Rough paths, kernels, differential equations and an algebra of functions on streams

Cristopher Salvi

St Edmund Hall

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2021

This thesis is dedicated *al nonno Bruno*

Acknowledgements

I deeply thank my supervisor Terry Lyons for his guidance and patience during these years. Our early morning discussions on the train to London, or in the tube on our way back to Oxford were maybe strange to hear for other passengers but memorable for us students. Alternating periods of excitement with ones of frustration taught me the essence of mathematical research. Our interaction has had, is having and will certainly continue to have a deep effect on my future personal and professional trajectories.

I feel blessed for having been part of a group of extremely smart and talented individuals with whom it was a pleasure to collaborate, exchange ideas and travel abroad. Finally, this experience would not have been the same without the support of my bro. Pippi, my partner (in crime) Maud, my parents Sonia & Pietro and my *nonni* Lina & Bruno: thanks for all you have done for me.

Abstract

This thesis is organised in the following four chapters. Appendix A provides a summary of rough path theory.

Chapter 1: The Signature Kernel Recently, there has been an increased interest in the development of kernel methods for learning with sequential data. The signature kernel is a learning tool with potential to handle irregularly sampled, multivariate time series. In [1] the authors introduced a kernel trick for the truncated version of this kernel avoiding the exponential complexity that would have been involved in a direct computation. In this chapter we show that for continuously differentiable paths, the signature kernel solves a hyperbolic PDE and recognize the connection with a well known class of differential equations known in the literature as Goursat problems. This Goursat PDE only depends on the increments of the input sequences, does not require the explicit computation of signatures and can be solved efficiently using state-of-the-art hyperbolic PDE numerical solvers, giving a kernel trick for the untruncated signature kernel, with the same raw complexity as the method from [1], but with the advantage that the PDE numerical scheme is well suited for `GPU` parallelization, which effectively reduces the complexity by a full order of magnitude in the length of the input sequences. In the last section of this chapter we extend the previous analysis to the space of geometric rough paths and establish, using classical results from rough path theory, that the rough version of the signature kernel solves a rough integral equation analogous to the aforementioned Goursat problem. Empirical evidence is given on the effectiveness of this PDE kernel as a machine learning tool in various data science applications. We release the python library `sigkernel` publicly available at `https://github.com/crispitagorico/sigkernel`.

Chapter 2: Distribution Regression for Sequential Data Distribution Regression refers to the supervised learning problem where inputs are probability distributions (or some samples) and outputs are scalars. In practice it can be used to address situations where labels are only available for groups of

inputs instead of individual inputs. In this chapter, we develop a rigorous mathematical framework for distribution regression where inputs are streams of data. Leveraging properties established in **Chapter 1** two new learning techniques are introduced, one feature-based and the other kernel-based. Each is suited to a different data regime in terms of the number of data streams and the dimensionality of the individual streams. We provide theoretical results on the universality of both approaches and demonstrate empirically their robustness to irregularly sampled time-series, achieving state-of-the-art performance on examples from thermodynamics, agricultural science and cyber security.

Chapter 3: Neural Differential Equations Neural controlled differential equations (CDEs) are the continuous-time analogue of recurrent neural networks (RNNs), as Neural ordinary differential equations (ODEs) are to residual networks (ResNets), and offer a memory-efficient continuous-time way to model functions of potentially irregular time series. Existing methods for computing the forward pass of a Neural CDE involve embedding the incoming time series into path space via interpolation and using evaluations of this path to drive the hidden state. Using rough path theory it is possible to extend this formulation. Instead of directly embedding the data into a path via interpolation, we represent the input signal over small time intervals through its log-signature. This approach derives from the numerical log-ODE method, an efficient numerical method for solving rough differential equations (RDEs). Experimental results suggest that this extension is particularly well suited for problems with long time series.

Chapter 4: Structure theorems for streamed information This chapter provides an algebraic description of some classes of real valued functions on streams. More precisely we identify explicit bases for the algebra of linear functionals on grouplike elements with point-wise multiplication, or equivalently for the space of continuous functions on unparametrized paths. The results can be interpreted as structure theorems for streams, splitting functions on streams (possibly uniquely) into two components: a) a first that extracts and packages the information of a stream into recursively defined atomic objects and b) a second that evaluates a polynomial in these elementary components without further reference to the original stream. The atomic objects in a) are described via different algebraic products and using classical combinatorial objects (Hall sets) arising in the study of free Lie algebras.

Contents

Chapter 1

The Signature Kernel

Nowadays, sequential data is being produced and stored at an unprecedented rate. Examples include daily fluctuations of asset prices in the stock market, medical and biological records, readings from mobile apps, weather measurements etc. An efficient learning algorithm must be able to handle data streams that are often irregularly sampled and/or partially observed and at the same time scale well with a high number of channels.

