack.**	**Parity**	**Average**
Recurrent					
LSTM	100.0 ± 0.0	100.0 ± 0.0	99.9 ± 0.1	100.0 ± 0.0	100
sLSTM	32.5 ± 0.4	100.0 ± 0.0	27.7 ± 0.6	100.0 ± 0.0	65.1
xLSTM[1:1]	53.5 ± 5.6	99.0 ± 1.9	29.3 ± 1.6	100.0 ± 0.0	70.5
Parallel					
DeltaNet	49.8 ± 4.7	100.0 ± 0.0	42.2 ± 4.8	57.8 ± 0.8	62.5
DeltaNet$[-1, 1]$	46.7 ± 6.1	100.0 ± 0.0	66.4 ± 8.8	97.7 ± 2.0	77.7
Gated DeltaNet	53.8 ± 8.8	100.0 ± 0.0	42.8 ± 8.2	56.5 ± 1.9	63.3
Gated DeltaProduct[-1,1]	46.3 ± 6.6	100.0 ± 0.0	78.4 ± 10.9	98.0 ± 1.4	80.7
RWKV-7	37.8 ± 5.0	88.1 ± 14.2	39.5 ± 6.1	51.1 ± 0.3	54.1
mLSTM	52.4 ± 10.5	99.9 ± 0.1	28.8 ± 3.1	53.0 ± 2.1	58.5
Transformer	24.4 ± 0.5	90.4 ± 10.4	23.6 ± 0.7	52.2 ± 0.4	47.7
Mamba	48.4 ± 2.2	100.0 ± 0.0	33.1 ± 6.6	54.2 ± 2.1	58.9
S4D	23.7 ± 1.1	68.7 ± 4.7	21.7 ± 0.4	51.2 ± 1.0	41.3
Diagonal$_{d_h=512}$	59.7 ± 5.1	100.0 ± 0.0	20.9 ± 0.1	100.0 ± 0.0	70.2
WH$_{d_h=128}$	69.7 ± 8.8	93.1 ± 13.9	20.5 ± 0.3	50.7 ± 0.3	58.5
BD$_{d_h=128,\ b=4}$	99.8 ± 0.2	85.9 ± 11.3	54.0 ± 12.5	95.3 ± 3.9	83.8
D–DE$_{d_h=272\ b=16}$	73.3 ± 29.4	84.8 ± 8.5	98.4 ± 0.7	83.8 ± 11.3	85.1
DPLR$_{d_h=57,r=4}$	81.1 ± 16.6	100.0 ± 0.0	68.3 ± 19.3	91.0 ± 18.0	85.1
Random	20.0	50.0	20.0	50.0	35.0

Table 5.5: Mean and standard deviation of test set accuracy over five data splits on a subset of the UEA-MTSCA. The best-performing model in each column is highlighted in bold, and the second-best is underlined. Models are sorted by average rank, with the top performing model first.

Model	EW	EC	HB	MI	SCP1	SCP2
BD-SLiCE	86.1 ± 3.6	28.6 ± 6.4	77.4 ± 5.6	53.0 ± 2.1	$\underline{84.9 \pm 1.9}$	$\mathbf{54.0 \pm 7.4}$
Log-NCDE	85.6 ± 5.1	$\mathbf{34.4 \pm 6.4}$	75.2 ± 4.7	53.7 ± 5.3	83.1 ± 2.9	53.7 ± 4.1
D-DE-SLiCE	85.6 ± 6.0	27.3 ± 6.9	73.9 ± 3.8	$\mathbf{54.7 \pm 3.5}$	84.7 ± 3.7	51.6 ± 5.0
WH-SLiCE	85.0 ± 6.0	30.1 ± 4.8	76.1 ± 5.9	49.5 ± 6.8	82.4 ± 2.2	51.9 ± 5.4
DPLR-SLiCE	84.4 ± 5.2	27.6 ± 4.7	74.2 ± 5.4	51.6 ± 5.8	83.5 ± 2.2	50.9 ± 5.8
D-SLiCE	79.4 ± 5.8	27.1 ± 4.6	72.9 ± 5.1	$\underline{54.4 \pm 6.3}$	83.5 ± 2.2	53.0 ± 5.9
LRU	$\underline{87.8 \pm 2.9}$	21.5 ± 2.2	$\mathbf{78.4 \pm 6.7}$	48.4 ± 5.1	82.6 ± 3.5	51.2 ± 3.6
S6	85.0 ± 16.1	26.4 ± 6.5	76.5 ± 8.3	51.3 ± 4.8	82.8 ± 2.8	49.9 ± 9.5
S5	81.1 ± 3.7	24.1 ± 4.4	$\underline{77.7 \pm 5.6}$	47.7 ± 5.5	$\mathbf{89.9 \pm 4.7}$	50.5 ± 2.6
S-SLiCE	$\underline{87.8 \pm 4.2}$	$\underline{30.4 \pm 6.7}$	72.6 ± 5.5	47.7 ± 2.4	82.8 ± 1.6	49.5 ± 4.0
DE-LNCDE	$\mathbf{88.3 \pm 3.7}$	25.8 ± 4.1	74.2 ± 4.5	49.8 ± 5.3	81.9 ± 4.0	49.8 ± 2.4
NCDE	75.0 ± 4.0	29.9 ± 6.6	73.9 ± 2.6	49.5 ± 2.9	79.8 ± 5.7	53.0 ± 2.9
NRDE	83.9 ± 7.3	25.3 ± 1.8	72.9 ± 4.9	47.0 ± 5.8	80.9 ± 2.6	$\underline{53.7 \pm 6.9}$
Mamba	70.9 ± 15.9	27.9 ± 4.6	76.2 ± 3.9	47.7 ± 4.5	80.7 ± 1.4	48.2 ± 4.0

field with a dense, BD, WH, diagonal, D-DE, sparse, or DPLR structured linear vector field. The original baselines from Table 4.4 are also included. A tie-corrected Friedman test across the 14 models and 6 datasets did not detect a statistically significant difference in performance at the 5% significance level, with $\chi^2 = 19.89$, $df = 13$, and $p = 0.098$. Nevertheless, the results show that the block-diagonal structure achieves very similar performance to the non-linear model, whilst reducing the average time per training step by a factor of nearly 20. Notably, BD-SLiCE also outperforms a dense linear vector field of the same hidden dimension, which suggests that the block-diagonal structure may provide benefits beyond simply reducing computational cost. A possible explanation for this is the structure's conceptual link to the multi-head attention mechanism found in Transformers. Each block in BD-SLiCE can be viewed as an independent head, processing the input path in its own distinct subspace. This architectural choice encourages the model to learn multiple specialised representations of the time-series dynamics in parallel, with each block potentially focusing on different features or patterns. This inherent modularity could prevent the overfitting that may affect a single dense model, ultimately contributing to the strong performance of BD-SLiCE.

Table 5.6: Average test accuracy, rank, training time per 1,000 steps, and GPU memory usage across six datasets from the UEA-MTSCA. All SLiCE variants use a parallel associative scan during training. Therefore, the Walsh–Hadamard, DPLR, and sparse SLiCEs are treated as dense LNCDEs (see Section 5.4.4), and their timing and GPU memory results are omitted. All timing and GPU memory results were performed on an NVIDIA H100 GPU.

Model	**Av. Acc**	**Av. Rank**	**Av. Time / 1000 Steps (s)**	**Av. GPU mem (MB)**
BD-SLiCE	64.0	3.2	68.1	2344
Log-NCDE	64.3	4.0	1321.7	2177
D-DE-SLiCE	63.0	5.7	66.7	2302
WH-SLiCE	62.5	6.7	–	–
DPLR-SLiCE	62.0	7.0	–	–
D-SLiCE	61.7	7.2	11.0	1875
LRU	61.7	7.3	26.9	4308
S6	62.0	7.7	20.1	2938
S5	61.8	8.0	21.9	3327
S-SLiCE	61.8	8.2	–	–
DE-LNCDE	61.6	8.3	77.2	12756
NCDE	60.2	8.8	6923	1962
NRDE	60.6	10.3	3431	2858
Mamba	58.6	10.8	60.0	4535

Figure 5.6 provides a visual summary of the accuracy–speed trade-off on the UEA-MTSCA benchmark, with the Pareto frontier highlighting the non-dominated models. The frontier consists of diagonal SLiCE, S6, block-diagonal SLiCE, and Log-NCDE, each representing a different balance between training time and predictive accuracy. Log-NCDE attains the strongest test accuracy, but at a substantially higher average time per training step. Moving along the frontier, block-diagonal SLiCE reduces the average time per training step by a factor of 20 relative to Log-NCDE while maintaining very similar test accuracy. Diagonal SLiCE pushes training time lower still, below that of the SSM baselines, though with some further loss in accuracy consistent with its reduced expressivity. S6 occupies an intermediate point on the frontier, offering a modest improvement in accuracy over diagonal SLiCE, although the gain is smaller than that obtained by moving from S6 to block-diagonal SLiCE. By contrast, dense LNCDE slightly increases run-time and substantially raises GPU memory usage relative to the block-diagonal variant, while also reducing test accuracy, which may indicate over-fitting due to the large number of parameters per state transition.