An important obstacle that most machine learning models have to face is the potential symmetry present in the data. In computer vision for example, a good model should be able to recognize an image even if the latter is rotated by a certain angle. The 3D rotation group, often denoted $SO(3)$, is low dimensional (3), therefore it is relatively easy to add components to a model that build a rotation invariance, for example through data-augmentation. However, when dealing with sequential data one is confronted with a much bigger (infinite dimensional) group of symmetries given by all (time) reparametrizations of a path, i.e. continuous, increasing surjective functions from I to itself, where I is a closed (time) interval over which the path is defined. For example, consider the reparametrization $\phi : [0,1] \to [0,1]$ given by $\phi(t) = t^2$ and the path $\gamma : [0,1] \to \mathbb{R}^2$ defined by $\gamma_t = (\gamma_t^x, \gamma_t^y)$ where $\gamma_t^x = \cos(10t)$ and $\gamma_t^y = \sin(3t)$. As it is clearly depicted in Figure 1.1, both channels (γ^x, γ^y) of γ are individually affected by the reparametrization ϕ, but the shape of the curve γ is left unchanged.

Definition 1.0.1. Let V be a Banach space. Consider the space of formal polynomials $T(V) = \bigoplus_{k=0}^{\infty} V^{\otimes k}$ (truncated at any finite level) over V, where $\otimes$ denotes the (classical) tensor product of vector spaces, and the space of formal power series $T((V)) = \bigoplus_{k=0}^{\infty} V^{\otimes k}$ (untruncated) over V. Both $T(V)$ and $T((V))$ can be endowed with the operations of addition $+$ and multiplication $\otimes$ defined for any two elements $A = (a_0, a_1, \cdots)$ and $B = (b_0, b_1, \cdots)$

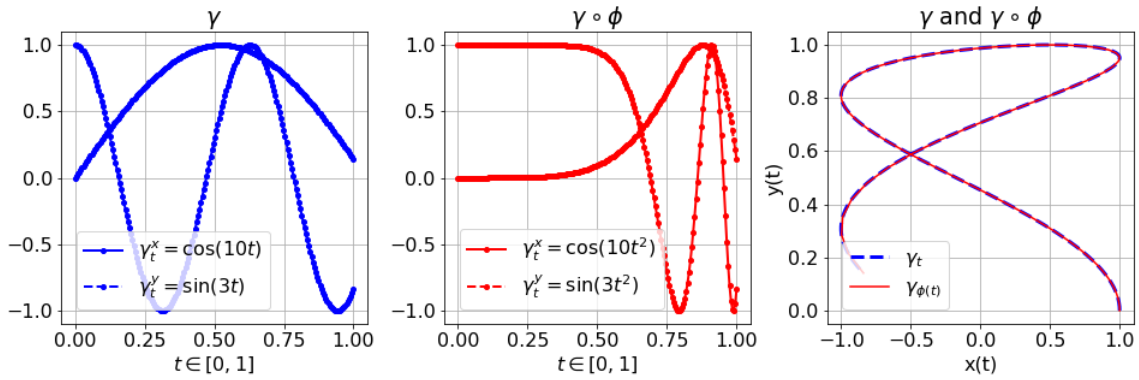


Figure 1.1: On the **left** are the individual channels (γ^x, γ^y) of a 2D paths γ. In the **middle** are the channels reparametrized under $\phi : t \mapsto t^2$. On the **right** are the path γ and its reparametrized version $\gamma \circ \phi$. The two curves overlap, meaning that the reparametrization ϕ represents irrelevant information if one is interested in understanding the shape of γ.

respectively as

$$A + B = (a_0 + b_0, a_1 + b_1, ...)\,, \tag{1.1}$$

$$A \otimes B = (c_0, c_1, c_2, ...)\,, \quad \text{where} \quad V^{\otimes k} \ni c_k = \sum_{i=0}^{k} a_i \otimes b_{k-i}, \quad \forall k \geq 0. \tag{1.2}$$

When endowed with these two operations and the natural action of $\mathbb{R}$ by $\lambda A = (\lambda a_0, \lambda a_1, ...)$, $T((V))$ becomes a real, non-commutative unital algebra with unit $\mathbf{1} = (1, 0, 0, ...)$ called the tensor algebra. The truncated tensor algebra over V of order $N \in \mathbb{N}$ is defined as the quotient $T^N(V) = T((V))/T_N$ by the ideal

$$T_N = \{A = (a_0, a_1, ...) \in T((V)) : a_0 = ... = a_N = 0\}. \tag{1.3}$$

A continuous path $x : I \to V$ is said to be of bounded variation if

$$\sup_{\mathcal{D}} \sum_{t_i \in \mathcal{D}} |x_{t_{i+1}} - x_{t_i}| < +\infty, \tag{1.4}$$

where the supremum is taken over all partitions $\mathcal{D}$ of an interval I, i.e. over all increasing sequences of ordered (time) indices such that $\mathcal{D} = \{0 = k_0 < k_1 < ... < k_r = T\}$.

Definition 1.0.2 (**p-variation**)**.** Let $p \geq 1$ be a real number and $x : I \to V$ be a continuous path. The p-variation of x on the interval I is defined by

$$||x||_{p,I} = \left(\sup_{\mathcal{D}} \sum_{t_i \in \mathcal{D}} |x_{t_{i+1}} - x_{t_i}|^p \right)^{1/p}. \tag{1.5}$$

Remark 1.0.1. For any continuous path $x : I \to V$, any time-reparametrization $\psi : I \to I$ and scalar $p \geq 1$ one has $||x||_{p,I} = ||x \circ \psi||_{p,I}$; in other words the p-variation of x is invariant to time-reparametrization. Furthermore, the function $p \to ||x||_{p,I}$ is decreasing. Hence, any path of finite p-variation is also a path of finite q-variation for any $q > p$.

Definition 1.0.3 (**Signature**)**.** Let $I \subset \mathbb{R}$ be a compact interval, V a Banach space and let $x : I \to V$ be a continuous path of finite p-variation with $p < 2$. For any $s, t \in I$ such that $s \leq t$, the signature $S(x)_{[s,t]} \in T((V))$ of the path x over the sub-interval $[s, t]$ is defined as the following infinite collection of iterated integrals

$$S(x)_{[s,t]} = \left(1, \int\limits_{s<u_1<t} dx_{u_1}, ..., \int\limits_{s<u_1<...<u_k<t} ... \int dx_{u_1} \otimes ... \otimes dx_{u_k}, ...\right). \tag{1.6}$$

The signature of a path x is invariant under reparametrization, therefore it acts on x as a filter that systematically removes this troublesome, infinite dimensional group of symmetries. Furthermore, it turns out that linear functionals acting on the range of the signature and separate points [2, chapter 2] and form an algebra (with pointwise multiplication) as stated in the following lemma.

Lemma 1.0.0.1 (**Shuffle identity**)**.** [2, Lemma 2.17] Let $L_1, L_2 : T(V) \to \mathbb{R}$ be two linear functionals. Then for any continuous path x of bounded variation

$$L_1(S(x))L_2(S(x)) = (L_1 \sqcup\!\sqcup L_2)(S(x)) \tag{1.7}$$

where $\sqcup\!\sqcup$ is the shuffle product, that will extensively studied in Chapter 4.

Hence, by the *Stone-Weierstrass theorem*, for any compact set C of continuous paths of bounded variation, the set of linear functionals on signatures of paths from C is dense in the set of continuous real-valued functions on C.

Theorem 1.0.1 (**Stone-Weierstrass for paths**)**.** Let C be a compact set of continuous paths of bounded variation and let $f : C \to \mathbb{R}$ be a continuous function in the 1-variation topology. Then, for any $\epsilon > 0$ there exists an integer $n \geq 0$ such that for any path $x \in C$

$$\left| f(x) - \sum_{k=0}^{n} \sum_{\substack{w=e_{i_1}\otimes...\otimes e_{i_k} \\ (i_1,...,i_k)\in\{1,...,d\}^k}} \alpha_w S(x)_I^w \right| < \epsilon, \tag{1.8}$$

where $\alpha_w \in \mathbb{R}$ are scalar coefficients and where $S(x)_I^w$ denotes the term in front of the coordinate w in the signature of x over I.

See for example [3, Thm. 2.3.5] for a formal proof. For any path x of finite p-variation ($p > 1$) the terms in the signature *decay factorially* according to the following uniform estimate [4, Lemma 5.1]

$$\left\| \int_{s<u_1<\ldots<u_k<t} \ldots \int dx_{u_1} \otimes \ldots \otimes dx_{u_k} \right\|_{V^{\otimes k}} \leq \frac{\|x\|^k_{p,[s,t]}}{k!}, \tag{1.9}$$

where $\|\cdot\|_{V^{\otimes k}}$ denotes any norm on $V^{\otimes k}$ and $\|x\|_{p,[s,t]}$ denotes the p-variation of the path x restricted to the interval $[s,t]$. Therefore, the collection of iterated integrals in the signature is *graded*. This grading allows one to *truncate* the signature at a finite level $N \in \mathbb{N}$ and consider only a finite collection of integrals as features extracted from x

$$S^N(x)_{[s,t]} = \left(1, \int_{s<u_1<t} dx_{u_1}, \ldots, \int_{s<u_1<\ldots<u_N<t} \ldots \int dx_{u_1} \otimes \ldots \otimes dx_{u_N} \right) \in T^N(V). \tag{1.10}$$

The signature satisfies another important algebraic relation.

Lemma 1.0.1.1 (**Chen's identity**)**.** For any two continuous path x, y of bounded variation one has $S(x \star y) = S(x) \otimes S(y)$, where $\star$ denotes path-concatenation.

It's easy to see that the signature of an increment $\Delta x_t \in V$ is equal to its tensor exponential $S(\Delta x)_t = \exp(\Delta x_t)$. Given a piecewise linear path $x = \Delta x_{t_2} \star \ldots \star \Delta x_{t_\ell}$ obtained by concatenating individual increments Δx_{t_k}, the Chen's identity implies that $S(x) = \exp(\Delta x_{t_2}) \otimes \ldots \otimes \exp(\Delta x_{t_\ell})$. A practical consequence of this relation, which is leveraged by many existing software packages for signatures, is that computing the signature (truncated at level N) of a sequence of length ℓ has complexity $\mathcal{O}(\ell d^N)$.

All these properties make the signature an ideal *feature map* for data streams. However, it is clear that the (truncated) signature has an exponential growth in the number of features, limiting its successful direct usage to machine learning applications where the ambient space of the data streams is relatively low dimensional [5, 6, 7, 8, 9, 10, 11].

Kernel methods [12] have been shown to be efficient learning techniques in situations where inputs are non-Euclidean, high-dimensional (not necessarily sequential) and the number of training instances is limited [13] so that deep learning methods cannot be easily deployed. Most kernels used in practice can be computed efficiently without referring back to the corresponding feature map, a mechanism known as *kernel trick*. When the data is sequential, the design of appropriate kernel functions is a notably challenging task [14]. In [1] the authors introduce the *truncated signature kernel* as the inner product in $T^N(V)$ of two truncated signatures (at level N) and propose an efficient algorithm to compute this kernel starting from any "static" kernel on the ambient space of the input paths.