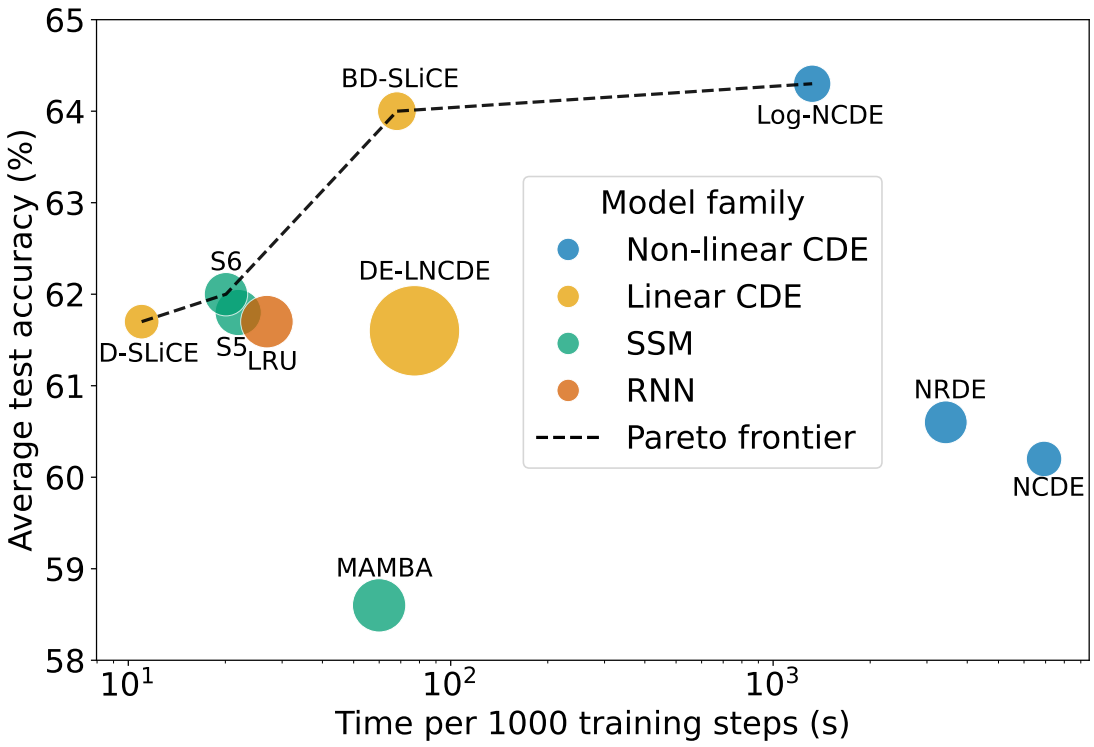


Figure 5.6: Average per-step training time versus average validation accuracy across six multivariate time series classification datasets from the UEA-MTSCA. Each point represents a model, with circle area proportional to average GPU memory usage. We compare four families of models: a recurrent neural network (LRU), SSMs (S5, S6, and Mamba), non-linear NCDEs (NCDE, NRDE, and Log-NCDE), and Linear NCDEs (Diagonal SLiCE, Block-Diagonal SLiCE, and Dense LNCDE). All timing and GPU memory results were performed on an NVIDIA H100 GPU.

Finally, Figure 5.6 shows that Mamba, NCDE, and NRDE do not offer competitive trade-offs on either axis on this benchmark.

To understand their relative impact, we evaluate how the Log-ODE method and parallel associative scan influence time per training step and GPU memory for the diagonal SLiCE, block-diagonal SLiCE, and dense LNCDE. The EigenWorms dataset is chosen for this comparison, as it contains approximately 18,000 observations per time series. Table 5.7 summarises the effect of applying a parallel associative scan with varying chunk sizes on time per 1,000 training steps without the Log-ODE method. For all three models, increasing the number of parallel steps yields strong reductions in time per training step. The impact of the high I/O costs associated with an as-

Table 5.7: Time per 1,000 training steps (s) for diagonal SLiCE, block-diagonal SLiCE, and dense LNCDE on EigenWorms when a parallel associative scan is applied with various chunk sizes. Experiments were performed on an NVIDIA H100, and the batch size is 1.

Parallel Steps	D-SLiCE	BD-SLiCE	DE-LNCDE
None	311.0	374.89	444.68
4	134.2	326.75	439.51
16	57.50	161.74	257.56
64	31.01	68.59	126.01
256	21.10	30.53	71.54

sociative scan is evident from the diminishing benefit of a small number of parallel steps as you move from diagonal, through block-diagonal, to dense matrices.

Table 5.8 compares the time per 1,000 training steps and GPU memory for diagonal SLiCE, block-diagonal SLiCE, and dense LNCDE on EigenWorms when using a parallel associative scan and the Log-ODE method. As expected, both GPU memory and run-time increase monotonically for every combination of associative scan and Log-ODE method as the model structure transitions from diagonal, through block-diagonal, to dense matrices. The consistent 2.69 GB floor across several configurations likely reflects peak memory usage from fixed operations outside the recurrence. Without the Log-ODE method, the dense LNCDE exhibits high GPU memory consumption, which constrained experiments to a batch size of 4. When both the Log-ODE method and a parallel associative scan are applied, the diagonal and block-diagonal SLiCE achieve comparable times per training step, indicating that the recurrence contributes less to overall computation under these parameter settings. Overall, combining the Log-ODE method with a parallel associative scan reduces the time per training step by a factor of approximately 30 for the diagonal and block-diagonal SLiCE without affecting GPU memory, and by a factor of 16 for the dense LNCDE while lowering GPU memory usage by over a factor of 5.

5.4.8 Limitations

First, we do not impose stability constraints on the matrices A_θ^i in the models considered here. Consequently, although we did not observe instability on the long time

Table 5.8: Comparison of training time and GPU memory for EigenWorms using a parallel associative scan and applying the Log-ODE method. Experiments were performed on an NVIDIA H100 with a batch size of 4.

Model	Metric	Log-ODE Interval	Parallel Steps	
			None	128
D-SLiCE	Time / 1k steps [s]	1	317.49	23.51
		12	33.03	10.96
	GPU Memory [GB]	1	2.69	2.69
		12	2.69	2.69
BD-SLiCE	Time / 1k steps [s]	1	378.20	48.74
		12	46.98	13.24
	GPU Memory [GB]	1	2.69	4.73
		12	2.69	2.69
DE-LNCDE	Time / 1k steps [s]	1	465.94	167.86
		12	47.95	29.37
	GPU Memory [GB]	1	35.45	51.84
		12	2.69	6.79

series considered in this chapter, one should not expect the current parameterisations to remain uniformly stable as the sequence length grows arbitrarily large. Therefore, investigating stable matrix structures for the A_θ^i is a natural direction for extending SLiCEs to more extreme long-range settings.

Second, SLiCEs are sequence-to-sequence models. They produce a state update for each new sample in the input, and are therefore susceptible to over-sampled data in the same manner as other discrete sequence models. Combining SLiCEs with the Log-ODE method allows them to consume paths as input, by abstracting the flow on each interval away from the individual samples. However, the output of a Log-SLiCE is itself a sequence of samples, corresponding to the value of the output path at the end of each interval to which the Log-ODE method is applied. This means that one cannot stack two Log-SLiCEs directly, as the first level produces a sequence whereas the second consumes a path. A natural next step is therefore to develop a path-to-path model, as discussed in more detail in Section 6.2.