One of the goals of this chapter will be to extend the results in [1] and consider the *(untruncated) signature kernel* as an inner product in the completion $\overline{T(V)}$ of $T(V)$ of two (untruncated) signatures. For this, leveraging a fundamental property of the signature (Theorem 1.1.1), we prove in Section 1.1 that if the two input paths are continuously differentiable then the signature kernel is the solution of a linear, second order, hyperbolic *partial differential equation* (PDE). In Section 1.2 we recognize the connection between the signature kernel PDE and a class of differential equations known in the literature as *Goursat problems* [15]. This PDE represents a kernel trick for the signature kernel and can be efficiently solved numerically leveraging state-of-the-art hyperbolic PDE solvers; we provide ourselves a competitive finite difference explicit scheme. In Section 1.4 we extend the previous analysis to the broader class of *geometric rough paths*, and show that in this case the signature kernel satisfies an integral equation analogous to the aforementioned Goursat PDE. Finally in Section 1.5, we empirically demonstrate the effectiveness of the signature kernel on various data science applications dealing with sequential data.

We released the python library `sigkernel` implementing our signature PDE kernel and various other functionalities deriving from it. All the experiments presented in this chapter are reproducible following the instructions in `https://github.com/crispitagorico/sigkernel`.

Remark 1.0.2. A concise summary of rough path theory is presented in Appendix A. We note that an efficient algorithm for computing the truncated signature kernel was derived in [1] and then used in [16] in the context of Gaussian processes indexed by time series. Finally, we note that the article [17] first treated the truncated signature kernel in the case of branched rough paths. Integration of two parameters rough integrals is also discussed in [18].

1.1 The signature kernel is the solution of a Goursat PDE

In this section we show that the signature kernel evaluated at two continuously differentiable paths is the solution of a hyperbolic PDE. We denote by $C^1(I, V)$ the space of continuously differentiable paths defined over an interval $I = [u, u']$ and with values on a Banach space V. We will also use the lighter notation $S(x)_t$ to denote the signature of a path x over the interval $[u, t]$, for any $t \in I$.

Definition 1.1.1. Let V be a d-dimensional Banach space with canonical basis $\{e_1, ..., e_d\}$. It is easy to verify that for any $k \geq 1$ the elements

$$\{e_{i_1} \otimes ... \otimes e_{i_k} : (i_1, ..., i_k) \in \{1, ..., d\}^k\} \tag{1.11}$$

form a basis of $V^{\otimes k}$. Consider the inner product on $V^{\otimes k}$ defined on basis elements as

$$\langle e_{i_1} \otimes ... \otimes e_{i_k}, e_{j_1} \otimes ... \otimes e_{j_k} \rangle_{V^{\otimes k}} = \langle e_{i_1}, e_{j_1} \rangle_V ... \langle e_{i_k}, e_{j_k} \rangle_V. \tag{1.12}$$

The inner product $\langle \cdot, \cdot \rangle_{V^{\otimes k}}$ can be extended by linearity to an inner product on $T(V)$ defined for any $A = (a_0, a_1, ...), B = (b_0, b_1, ...)$ in $T(V)$ as

$$\langle A, B \rangle = \sum_{k=0}^{\infty} \langle a_k, b_k \rangle_{V^{\otimes k}}. \tag{1.13}$$

Remark 1.1.1. It is easy to verify that the space $T((V))$ has the following algebraic property that we will refer to as the *coproduct property*. Let $m, n \in \mathbb{N}$ be two positive integers and consider any two elements $A, B \in T((V))$ and any two basis elements $e_{i_1}, e_{i_2} \in V$ seen as elements of $T((V))$, i.e. as $(0, e_{i_1}, 0, ...)$ and $(0, e_{i_2}, 0, ...)$ respectively. Then the following identity holds

$$\langle A \otimes e_{i_1}, B \otimes e_{i_2} \rangle = \langle A, B \rangle \langle e_{i_1}, e_{i_2} \rangle. \tag{1.14}$$

The signature has a fundamental characterisation in terms of *controlled differential equations* (CDEs) (see Chapter 3 for additional details on CDEs). In effect, the signature solves the universal differential equation stated in the next theorem.

Theorem 1.1.1. [2, Lemma 2.10] Let $x : I \to V$ be a continous path of finite p-variation for $p < 2$ and $A = (a_0, a_1, ...) \in T(V)$. Consider the vector field $f : T(V) \to L(V, T(V))$[1]

$$f(A)(v) = A \otimes v = (0, a_0 \otimes v, a_1 \otimes v, ...). \tag{1.15}$$

Then, the unique solution to the following controlled differential equation

$$dS_t = f(S_t)dx_t, \quad S_0 = (1, 0, 0, ...), \tag{1.16}$$

is the signature $S(x)_t$ of the path x. Equation (1.16) can be formally rewritten as

$$dS(x)_t = S(x)_t \otimes dx_t, \quad S(x)_0 = (1, 0, 0, ...), \tag{1.17}$$

which explains why the signature of x can be described as its non-commutative exponential.

[1] $L(V, T(V))$ denotes the space of bounded linear maps from V to $T(V)$.

1.1.1 The signature kernel PDE

Definition 1.1.2 (**Linear signature kernel**). Let $I = [u, u']$ and $J = [v, v']$ be two compact intervals and let $x \in C^1(I, V)$ and $y \in C^1(J, V)$. The (linear) signature kernel $k_{x,y} : I \times J \to \mathbb{R}$ is defined as

$$k_{x,y}(s,t) = \langle S(x)_s, S(y)_t \rangle. \tag{1.18}$$

The following is the main result of this section; it unveils a simple relation between the signature kernel and a class of hyperbolic PDEs.

Theorem 1.1.2. Let $I = [u, u']$ and $J = [v, v']$ be two compact intervals and let $x \in C^1(I, V)$ and $y \in C^1(J, V)$. The signature kernel $k_{x,y}$ is a solution of the following linear, second order, hyperbolic PDE

$$\frac{\partial^2 k_{x,y}}{\partial s \partial t} = \langle \dot{x}_s, \dot{y}_t \rangle_V k_{x,y}, \quad k_{x,y}(u, \cdot) = k_{x,y}(\cdot, v) = 1, \tag{1.19}$$

where $\dot{x}_s = \frac{dx_p}{dp}\big|_{p=s}$, $\dot{y}_t = \frac{dx_q}{dq}\big|_{q=t}$ are the derivatives of x and y at time s and t respectively.

Proof. Clearly, for any $t \in J$ one has

$$\begin{aligned} k_{x,y}(u,t) &= \Big\langle S(x)_{[u,u]}, S(y)_{[v,t]} \Big\rangle \\ &= \Big\langle (1,0,...), S(y)_{[v,t]} \Big\rangle \\ &= 1 \end{aligned}$$

and similarly $k_{x,y}(s, v) = 1$ for any $s \in I$. Recall that the signature of a path $x : I \to V$ satisfies Equation (1.17), which is equivalent to the following integral equation

$$S(x)_s = \mathbf{1} + \int_{p=u}^{s} S(x)_p \otimes dx_p,$$

where $\mathbf{1} = (1, 0, 0, ...)$. Similarly for $S(y)_t$. Hence, we can compute

$$\begin{aligned} k_{x,y}(s,t) &= \langle S(x)_s, S(y)_t \rangle \\ &= \Big\langle \mathbf{1} + \int_{p=u}^{s} S(x)_p \otimes dx_p,\ \mathbf{1} + \int_{q=v}^{t} S(y)_q \otimes dy_q \Big\rangle && \text{(Theorem 1.1.1)} \\ &= 1 + \Big\langle \int_{p=u}^{s} S(x)_p \otimes \dot{x}_p dp,\ \int_{q=v}^{t} S(y)_q \otimes \dot{y}_q dq \Big\rangle && \text{(differentiability)} \\ &= 1 + \int_{p=u}^{s} \int_{q=v}^{t} \langle S(x)_p \otimes \dot{x}_p,\ S(y)_q \otimes \dot{y}_q \rangle dp dq && \text{(linearity)} \\ &= 1 + \int_{p=u}^{s} \int_{q=v}^{t} \langle S(x)_p, S(y)_q \rangle \langle \dot{x}_p, \dot{y}_q \rangle_V dp dq && \text{(coproduct property (1.14))} \\ &= 1 + \int_{p=u}^{s} \int_{q=v}^{t} k_{x,y}(p,q) \langle \dot{x}_p, \dot{y}_q \rangle_V dp dq. && \text{(definition of } k_{x,y}) \end{aligned} \tag{1.20}$$

Note that the inner product and the double integral can be interchanged in Equation (1.20) because of the factorial decay (Equation (1.9)) of the terms in the signature. By the *fundamental theorem of calculus* we can differentiate firstly with respect to s

$$\frac{\partial k_{x,y}(s,t)}{\partial s} = \int_{q=v}^{t} k_{x,y}(s,q)\langle \dot{x}_s, \dot{y}_q \rangle_V dq$$

and then with respect to t to obtain the PDE (1.19)

$$\frac{\partial^2 k_{x,y}(s,t)}{\partial s \partial t} = \langle \dot{x}_s, \dot{y}_t \rangle_V k_{x,y}(s,t).$$

□

Remark 1.1.2. In Theorem 1.1.2 we have assumed the two input paths x, y to be of class C^1. However, one can lower this regularity assumption and consider two continuous paths x, y of bounded variation and obtain the following integral equation

$$k_{x,y}(s,t) = 1 + \int_{p=u}^{s}\int_{q=v}^{t} \langle S(x)_p, S(y)_q \rangle \langle dx_p, dx_q \rangle_V, \tag{1.21}$$

where $\langle dx_p, dx_q \rangle_V$ is a quantity we are going to give rigorous meaning to in Section 1.4, when we will consider the broader class of *geometric rough paths* as input signals to the kernel, which will cover in particular the case of paths of bounded variation.