5.5 Conclusion

This chapter introduced Linear NCDEs, where the non-linear vector field $f_\theta(h_s)$ is replaced by a linear one, $A_\theta h_s$. Linear NCDEs retain the theoretical expressivity of their non-linear counterparts while enabling parallel-in-time computation. The core innovation is removing the recurrent differential equation solve and instead solving for the flow independently on each interval. These flows are then composed using a parallel associative scan. This yields substantial performance gains, particularly for long sequences. On the EigenWorms dataset, which has nearly 18,000 observations per time series, this method reduces the time per training step by a factor of over 250. When combined with the Log-ODE method from Chapter 4, this performance gain increases to a factor of nearly 1500, reducing the time for 10,000 training steps from 3 days to 3 minutes.

Section 5.3 used the Linear NCDE framework to theoretically analyse the expressivity of SSMs. This analysis demonstrated that prominent models like S4D and Mamba correspond to affine Linear NCDEs whose state-transition matrices are constrained to be diagonal. The diagonal structure was shown to severely limit theoretical expressivity. This limitation was confirmed empirically on state-tracking benchmarks, where diagonal models required stacking multiple layers to solve tasks that dense Linear NCDEs could solve with one.

While theoretically powerful, the number of parameters and computational cost of dense state-transition matrices scales with the cube of the hidden dimension, making them impractical for large models. To address this, Section 5.4 introduced SLiCEs, which bridge the efficiency-expressivity gap. By employing structured matrices, such as block-diagonal, diagonal-plus-low-rank, and Walsh-Hadamard, SLiCEs reduce computational complexity while retaining the maximal probabilistic expressivity of their dense counterparts. Extensive experiments validated the SLiCE approach. On formal language benchmarks that test state-tracking, SLiCEs outperformed specialised SSM baselines. In particular, a block-diagonal and DPLR structure established the new state-of-the-art for parallel models on these tasks. This success extends to the real-world UEA datasets, where replacing the non-linear vector field of a Log-NCDE with a block-diagonal linear vector field does not decrease performance, while bringing the training time in line with other modern sequence models.

SLiCEs provide a strong foundation for future work in several key directions. An immediate technical goal is the development of efficient GPU kernels for the matrix exponential to allow for exact flow computations. Further research can address open questions around path-to-path training, input symmetries, and the inclusion of bidirectional context in the vector fields, all of which are discussed in more detail in Section 6.2. Addressing these challenges will be crucial for the ultimate goal of scaling these expressive, parallelisable architectures to new and more complex domains.

Chapter 6

Conclusion

> *We can only see a short distance ahead, but we can see plenty there that needs to be done.*

—Alan Turing, *Computing Machinery and Intelligence* (1950)

6.1 Principal Contributions

Chapter 2 established a mathematical framework for understanding data that evolves continuously in time. Paths are the central objects, signatures provide principled summaries of path segments, CDEs describe how paths drive the evolution of hidden states, and the Log-ODE method provides an efficient way to approximate the resulting dynamics. NCDEs translate this framework into machine learning, but their practical scalability is limited by the need to solve a non-linear differential equation sequentially during training. The principal contributions of this thesis addressed this limitation in three complementary stages. First, Chapter 4 applied the Log-ODE method, an efficient and accurate approach to approximating the solution of a CDE, to NCDEs. This improved both their computational efficiency and empirical performance. Second, Section 5.2 introduced Linear NCDEs, which replaced the non-linear vector field of an NCDE with a linear one. This yielded explicit solutions for the flow, removing the need for a non-linear differential equation solver and enabling parallel-in-time computation. Third, Section 5.4 developed SLiCEs, which improved the balance between expressivity and computational efficiency through the use of structured matrices. Together, these contributions reduced the time per training step by up to three orders of magnitude while improving performance on real-world

benchmarks, making continuous-time models feasible at scales that were previously impractical.

Applying the Log-ODE method to NCDEs required showing that the neural network vector field satisfied the necessary regularity assumptions. Chapter 3 strengthened the regularity theory needed for this step by proving an explicit bound for the composition of $\mathrm{Lip}(\gamma)$ functions when $1 < \gamma \leq 2$. Chapter 4 then applied this theory to a class of fully connected neural networks parametrising NCDE vector fields, proving explicit $\mathrm{Lip}(\gamma)$ bounds for these architectures. This justified the use of the Log-ODE method during training. Empirically, Log-NCDEs improved upon standard NCDEs across real-world multivariate time series benchmarks, increasing predictive performance while reducing training cost. However, because the hidden state was still evolved by a non-linear differential equation, the forward pass remained inherently sequential.

Linear NCDEs addressed this bottleneck by replacing the non-linear vector field with one that is linear in the hidden state. This change yielded closed-form flows on each interval, removing the need for a recurrent differential equation solver and enabling parallel-in-time computation via an associative scan. Section 5.2 further showed that this linearisation preserves the theoretical expressivity of NCDEs while maintaining strong empirical performance on the benchmarks considered. Section 5.3.2 provided a unified interpretation of modern structured state-space models, such as S4D and Mamba, as affine Linear NCDEs with diagonal structure. Both the theory and the state-tracking experiments showed that, although efficient, diagonal structure discards hidden-state interactions that are important for expressivity.

SLiCEs refined this picture by improving the trade-off between expressivity and efficiency within the Linear NCDE framework. Rather than using dense or diagonal matrices, Section 5.4 introduced structured matrices, such as block-diagonal and diagonal-plus-low-rank, which reduce the computational cost while retaining maximal probabilistic expressivity. This showed that poor state-tracking is not an unavoidable consequence of efficient parallel training, but instead a consequence of overly restrictive recurrence structure. Empirically, SLiCEs outperformed specialised parallel baselines on formal language and state-tracking benchmarks. On real-world multivariate time series tasks, they also delivered a substantially stronger accuracy-speed trade-off than either non-linear continuous-time models or diagonal structured

state-space models.

Overall, this thesis showed that the main obstacle to scalable continuous-time learning was not the continuous-time viewpoint itself, but the combination of non-linear vector fields and inefficient numerical methods. By combining the Log-ODE method, closed-form linear flows, and expressive structured recurrences, it showed that continuous-time models can be made scalable without giving up the advantages of the continuous-time framework.

6.2 Future Work

There are a number of interesting directions for future work, from which we highlight five we view as particularly promising or important.

First, parallel associative scans provide significant benefit for training models on the million parameter scale. However, the high I/O costs prevent scaling to billions of parameters, as discussed in Section 5.4.4. We are currently exploring the possibility of a GPU kernel that will allow fast matrix exponentials and parallel associative scans when we have many small independent systems, i.e. a block-diagonal SLiCE. Another approach to overcoming this barrier is building on the work of Yang et al. 2024b, and developing fast chunk-wise methods for a wider range of structured matrices, as explored by Cirone et al. [2025b].

Second, there is an inherent over-parametrisation in Linear NCDEs. To see why, consider a trained Linear NCDE

$$\mathrm{d}h_s = \sum_{i=1}^{d_\omega} A_\theta^i h_s \mathrm{d}\omega_s^{X,i}, \tag{6.1}$$

with a linear readout $L_\theta h_s$. For any invertible matrix $P \in \mathbb{R}^{d_h \times d_h}$, we can apply a change of basis $h'_s = Ph_s$. The dynamics of the transformed state are then given by

$$\mathrm{d}h'_s = \sum_{i=1}^{d_\omega} PA_\theta^i P^{-1} h'_s \mathrm{d}\omega_s^{X,i} = \sum_{i=1}^{d_\omega} B_\theta^i h'_s \mathrm{d}\omega_s^{X,i}. \tag{6.2}$$

When equipped with the transformed linear readout $L_\theta P^{-1} h'_s$, the system described by (6.2) yields an identical output to (6.1), despite having different learnt parameters. This establishes an entire equivalence class of models for any single solution found

during training, with each member of the class corresponding to a different choice of basis for the hidden state space. Similarly to the identifiability issues arising in the classical system identification setting discussed in Section 4.2.1, the parameters of a Linear NCDE are therefore only identifiable up to similarity transformations.

This equivalence has significant potential implications for optimisation and interpretation. The existence of a continuous family of equivalent solutions creates manifolds in the loss landscape along which the gradient is zero, corresponding to moves in parameter space that amount only to a change of basis [Li et al. 2019]. In principle, this can lead to ill-conditioned optimisation and may hinder the convergence of gradient-based methods [Saarinen et al. 1993]. However, in the SLiCE experiments, we did not observe clear optimisation failures that could be specifically attributed to this symmetry. Even so, it implies that different training runs may converge to parametrisations that realise the same input-output map in different hidden bases, which makes the learned matrices difficult to compare or interpret directly. Therefore, a natural direction for future work is to optimise over similarity classes rather than individual parametrisations. Similar issues have motivated SGD variants for ReLU neural networks that operate on a quotient space of the weights [Meng et al. 2019]. Alternatively, one could reduce the redundancy by imposing additional structure on the hidden-state representation, for example by constraining one of the A_θ^i to be upper triangular.