Sequential information often arrives in the form of complex data streams taking their values in non-trivial *ambient spaces* (for example a video is a stream of images). A good learning strategy would be to first *lift* the underlying ambient space to a (possibly infinite dimensional) *feature space* by means of a *feature map* on static data (RBF, Matern etc. See Table 2.1 in Chapter 2 for a definition of these kernels), and then consider the signature kernel of the lifted paths as the final object [1]. A question that might arise is whether one can compute the signature PDE kernel of the lifted paths from the static kernel on the original ambient space. [1] propose an algorithm to perform this procedure. Next, we provide an abstract explanation of their procedure in the language of Banach spaces and PDEs.

1.1.2 The signature kernel from static kernels on the ambient space

A kernel can be identified with a pair of embeddings of a set $\mathcal{X}$ into a Banach space E and its topological dual E^*; we denote this pair of maps by $\phi : \mathcal{X} \to E$ and $\psi : \mathcal{X} \to E^*$. A kernel induces a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ through the natural pairing between a Banach space and its dual

$$\kappa(a,b) := (\phi(a), \psi(b))_E, \quad \text{for all } a, b \in \mathcal{X}. \tag{1.22}$$

Commonly, E is assumed to be a Hilbert space, in which case ψ can be taken to be the composition $e \circ \phi$ where $e : E \to E^*$ is the canonical isomorphism coming from the *Riesz*

representation theorem, yielding $\kappa(a,b) = \langle \phi(a), \phi(b) \rangle_E$. It is unnecessary however for the general picture for E to be a Hilbert space. In the general framework, a given pair of paths $x : I \to \mathcal{X}$ and $y : J \to \mathcal{X}$, with $I = [u, u'], J = [v, v']$, can be lifted to paths on E and E^* respectively as follows

$$X_s = \phi(x_s), \quad Y_t = \psi(y_t), \quad \text{for all } s \in I, t \in J. \tag{1.23}$$

If we assume that X and Y are continuous and have bounded variation, then their signatures are well defined as elements of $T(E)$, which is again a Banach space with $T^N(E)^* \cong T^N(E^*)$ for any $N \geq 1$ [2]. Hence, starting with a kernel κ on $\mathcal{X}$, the (non-linear) signature kernel is defined as follows

$$k_{X,Y}(s,t) = (S(\phi \circ x)_s, S(\psi \circ y)_t)_{T(E)} = (S(X)_s, S(Y)_t)_{T(E)}. \tag{1.24}$$

If furthermore we assume that the lifted paths X, Y are of class C^1, Theorem 1.1.2 applies, yielding the following PDE

$$\frac{\partial^2 k_{X,Y}}{\partial s \partial t} = (\dot{X}_s, \dot{Y}_t)_E \; k_{X,Y}, \quad k_{X,Y}(u, \cdot) = k_{X,Y}(\cdot, v) = 1. \tag{1.25}$$

With a first order finite difference approximations for the derivatives $\dot{X}_s, \dot{Y}_t$, the PDE can be entirely expressed in terms of the static kernel κ and underlying paths x, y

$$\begin{aligned} \frac{\partial^2 k_{X,Y}}{\partial s \partial t} &\approx ((X_s, Y_t)_E - (X_{s-1}, Y_t)_E - (X_s, Y_{t-1})_E + (X_{s-1}, Y_{t-1})_E) k_{X,Y} \\ &= (\kappa(x_s, y_t) - \kappa(x_{s-1}, x_t) - \kappa(x_s, y_{t-1}) + \kappa(x_{s-1}, y_{t-1})) k_{X,Y}. \end{aligned} \tag{1.26}$$

Remark 1.1.3. We note that it is possible to establish our Goursat PDE (1.19) from the results [19, Proposition 4.7 p. 16], [19, Theorem 4., Appendix A], but this requires additional arguments as we shall explain next. Using their notation, given two differentiable paths σ, τ, the truncated (at level M) signature kernel $k^{\oplus}_{\leq M}(\sigma, \tau)$ satisfies the following equation

$$k^{\oplus}_{\leq M}(\sigma,\tau)_{u,v} = 1 + \iint\limits_{(s_1,t_1)\in(0,u)\times(0,v)} \Big(1 + \ldots \iint\limits_{(s_M,t_M)\in(0,s_{M-1})\times(0,t_{M-1})} d\kappa_{\sigma,\tau}(s_M,t_M)\Big) d\kappa_{\sigma,\tau}(s_1,t_1),$$

where $d\kappa_{\sigma,\tau}(s,t) = k(\dot{x}_s, \dot{y}_t) ds dt$ and where k is a kernel on the ambient space of the paths σ, τ. The first step towards a PDE is to realise that the first integrand in the last equation is itself the truncated signature kernel, truncated at level $M-1$, yielding the expression

$$k^{\oplus}_{\leq M}(\sigma,\tau)_{u,v} = 1 + \iint\limits_{(s_1,t_1)\in(0,u)\times(0,v)} k^{\oplus}_{\leq M-1}(\sigma,\tau)_{s_1,t_1} d\kappa_{\sigma,\tau}(s_1,t_1). \tag{1.27}$$

The untruncated signature kernel is obtained by taking the limit in Equation (1.27) when $M \to \infty$. The factorial decay of the terms of the signature yields uniform convergence of

this limiting process. Two uniformly convergent sequences of functions that are equal for all finite levels M they are also equal in the limit, which implies

$$k^{\oplus}_{\leq\infty}(\sigma,\tau)_{u,v} = 1 + \iint\limits_{(s_1,t_1)\in(0,u)\times(0,v)} k^{\oplus}_{\leq\infty}(\sigma,\tau)_{s_1,t_1} d\kappa_{\sigma,\tau}(s_1,t_1). \tag{1.28}$$

Finally, one substitutes $d\kappa_{\sigma,\tau}(s,t) = k(\dot{x}_s, \dot{y}_t)dsdt$. At that point (and similarly to our argument in Theorem 2.5) one can differentiate both sides of Equation (1.28) to get the Goursat PDE.

In the next section we recognize the link between Equation (1.19) and a class of differential equations known in the literature as Goursat problems and propose a numerical solver for our specific PDE.

1.2 A Goursat problem

Equation (1.19) is an instance of a Goursat problem, which is a class of hyperbolic PDEs introduced in [15]. The PDE (1.19) is defined on the bounded domain

$$\mathcal{D} := \{(s,t) \mid u \leq s \leq u', v \leq t \leq v'\} \subset I \times J \tag{1.29}$$

and its existence and uniqueness (for paths of class C^1) are guaranteed by setting the functions $C_1 = C_2 = C_4 = 0$ and $C_3(s,t) = \langle \dot{x}_s, \dot{y}_t \rangle_V$ in the following result.

Theorem 1.2.1. [20, Theorems 2 & 4] Let $\sigma : I \to \mathbb{R}$ and $\tau : J \to \mathbb{R}$ be two absolutely continuous functions whose first derivatives are square integrable and such that $\sigma(u) = \tau(v)$. Let $C_1, C_2, C_3 : \mathcal{D} \to \mathbb{R}$ be bounded and measurable over $\mathcal{D}$ and $C_4 : \mathcal{D} \to \mathbb{R}$ be square integrable. Then there exists a unique function $z : \mathcal{D} \to \mathbb{R}$ such that $z(s,v) = \sigma(s), z(u,t) = \tau(t)$ and (almost everywhere on $\mathcal{D}$)

$$\frac{\partial^2 z}{\partial s \partial t} = C_1(s,t)\frac{\partial z}{\partial s} + C_2(s,t)\frac{\partial z}{\partial t} + C_3(s,t)z + C_4(s,t). \tag{1.30}$$

If in addition $C_i \in C^{p-1}(\mathcal{D})$ $(i = 1,2,3,4)$ and σ and τ are C^p, then the unique solution $z : \mathcal{D} \to \mathbb{R}$ of the Goursat problem is of class C^p.

In the case of the signature PDE kernel, if the two input paths x, y are of class C^p, then their derivatives will be of class C^{p-1}, and therefore Theorem 1.2.1 implies that the solution $k_{x,y}$ of the PDE (1.19) will be of class C^p.

1.2.1 Finite difference approximation

In this section, we propose a numerical method based on an explicit finite difference scheme to approximate the solution of the Goursat PDE (1.19). To simplify the notation, we consider the case where $V = \mathbb{R}^d$. If x and y are piecewise linear, then the PDE (1.19) becomes

$$\frac{\partial^2 k_{x,y}}{\partial s \partial t} = C_3 k_{x,y}\,, \tag{1.31}$$

on each sub-domain $\mathcal{D}_{ij} = \{(s,t) \mid u_i \le s \le u_{i+1}, v_j \le t \le v_{j+1}\}$ where $C_3 = \langle \dot{x}_s, \dot{y}_t \rangle_V$ is constant. In integral form, the PDE (1.31) can be written as

$$k_{x,y}(s,t) = k_{x,y}(s,v) + k_{x,y}(u,t) - k_{x,y}(u,v) + C_3 \int_u^s \int_v^t k_{x,y}(r,w)\, dr\, dw, \tag{1.32}$$

for $(s,t),(u,v) \in \mathcal{D}_{ij}$ with $u \le s$ and $v \le t$. By approximating the double integral in Equation (1.32), we can derive the following numerical explicit scheme

$$\begin{aligned} k_{x,y}(s,t) \approx\ & k_{x,y}(s,v) + k_{x,y}(u,t) - k_{x,y}(u,v) \\ & + \frac{1}{2} C_3 \big(k_{x,y}(s,v) + k_{x,y}(u,t)\big)(u-s)(t-v). \end{aligned} \tag{1.33}$$

Remark 1.2.1. An implicit scheme can be obtained by estimating Equation (1.32) with all four values of $k_{x,y}$ as follows

$$\begin{aligned} k_{x,y}(s,t) \approx\ & k_{x,y}(s,v) + k_{x,y}(u,t) - k_{x,y}(u,v) \\ & + \frac{1}{4} C_3 \big(k_{x,y}(u,v) + k_{x,y}(s,v) + k_{x,y}(u,t) + k_{x,y}(s,t)\big)(u-s)(t-v). \end{aligned} \tag{1.34}$$

As one might expect, more sophisticated approximations can be derived by applying higher order quadrature methods to the double integral in (1.32) (see [21, 22] for specific examples).

Let $\mathcal{D}_I = \{u = u_0 < u_1 < ... < u_{m-1} < u_m = u'\}$ be a partition of the interval I and $\mathcal{D}_J = \{v = v_0 < v_1 < ... < v_{n-1} < v_n = v'\}$ be a partition of the interval J. Using the above, we can define *finite difference schemes* on the grid $P_0 := \mathcal{D}_I \times \mathcal{D}_J$ (and its dyadic refinements).