Third, Cirone et al. [2024] established that replacing the Linear NCDE's linear readout with a non-linear function is sufficient to extend their path-to-point universality to path-to-path universality. However, the optimal approach to training a path-to-path Linear NCDE remains an open question. Standard objectives, such as pointwise mean squared error, treat paths as collections of independent samples. This neglects global temporal structure and leads to over-fitting local fluctuations, while failing to capture essential features such as ordering, variability, or long-range dependence [Ramsay et al. 2005; Cuturi et al. 2017; Chen et al. 2025]. Alternative choices include integral norms [Ferraty et al. 2007], distances in signature space [Lyons 1998; Chevyrev et al. 2016b; Király et al. 2019; Salvi et al. 2021; Cass et al. 2024a], or Kernel/MMD-based metrics tailored to streams [Cuturi et al. 2007; Mikalsen et al. 2018; Wynne et al. 2022]. These options can handle irregular sampling, are reparametrisation-invariant when desired, and are suitable for models whose outputs are paths.

Fourth, Log-NCDEs are path-to-sequence models, as they produce the value of the output path only at the endpoints of the intervals where the Log-ODE method has been applied. Linear NCDEs provide a tractable approach to path-to-path models, by extending their flow from describing the hidden state trajectory to the signature of those trajectories. In this way, it is possible to query the output path and obtain the signature over an interval in the same manner we would query the input path. This elevates continuous-time learning from path-to-sequence architectures to genuine path-to-path models. Furthermore, this would enable path-to-path stacking, where the output of one Log-Linear NCDE becomes the input to another.

Finally, as Linear CDEs allow us to contextualise the state-transition matrix based on the input, we could seek to contextualise the CDE based on the past, and potentially future, of the input path. A possible implementation would be allowing the Linear NCDE's vector field to depend linearly on the signature of the input path. However, this significantly increases the parameter count and computational complexity of the model. An alternative approach is conditioning the vector field to depend on the output of another Linear NCDE. This is an alternative form of model stacking, where instead of using the output of the Linear NCDE from one layer as the input to the next, we instead condition the vector field of the next layer based on the output of the previous.

Each of these directions offers an opportunity to advance both the theoretical underpinnings and the practical scalability of continuous-time models. Pursuing them will mature the path-based methodologies advocated for throughout this thesis, enhancing their power and broadening their applicability.

Bibliography

Albrecht, F., H. G. Diamond, and M. Heins (1971). "A Converse of Taylor's Theorem", *Indiana University Mathematics Journal* 21.4, pp. 347–350.

Apostol, T. M. (1974). *Mathematical Analysis*. 2nd. Addison-Wesley.

Arora, S., S. Eyuboglu, A. Timalsina, I. Johnson, M. Poli, J. Zou, A. Rudra, and C. Ré (2024). "Zoology: Measuring and improving recall in efficient language models", *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Åström, K. J. and P. Eykhoff (1971). "System Identification—A Survey", *Automatica* 7.2, pp. 123–162.

Åström, K. J. and T. Bohlin (1965). "Numerical Identification of Linear Dynamic Systems from Normal Operating Records", *Proc. IFAC Conference on Self-Adaptive Control Systems*. Vol. 2.

Atkinson, K. E., W. Han, and D. E. Stewart (2009). *Numerical Solution of Ordinary Differential Equations*. John Wiley & Sons.

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016). "Layer normalization", *arXiv preprint arXiv:1607.06450*.

Bagnall, A., H. A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh (2018). "The UEA multivariate time series classification archive, 2018", *arXiv preprint, arXiv:1811.00075*.

Bahdanau, D., K. Cho, and Y. Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate", *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Baldi, P., M. Berti, E. Haus, and R. Montalto (2018). "Time quasi-periodic gravity water waves in finite depth", *Inventiones mathematicae* 214.2, pp. 739–911.

Bayer, C., S. Breneis, and T. Lyons (2023). "An Adaptive Algorithm for Rough Differential Equations", *arXiv preprint arXiv:2307.12590*.

Beck, M., K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter (2024). "xLSTM: Extended Long Short-Term

Memory", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Behrouz, A., P. Zhong, and V. Mirrokni (2025). "Titans: Learning to Memorize at Test Time", *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*.

Berlinet, A. and C. Thomas-Agnan (2004). *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer.

Berndt, T., B. Walker, T. Qin, J. Stühmer, and A. Kormilitzin (2025). "Permutation Equivariant Neural Controlled Differential Equations for Dynamic Graph Representation Learning", *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*.

Blelloch, G. E. (1993). "Prefix Sums and Their Applications", *Synthesis of Parallel Algorithms*, pp. 35–60.

Boedihardjo, H., X. Geng, T. Lyons, and D. Yang (2016). "The signature of a rough path: Uniqueness", *Advances in Mathematics* 293, pp. 720–737.

Bonnier, P., P. Kidger, I. P. Arribas, C. Salvi, and T. Lyons (2019). "Deep Signature Transforms", *Neural Information Processing Systems (NeurIPS)*.

Boutaib, Y. (2016). "Lipschitz Geometry and Rough Paths". PhD thesis. University of Oxford.

Boutaib, Y., L. Gyurkó, T. Lyons, and D. Yang (2013). "Dimension-free Euler estimates of rough differential equations", *Revue Roumaine des Mathematiques Pures et Appliquees* 59.1, pp. 25–53.

Box, G. E. P. and G. M. Jenkins (1970). *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Bradbury, J., R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang (2018). *JAX: composable transformations of Python+NumPy programs*. URL: http://github.com/google/jax.

Brouwer, E. D., J. Simm, A. Arany, and Y. Moreau (2019). "GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series", *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*.

Brown, T. B., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. (2020). "Language models are few-shot learners", *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*.

Bryson, A. E. and W. F. Denham (1962). "A Steepest-Ascent Method for Solving Optimum Programming Problems", *Journal of Applied Mechanics* 29.2, pp. 247–257.

Cass, T., C. Litterer, and T. Lyons (2012). *New Trends in Stochastic Analysis and Related Topics: A Volume in Honour of Professor K. D. Elworthy*. Interdisciplinary mathematical sciences. World Scientific. Chap. 2.

Cass, T., T. Lyons, and X. Xu (2024a). "Weighted Signature Kernels", *Annals of Applied Probability* 34.1A, pp. 585–626.

Cass, T. and C. Salvi (2024b). "Lecture notes on rough paths and applications to machine learning", *arXiv preprint arXiv:2404.06583*.

Castell, F. and J. Gaines (1995). "An efficient approximation method for stochastic differential equations by means of the exponential Lie series", *Mathematics and Computers in Simulation* 38.1, pp. 13–19.

Chang, J., T. Lyons, and H. Ni (2018). "Super-multiplicativity and a lower bound for the decay of the signature of a path of finite length", *Comptes Rendus Mathématique* 356.7, pp. 720–724.

Chen, K. T. (1954). "Iterated Integrals and Exponential Homomorphisms", *Proceedings of the London Mathematical Society* s3-4.1, pp. 502–512.

Chen, K. T. (1957). "Integration of Paths, Geometric Invariants and a Generalized Baker-Hausdorff Formula", *Annals of Mathematics* 65.1.

Chen, R. T., Y. Rubanova, J. Bettencourt, and D. K. Duvenaud (2018). "Neural Ordinary Differential Equations", *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*.

Chen, Y., J. Zhang, X. Li, et al. (2025). "Patch-wise Structural Loss for Time Series Forecasting", *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Chevyrev, I. and A. Kormilitzin (2016a). "A Primer on the Signature Method in Machine Learning", *arXiv preprint arXiv:1603.03788*.

Chevyrev, I. and T. J. Lyons (2016b). "Characteristic functions of measures on geometric rough paths", *Annals of Probability* 44.6, pp. 4049–4082.

Cho, K., B. van Merriënboer, D. Bahdanau, and Y. Bengio (2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches", *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111.

Choi, J., H. Choi, J. Hwang, and N. Park (2022). "Graph Neural Controlled Differential Equations for Traffic Forecasting", *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI-22)*.

Choi, J. and N. Park (2023). "Graph Neural Rough Differential Equations for Traffic Forecasting", *ACM Transactions on Intelligent Systems and Technology* 14.4.

Cirone, N. M., M. Lemercier, and C. Salvi (2023). "Neural signature kernels as infinite-width-depth-limits of controlled ResNets", *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Cirone, N. M., J. Hamdan, and C. Salvi (2025a). "Genus expansion for non-linear random matrix ensembles with applications to neural networks", *arXiv preprint arXiv:2407.08459*.

Cirone, N. M., A. Orvieto, B. Walker, C. Salvi, and T. Lyons (2024). "Theoretical Foundations of Deep Selective State-Space Models", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Cirone, N. M. and C. Salvi (2025b). "ParallelFlow: Parallelizing Linear Transformers via Flow Discretization", *arXiv preprint arXiv:2504.00492*.

Cirone, N. M. and C. Salvi (2025c). "Rough kernel hedging", *arXiv preprint arXiv:2501.09683*.

Cochrane, T., P. Foster, V. Chhabra, M. Lemercier, T. Lyons, and C. Salvi (2021). "SK-Tree: a systematic malware detection algorithm on streaming trees via the signature kernel", *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, pp. 35–40.

Cohen, S. N., J. Foster, P. Foster, H. Lou, T. Lyons, S. Morley, J. Morrill, H. Ni, E. Palmer, B. Wang, Y. Wu, L. Yang, and W. Yang (2024). "Subtle variations in sepsis-III definitions markedly affect predictive performance", *Nature Scientific Reports* 14.1920.

Cohen, S. N., S. Lui, W. Malpass, G. Mantoan, L. Nesheim, Á. de Paula, A. Reeves, C. Scott, E. Small, and L. Yang (2023). "Nowcasting with signature methods", *arXiv preprint arXiv:2305.10256*.

Compagnoni, E., A. Scampicchio, L. Biggio, A. Orvieto, T. Hofmann, and J. Teichmann (2023). "On the Effectiveness of Randomized Signatures as Reservoir for Learning Rough Dynamics", *International Joint Conference on Neural Networks (IJCNN)*.

Cuchiero, C., L. Gonon, L. Grigoryeva, J.-P. Ortega, and J. Teichmann (2021). "Expressive Power of Randomized Signature", *NeurIPS 2021 Workshop on the Symbiosis of Deep Learning and Differential Equations (DLDE)*.

Cuchiero, C., L. Gonon, L. Grigoryeva, J.-P. Ortega, and J. Teichmann (2022). "Discrete-time Signatures and Randomness in Reservoir Computing", *IEEE Transactions on Neural Networks and Learning Systems* 33.11, pp. 6321–6330.

Cunningham, J., Z. Ghahramani, and C. Rasmussen (2012). "Gaussian Processes for time-marked time-series data", *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Cuturi, M. and M. Blondel (2017). "Soft-DTW: a Differentiable Loss Function for Time-Series", *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Cuturi, M., J.-P. Vert, Ø. Birkenes, and T. Matsui (2007). "A Kernel for Time Series Based on Global Alignments", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems* 2, pp. 303–314.

Dao, T. and A. Gu (2024). "Transformers are SSMs: generalized models and efficient algorithms through structured state space duality", *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Dasgupta, S. and A. Gupta (2003). "An elementary proof of a theorem of Johnson and Lindenstrauss", *Random Structures & Algorithms* 22.1, pp. 60–65.

Dauphin, Y. N., A. Fan, M. Auli, and D. Grangier (2017). "Language Modeling with Gated Convolutional Networks", *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Delétang, G., A. Ruoss, J. Grau-Moya, T. Genewein, L. K. Wenliang, E. Catt, C. Cundy, M. Hutter, S. Legg, J. Veness, and P. A. Ortega (2023). "Neural Networks and the Chomsky Hierarchy", *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Demšar, J. (2006). "Statistical Comparisons of Classifiers over Multiple Data Sets", *Journal of Machine Learning Research* 7.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database", *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 248–255.

Elfwing, S., E. Uchibe, and K. Doya (2018). "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning", *Neural Networks* 107. Special issue on deep reinforcement learning, pp. 3–11.

Elman, J. L. (1990). "Finding Structure in Time", *Cognitive Science* 14.2, pp. 179–211.

Faà di Bruno, F. (1855). "Sullo sviluppo delle funzioni", *Annali di Scienze Matematiche e Fisiche* 6, pp. 479–480.

Fan, T.-H., T.-C. Chi, and A. Rudnicky (2024). "Advancing Regular Language Reasoning in Linear Recurrent Neural Networks", *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 45–53.

Fefferman, C. (2006). "Whitney's extension problem for C^m", *Annals of Mathematics* 164.1, pp. 313–359.

Ferraty, F., A. Mas, and P. Vieu (2007). "Nonparametric regression on functional data: Inference and practical aspects", *Australian & New Zealand Journal of Statistics* 49.3, pp. 267–286.

Friz, P. K. and N. B. Victoir (2010). *Multidimensional Stochastic Processes as Rough Paths: Theory and Applications*. Cambridge Studies in Advanced Mathematics. Cambridge University Press.

Fu, D. Y., T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Re (2023). "Hungry Hungry Hippos: Towards Language Modeling with State Space Models", *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Galtieri, C. A. (1964). *Problems of Estimation in Discrete-Time Processes*. Tech. rep. RJ-315. San Jose, California: IBM Research Laboratory.

Garnelo, M., D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami (2018). "Conditional Neural Processes", *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Gelb, A. (1974). *Applied Optimal Estimation*. Cambridge, MA: MIT Press. Chap. 6.

Giles, C. L., G. Z. Sun, H. H. Chen, Y. C. Lee, and D. Chen (1989). "Higher Order Recurrent Networks and Grammatical Inference", *Proceedings of the 3rd International Conference on Neural Information Processing Systems (NeurIPS)*.

Goel, K., A. Gu, C. Donahue, and C. Ré (2022). "It's Raw! Audio Generation with State-Space Models", *Proceedings of the 39th International Conference on Machine Learning (ICML)*.

Goldberger, A. L., L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley (2000). "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals", *Circulation* 101.23, e215–e220.

Gordon, N., D. Salmond, and A. Smith (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", *IEE Proceedings F (Radar and Signal Processing)* 140 (2), pp. 107–113.

Graham, B. (2013). "Sparse arrays of signatures for online character recognition", *arXiv preprint arXiv:1308.0371*.

Grazzi, R., J. Siems, J. K. H. Franke, A. Zela, F. Hutter, and M. Pontil (2025). "Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues", *Proceedings of the 13th International Conference on Learning Representations (ICLR)*.

Griewank, A. and A. Walther (2008). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. 2nd. Society for Industrial and Applied Mathematics.

Gu, A. and T. Dao (2024). "Mamba: Linear-Time Sequence Modeling with Selective State Spaces", *Proceedings of the First Conference on Language Modeling*.

Gu, A., K. Goel, and C. Ré (2022a). "Efficiently Modeling Long Sequences with Structured State Spaces", *Proceedings of The 10th International Conference on Learning Representations (ICLR)*.

Gu, A., A. Gupta, K. Goel, and C. Ré (2022b). "On the Parameterization and Initialization of Diagonal State Space Models", *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Gyurkó, L. G., T. Lyons, M. Kontkowski, and J. Field (2014). "Extracting information from the signature of a financial data stream", *arXiv preprint arXiv:1307.7244*.

Hall, M. (1950). "A basis for free Lie rings and higher commutators in free groups", *Proceedings of the American Mathematical Society*. Vol. 1, pp. 575–581.

Hambly, B. and T. Lyons (2010). "Uniqueness for the signature of a path of bounded variation and the reduced path group", *Annals of Mathematics* 171, pp. 109–167.

Hanson, J. and M. Raginsky (2020). "Universal Simulation of Stable Dynamical Systems by Recurrent Neural Nets", *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, pp. 384–392.

He, K., X. Zhang, S. Ren, and J. Sun (2016). "Deep Residual Learning for Image Recognition", *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hess, K., V. Melnychuk, D. Frauen, and S. Feuerriegel (2024). "Bayesian Neural Controlled Differential Equations for Treatment Effect Estimation", *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Heun, K. (1900). “Neue Methode zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen”, *Zeitschrift für Mathematik und Physik* 45, pp. 23–38.

Hinton, G. E. (1987). “Learning Translation Invariant Recognition in Massively Parallel Networks”, *Proceedings of the Parallel Architectures and Languages Europe, Volume I: Parallel Architectures*. Berlin, Heidelberg: Springer-Verlag, pp. 1–13.

Hochreiter, S. (1991). “Untersuchungen zu dynamischen neuronalen Netzen”. PhD thesis. Technische Universität München.

Hochreiter, S., Y. Bengio, P. Frasconi, and J. Schmidhuber (2001). “Gradient Flow in Recurrent Nets: The Difficulty of Learning Long–Term Dependencies”, *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. Wiley-IEEE Press, pp. 237–244.

Hochreiter, S. and J. Schmidhuber (1997). “Long Short-term Memory”, *Neural computation* 9.8, pp. 1735–80.

Holberg, C. and C. Salvi (2024). “Exact Gradients for Stochastic Spiking Neural Networks Driven by Rough Signals”, *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Hornik, K. (1991). “Approximation capabilities of multilayer feedforward networks”, *Neural Networks* 4.2, pp. 251–257.

Horvath, B., M. Lemercier, C. Liu, T. Lyons, and C. Salvi (2023). “Optimal Stopping via Distribution Regression: A Higher Rank Signature Approach”, *arXiv preprint arXiv:2304.01479*.

Ioffe, S. and C. Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Jelassi, S., D. Brandfonbrener, S. M. Kakade, and E. Malach (2024). “Repeat after me: Transformers are better than state space models at copying”, *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Jiang, L., W. Yang, X. Zhang, and H. Ni (2024). “GCN-DevLSTM: Path Development for Skeleton-Based Action Recognition”, *arXiv preprint, arXiv.2403.15212*.

Julier, S. J. and J. K. Uhlmann (1997). “New extension of the Kalman filter to nonlinear systems”, *Signal Processing, Sensor Fusion, and Target Recognition VI*. Vol. 3068. International Society for Optics and Photonics. SPIE, pp. 182–193.

Kalman, R. E. (1960). “A New Approach to Linear Filtering and Prediction Problems”, *ASME Journal of Basic Engineering* 82, pp. 35–45.

Katharopoulos, A., A. Vyas, N. Pappas, and F. Fleuret (2020). "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention", *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Kermack, W. and A. McKendrick (1927). "A contribution to the mathematical theory of epidemics", *Proceedings of the Royal Society of London. Series A* 115.772, pp. 700–721.

Kidger, P. (2022). "On Neural Differential Equations". PhD thesis. University of Oxford.

Kidger, P., J. Foster, X. Li, and T. J. Lyons (2021). "Neural SDEs as Infinite-Dimensional GANs", *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Kidger, P., J. Morrill, J. Foster, and T. Lyons (2020). "Neural Controlled Differential Equations for Irregular Time Series", *Proceedings of the 34th Conference on Neural Information Processing System (NeurIPS)*.

Kingma, D. P. and J. Ba (2015). "Adam: A Method for Stochastic Optimization", *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Király, F. J. and H. Oberhauser (2019). "Kernels for sequentially ordered data", *Journal of Machine Learning Research* 20.31, pp. 1–45.

Kolmogorov, A. N. (1941). "Интерполирование и экстраполирование стационарных случайных последовательностей", *Известия АН СССР. Серия математическая* 5.1, pp. 3–14.

Kolmogorov, A. N. (1962). *Interpolation and Extrapolation of Stationary Random Sequences*. Tech. rep. RM-3090-PR. English translation by Worthie L. Doyle and Ivan Selin. Santa Monica, CA: RAND Corporation.

Kreyszig, E. (1978). *Introductory Functional Analysis with Applications*. John Wiley & Sons.

Krogh, A. and J. A. Hertz (1991). "A Simple Weight Decay Can Improve Generalization", *Proceedings of the 5th International Conference on Neural Information Processing Systems (NeurIPS)*.

Lang, S. (1999). *Fundamentals of Differential Geometry*. Vol. 191. Graduate Texts in Mathematics. Springer.

Lee, J. M. (2013). *Introduction to Smooth Manifolds*. 2nd. Vol. 218. Graduate Texts in Mathematics. New York: Springer.

Lee, J., T. Kang, N. Kim, S. Han, H. Won, W. Gong, and I. Kwak (2022). "Deep Learning Based Heart Murmur Detection Using Frequency-time Domain Features of Heartbeat Sounds", *Computing in Cardiology (CinC)*. Vol. 49.

Lee, K., M. Kim, Y. Jun, and S. S. Woo (2024). "GDFlow: Anomaly Detection with NCDE-based Normalizing Flow for Advanced Driver Assistance System", *arXiv preprint arXiv:2409.05346*.

Lemercier, M., C. Salvi, T. Cass, E. V. Bonilla, T. Damoulas, and T. Lyons (2021a). "SigGPDE: Scaling Sparse Gaussian Processes on Sequential Data", *International Conference on Machine Learning (ICML)*. PMLR.

Lemercier, M., C. Salvi, T. Damoulas, E. Bonilla, and T. Lyons (2021b). "Distribution regression for sequential data", *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR.

Levin, D., T. Lyons, and H. Ni (2016). "Learning from the past, predicting the statistics for the future, learning an evolving system", *arXiv preprint arXiv:1309.0260*.

Li, X., J. Lu, R. Arora, J. Haupt, H. Liu, Z. Wang, and T. Zhao (2019). "Symmetry, Saddle Points, and Global Optimization Landscape of Nonconvex Matrix Factorization", *IEEE Transactions on Information Theory* 65.6, pp. 3489–3514.

Li, Y., T. Cai, Y. Zhang, D. Chen, and D. Dey (2023). "What Makes Convolutional Models Great on Long Sequence Modeling?", *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Li, Z., J. Han, W. E, and Q. Li (2022). "Approximation and Optimization Theory for Linear Continuous-Time Recurrent Neural Networks", *Journal of Machine Learning Research* 23.42.

Liao, S., T. Lyons, W. Yang, K. Schlegel, and H. Ni (2021). "Logsig-RNN: a novel network for robust and efficient skeleton-based action recognition", *Proceedings of the 32nd British Machine Vision Conference (BMVC 2021)*.

Lim, S., J. Choi, N. Park, S.-H. Yoon, S. Kang, Y.-M. Kim, and H. Kang (2024). "Bridging Dynamic Factor Models and Neural Controlled Differential Equations for Nowcasting GDP", *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*.

Lizaire, M., M. Rizvi-Martel, M. Gamal, and G. Rabusseau (2024). "A Tensor Decomposition Perspective on Second-order RNNs", *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Lotka, A. J. (1925). *Elements of Physical Biology*. Williams & Wilkins Company.

Lou, H., S. Li, and H. Ni (2024). "Path Development Network with Finite-dimensional Lie Group Representation", *Transactions on Machine Learning Research*.

Lu, C., Y. Schroecker, A. Gu, E. Parisotto, J. Foerster, S. Singh, and F. Behbahani (2023). "Structured State Space Models for In-Context Reinforcement Learning",

Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS).

Lu, H., J. B. Yip, T. Steigleder, S. Grießhammer, N. V. S. J. Jami, B. Eskofier, C. Ostgathe, and A. Koelpin (2022). "A Lightweight Robust Approach for Automatic Heart Murmurs and Clinical Outcomes Classification from Phonocardiogram Recordings", *Computing in Cardiology (CinC)*. Vol. 49.

Luenberger, D. G. (1969). *Optimization by Vector Space Methods*. New York: John Wiley & Sons.

Lyons, T., M. Caruana, and T. Lévy (2007). *Differential Equations Driven by Rough Paths: École D'été de Probabilités de Saint-Flour XXXIV-2004*. no. 1908. Springer.

Lyons, T. (1994). "Differential Equations Driven by Rough Signals (I): An Extension of an Inequality of L. C. Young", *Mathematical Research Letters* 1, pp. 451–464.

Lyons, T. (1998). "Differential equations driven by rough signals.", *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

Lyons, T. (2014). "Rough Paths, Signatures and the Modelling of Functions on Streams", *Proceedings of the International Congress of Mathematicians (ICM)*. Vol. 4.

Lyons, T. and A. McLeod (2025). "Higher order Lipschitz Sandwich theorems", *Journal of the London Mathematical Society* 111.3.

Lyons, T. and Z. Qian (2002). *System Control and Rough Paths*. Oxford Mathematical Monographs. Oxford University Press.

Magnus, W. (1954). "On the exponential solution of differential equations for a linear operator", *Communications on Pure and Applied Mathematics* 7.4, pp. 649–673.

Manten, G., C. Casolo, E. Ferrucci, S. W. Mogensen, C. Salvi, and N. Kilbertus (2025). "Signature Kernel Conditional Independence Tests in Causal Discovery for Stochastic Processes", *Proceedings of the 13th International Conference on Learning Representations (ICLR)*.

McDonald, A., M. J. F. Gales, and A. Agarwal (2022). "Detection of Heart Murmurs in Phonocardiograms with Parallel Hidden Semi-Markov Models", *Computing in Cardiology (CinC)*. Vol. 49.

McLeod, A. and T. Lyons (2025). "Signature methods in machine learning", *EMS Surveys in Mathematical Sciences*.

Meng, Q., S. Zheng, H. Zhang, W. Chen, Q. Ye, Z.-M. Ma, N. Yu, and T.-Y. Liu (2019). "G-SGD: Optimizing ReLU Neural Networks in Its Positively Scale-Invariant Space", *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Merrill, W., J. Petty, and A. Sabharwal (2024). "The illusion of state in state-space models", *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Mikalsen, K. Ø., F. M. Bianchi, C. Soguero-Ruiz, and R. Jenssen (2018). "Time series cluster kernel for learning similarities between multivariate time series with missing data", *Pattern Recognition* 76, pp. 569–581.

Moan, P. C. and J. Niesen (2008). "Convergence of the Magnus Series", *Foundations of Computational Mathematics* 8, pp. 291–301.

Moore, P. J., T. J. Lyons, J. Gallacher, and Alzheimer's Disease Neuroimaging Initiative (2019). "Using path signatures to predict a diagnosis of Alzheimer's disease", *PLoS One* 14.9.

Moreno-Pino, F., Á. Arroyo, H. Waldon, X. Dong, and Á. Cartea (2024). "Rough Transformers: Lightweight and Continuous Time-Series Modelling through Signature Patching", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Morrill, J., P. Kidger, L. Yang, and T. Lyons (2022). "On the Choice of Interpolation Scheme for Neural CDEs", *Transactions on Machine Learning Research* 2022.9.

Morrill, J., A. Kormilitzin, A. Nevado-Holgado, S. Swaminathan, S. Howison, and T. Lyons (2019). "The Signature-Based Model for Early Detection of Sepsis From Electronic Health Records in the Intensive Care Unit", *Computing in Cardiology (CinC)*.

Morrill, J., C. Salvi, P. Kidger, J. Foster, and T. Lyons (2021). "Neural Rough Differential Equations for Long Time Series", *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Movahedi, S., F. Sarnthein, N. M. Cirone, and A. Orvieto (2025). "Fixed-Point RNNs: From Diagonal to Dense in a Few Iterations", *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*.

Nguyen, E., K. Goel, A. Gu, G. W. Downs, P. Shah, T. Dao, S. A. Baccus, and C. Ré (2022). "S4ND: Modeling Images and Videos as Multidimensional Signals Using State Spaces", *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Nyquist, H. (1928). "Certain topics in telegraph transmission theory", *Transactions of the American Institute of Electrical Engineers* 47.2, pp. 617–644.

O'Hagan, A. (1978). "Curve Fitting and Optimal Design for Prediction", *Journal of the Royal Statistical Society. Series B (Methodological)* 40.1, pp. 1–42.

Orvieto, A., S. De, C. Gulcehre, R. Pascanu, and S. L. Smith (2024). "Universality of Linear Recurrences Followed by Non-linear Projections: Finite-Width Guarantees and Benefits of Complex Eigenvalues", *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Orvieto, A., S. L. Smith, A. Gu, A. Fernando, Ç. Gülçehre, R. Pascanu, and S. De (2023). "Resurrecting Recurrent Neural Networks for Long Sequences", *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *arXiv preprint arXiv:1912.01703*.

Pearlmutter, B. A. (1989). "Learning State Space Trajectories in Recurrent Neural Networks", *Neural Computation* 1.2, pp. 263–269.

Peng, B., E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, L. Derczynski, X. Du, M. Grella, K. Gv, X. He, H. Hou, P. Kazienko, J. Kocon, J. Kong, B. Koptyra, H. Lau, J. Lin, K. S. I. Mantri, F. Mom, A. Saito, G. Song, X. Tang, J. Wind, S. Woźniak, Z. Zhang, Q. Zhou, J. Zhu, and R.-J. Zhu (2023). "RWKV: Reinventing RNNs for the Transformer Era", *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Peng, B., R. Zhang, D. Goldstein, E. Alcaide, X. Du, H. Hou, J. Lin, J. Liu, J. Lu, W. Merrill, G. Song, K. Tan, S. Utpala, N. Wilce, J. S. Wind, T. Wu, D. Wuttke, and C. Zhou-Zheng (2025). "RWKV-7 'Goose with Expressive Dynamic State Evolution", *arXiv preprint arXiv:2503.14456*.

Peng, H., N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong (2021). "Random Feature Attention", *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.

Perez Arribas, I. (2018). "Derivatives pricing using signature payoffs", *arXiv preprint arXiv:1809.09466*.

Perez Arribas, I., G. M. Goodwin, J. R. Geddes, T. Lyons, and K. E. A. Saunders (2018). "A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder", *Transl Psychiatry* 8.1.

Perez Arribas, I., C. Salvi, and L. Szpruch (2020). "Sig-SDEs model for quantitative finance", *ACM International Conference on AI in Finance*.

Pineda, F. J. (1987). "Generalization of Back-Propagation to Recurrent Neural Networks", *Physical Review Letters* 59.19, pp. 2229–2232.

Qin, T., B. Walker, T. Lyons, H. Yan, and H. Li (2025). "Learning dynamic graph embeddings with neural controlled differential equations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Qin, Z., S. Yang, W. Sun, X. Shen, D. Li, W. Sun, and Y. Zhong (2024). "HGRN2: Gated linear RNNs with state expansion", *Proceedings of the 1st Conference on Language Modeling (COLM)*.

Ramsay, J. O. and B. W. Silverman (2005). *Functional Data Analysis*. 2nd ed. New York: Springer.

Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Ree, R. (1958). "Lie Elements and an Algebra Associated With Shuffles", *Annals of Mathematics* 68.2.

Reiss, A., I. Indlekofer, P. Schmidt, and K. Van Laerhoven (2019). "Deep PPG: Large-Scale Heart Rate Estimation with Convolutional Neural Networks", *Sensors* 19.14.

Reutenauer, C. (1993). *Free Lie Algebras*. London Mathematical Society Monographs. Clarendon Press.

Reyna, M. A., Y. Kiarashi, A. Elola, J. Oliveira, F. Renna, A. Gu, E. A. Perez Alday, N. Sadr, A. Sharma, J. Kpodonu, S. Mattos, M. T. Coimbra, R. Sameni, A. B. Rad, and G. D. Clifford (2023). "Heart murmur detection from phonocardiogram recordings: The George B. Moody PhysioNet Challenge 2022", *PLOS Digital Health* 2.9.

Rico-Martínez, R., K. Krischer, I. G. Kevrekidis, M. C. Kube, and J. L. Hudson (1992). "Discrete-vs. continuous-time nonlinear signal processing of Cu electrodissolution data", *Chemical Engineering Communications* 118.1, pp. 25–48.

Roman, S. (2007). *Advanced Linear Algebra*. Graduate Texts in Mathematics. Springer New York. Chap. 14.

Rubanova, Y., T. Q. Chen, and D. K. Duvenaud (2019). "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series", *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*.

Rudin, W. (1991). *Functional Analysis*. 2nd. New York: McGraw-Hill.

Ryan, R. A. (2002). *Introduction to Tensor Products of Banach Spaces*. Vol. 73. Springer Monographs in Mathematics. Springer.

Saarinen, S., R. Bramley, and G. Cybenko (1993). "Ill-Conditioning in Neural Network Training Problems", *SIAM Journal on Scientific Computing* 14.3, pp. 693–714.

Salvi, C. (2021). "Rough paths, kernels, differential equations and an algebra of functions on streams". PhD thesis. University of Oxford.

Salvi, C., T. Cass, J. Foster, T. Lyons, and W. Yang (2021). "The Signature Kernel is the solution of a Goursat PDE", *SIAM Journal on Mathematics of Data Science* 3.3, pp. 873–899.

Salvi, C., J. Diehl, T. Lyons, R. Preiss, and J. Reizenstein (2023). "A structure theorem for streamed information", *Journal of Algebra* 634, pp. 911–938.

Schlag, I., K. Irie, and J. Schmidhuber (2021). "Linear Transformers Are Secretly Fast Weight Programmers", *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Seedat, N., F. Imrie, A. Bellot, Z. Qian, and M. van der Schaar (2022). "Continuous-Time Modeling of Counterfactual Outcomes Using Neural Controlled Differential Equations", *Proceedings of the 39th International Conference on Machine Learning (ICML)*.

Shannon, C. E. (1949). "Communication in the presence of noise", *Proceedings of the IRE* 37.1, pp. 10–21.

Shmelev, D. and C. Salvi (2024). "Sparse Signature Coefficient Recovery via Kernels", *arXiv preprint arXiv:2412.08579*.

Siegelmann, H. T. and E. D. Sontag (1992). "On the computational power of neural nets", *Proceedings of the 5th Annual Workshop on Computational Learning Theory (COLT)*, pp. 440–449.

Siems, J., T. Carstensen, A. Zela, F. Hutter, M. Pontil, and R. Grazzi (2025). "DeltaProduct: Improving State-Tracking in Linear RNNs via Householder Products", *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*.

Slutsky, E. E. (1927). "The Summation of Random Causes as the Source of Cyclical Processes", *Voprosy Koniunktury* 3.1, pp. 34–64.

Slutsky, E. E. (1937). "The Summation of Random Causes as the Source of Cyclical Processes", *Econometrica* 5.2. English translation of Slutsky (1927), pp. 105–146.

Smith, J. T. H., A. Warrington, and S. W. Linderman (2023). "Simplified State Space Layers for Sequence Modeling", *Proceedings of The 11th International Conference on Learning Representations (ICLR)*.

Solow, R. M. (1956). "A Contribution to the Theory of Economic Growth", *The Quarterly Journal of Economics* 70.1, pp. 65–94.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research* 15.56, pp. 1929–1958.

Stein, E. M. (1970). *Singular Integrals and Differentiability Properties of Functions*. Princeton University Press.

Stieltjes, T. J. (1894). "Recherches sur les fractions continues", *Annales de la Faculté des Sciences de Toulouse pour les Sciences Mathématiques et les Sciences Physiques* 8.4, pp. 1–122.

Stone, M. H. (1948). "The Generalized Weierstrass Approximation Theorem", *Mathematics Magazine* 21.4, pp. 167–184.

Strichartz, R. S. (1987). "The Campbell–Baker–Hausdorff–Dynkin Formula and Solutions of Differential Equations", *Journal of Functional Analysis* 72.2, pp. 320–345.

Sun, Y., X. Li, K. Dalal, J. Xu, A. Vikram, G. Zhang, Y. Dubois, X. Chen, X. Wang, S. Koyejo, T. Hashimoto, and C. Guestrin (2025). "Learning to (Learn at Test Time): RNNs with Expressive Hidden States", *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Sutskever, I., J. Martens, and G. E. Hinton (2011). "Generating Text with Recurrent Neural Networks", *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus (2014). "Intriguing properties of neural networks", *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*.

Tay, Y., M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler (2021). "Long Range Arena: A Benchmark for Efficient Transformers", *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.

Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample (2023). "LLaMA: Open and Efficient Foundation Language Models", *arXiv pre-print arXiv:2302.13971*.

Tsitouras, C. (2011). "Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption", *Computers & Mathematics with Applications* 62.2, pp. 770–775.

Van Overschee, P. and B. De Moor (1994). "N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems", *Automatica* 30.1, pp. 75–93.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). "Attention Is All You Need", *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*.

Vauvelle, A., P. Creed, and S. Denaxas (2022). "Neural-signature methods for structured EHR prediction", *BMC Medical Informatics and Decision Making* 22.1.

Volterra, V. (1926). "Fluctuations in the abundance of a species considered mathematically", *Nature* 118, pp. 558–560.

Walker, B., F. Krones, I. Kiskin, G. Parsons, T. Lyons, and A. Mahdi (2022). "Dual Bayesian ResNet: A Deep Learning Approach to Heart Murmur Detection", *Computing in Cardiology (CinC)*. Vol. 49.

Walker, B., A. D. McLeod, T. Qin, Y. Cheng, H. Li, and T. Lyons (2024). "Log Neural Controlled Differential Equations: The Lie Brackets Make a Difference", *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Walker, B., L. Yang, N. M. Cirone, C. Salvi, and T. Lyons (2025). "Structured Linear CDEs: Maximally Expressive and Parallel-in-Time Sequence Models", *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*.

Walker, G. T. (1931). "On Periodicity in Series of Related Terms", *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences* 131.818, pp. 518–532.

Wang, B., M. Liakata, H. Ni, T. Lyons, A. J. Nevado-Holgado, and K. Saunders (2019). "A Path Signature Approach for Speech Emotion Recognition", *Proc. Interspeech 2019*, pp. 1661–1665.

Wang, J., J. N. Yan, A. Gu, and A. M. Rush (2023a). "Pretraining Without Attention", *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Wang, S. and B. Xue (2023b). "State-space Models with Layer-wise Nonlinearity are Universal Approximators with Exponential Decaying Memory", *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*.

Weierstrass, K. (1885). "Uber die analytische Darstellbarkeit sogenannter willkurlicher Functionen einer reellen Veranderlichen", *Sitzungsberichte der Akademie zu Berlin*, pp. 633–639, 789–805.

Wells, J. C. (1973). "Differentiable functions on Banach spaces with Lipschitz derivatives", *Journal of Differential Geometry* 8.1, pp. 135–152.

Whitney, H. (1934). "Analytic Extensions of Differentiable Functions Defined in Closed Sets", *Transactions of the American Mathematical Society* 36.1, pp. 63–89.

Whittle, P. (1951). *Hypothesis Testing in Time Series Analysis*. Almqvist & Wiksells.

Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. Cambridge, MA: The MIT Press.

Wilcoxon, F. (1945). "Individual Comparisons by Ranking Methods", *Biometrics Bulletin* 1.6.

Williams, C. K. I. and C. E. Rasmussen (1996). "Gaussian Processes for Regression", *Advances in Neural Information Processing Systems 8*. MIT Press, pp. 514–520.

Wynne, G. and A. B. Duncan (2022). "A kernel two-sample test for functional data", *Journal of Machine Learning Research* 23.

Xu, H. and S. Mannor (2012). "Robustness and Generalization", *Machine Learning* 86, pp. 391–423.

Xu, Y., X. Bao, H. Lam, and E. N. Kamavuako (2022). "Hierarchical Multi-Scale Convolutional Network for Murmurs Detection on PCG Signals", *Computing in Cardiology (CinC)*. Vol. 49.

Yang, S., J. Kautz, and A. Hatamizadeh (2025). "Gated Delta Networks: Improving Mamba2 with Delta Rule", *Proceedings of the 13th International Conference on Learning Representations (ICLR)*.

Yang, S., B. Wang, Y. Shen, R. Panda, and Y. Kim (2024a). "Gated Linear Attention Transformers with Hardware-Efficient Training", *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Yang, S., B. Wang, Y. Zhang, Y. Shen, and Y. Kim (2024b). "Parallelizing Linear Transformers with the Delta Rule over Sequence Length", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Yang, W., L. Jin, H. Ni, and T. Lyons (2016). "Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network", *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR)*.

Young, L. C. (1936). "An inequality of the Hölder type, connected with Stieltjes integration", *Acta Mathematica* 67, pp. 251–282.

Yue, B., J. Fu, and J. Liang (2018). "Residual Recurrent Neural Networks for Learning Sequential Representations", *Information* 9.3.

Yule, G. U. (1927). "On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers", *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 226, pp. 267–298.

Zeng, S., L. J. Liu, J. Wen, M. Yetisgen, R. Etzioni, and G. Luo (2025). "TrajSurv: Learning Continuous Latent Trajectories from Electronic Health Records for Trustworthy Survival Prediction", *Proceedings of the Machine Learning for Healthcare (MLHC) Conference*.

Zhang, Y., S. Yang, R. Zhu, Y. Zhang, L. Cui, Y. Wang, B. Wang, F. Shi, B. Wang, W. Bi, P. Zhou, and G. Fu (2024). "Gated Slot Attention for Efficient Linear-Time

Sequence Modeling", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.

Zucchet, N. and A. Orvieto (2024). "Recurrent neural networks: vanishing and exploding gradients are not the end of the story", *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*.