Definition 1.2.1. For $\lambda \in \{0, 1, 2, \cdots\}$, we define the grid P_λ as the dyadic refinement of P_0 such that $P_\lambda \cap ([u_i, u_{i+1}] \times [v_i, v_{i+1}]) = \{u_i + k\, 2^{-\lambda}(u_{i+1} - u_i), v_j + l\, 2^{-\lambda}(v_{j+1} - v_j)\}_{0 \le k,l \le 2^\lambda}$.

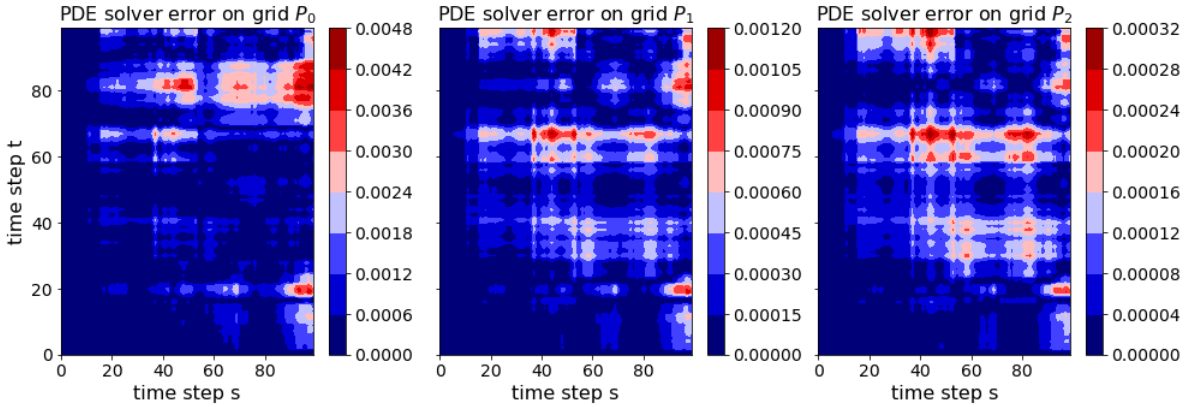


Figure 1.2: Example of error distribution of $k_{x,y}(s,t)$ on the grids P_0, P_1, P_2. The discretization is roughly four times more accurate on P_1 than on P_0 (as expected by Theorem 1.2.2).

On the grid $P_\lambda = \{(s_i, t_j)\}_{0 \le i \le 2^\lambda n,\, 0 \le j \le 2^\lambda m}$, we define the following explicit finite difference scheme for the PDE (1.19)

$$\begin{aligned} \hat{k}(s_{i+1}, t_{j+1}) &= \hat{k}(s_{i+1}, t_j) + \hat{k}(s_i, t_{j+1}) - \hat{k}(s_i, t_j) \\ &\quad + \frac{1}{2}\langle x_{s_{i+1}} - x_{s_i}, y_{t_{j+1}} - y_{t_j}\rangle (\hat{k}(s_{i+1}, t_j) + \hat{k}(s_i, t_{j+1})), \\ \hat{k}(s_0, \cdot\,) &= \hat{k}(\,\cdot, t_0) = 1, \end{aligned} \tag{1.35}$$

Remark 1.2.2. If x and y are piecewise linear paths with respect to the coarsest grid P_0 then $\langle x_{s_{i+1}} - x_{s_i}, y_{t_{j+1}} - y_{t_j}\rangle = \frac{1}{2^{2\lambda}}\langle x_{u_{p+1}} - x_{u_p}, y_{v_{q+1}} - y_{v_q}\rangle$ for some $0 \le p < n$ and $0 \le q < m$.

The explicit finite differences scheme (1.35) has a time complexity of $O(d^2\, 2^{2\lambda}\, mn)$ on the grid P_λ, where d is the dimension of the input streams x, y and m, n denote their respective lengths. Theorem 1.2.2 (which is the proved in [23, Appendix]) ensures that by refining the discretization of the grid used to approximate the PDE, we get convergence to the true value. In practice we found that provided the input paths are rescaled so that their maximum value across all times and all dimensions is not too large (≈ 1), coarse partitioning choices such as P_0 or P_1 are sufficient to obtain a highly accurate approximation, as shown in Figure 1.2.

Theorem 1.2.2. Let $\widetilde{k}$ be a numerical solution obtained by applying the proposed finite difference scheme (1.35) to the Goursat PDE (1.19) on P_λ where x and y are piecewise linear with respect to the grids $\mathcal{D}_I$ and $\mathcal{D}_J$. In particular, we are assuming there exists a constant M, that is independent of λ, such that

$$\sup_{\mathcal{D}} |\langle \dot{x}_s, \dot{y}_t\rangle| < M. \tag{1.36}$$

Then there exists a constant $K > 0$ depending on M and $k_{x,y}$, but independent of λ, such that

$$\sup_{\mathcal{D}} \left| k_{x,y}(s,t) - \widetilde{k}(s,t) \right| \leq \frac{K}{2^{2\lambda}}, \quad \text{for all } \lambda \geq 0. \tag{1.37}$$

1.2.2 GPU implementation of the Goursat PDE

As mentioned earlier, the time complexity for one signature PDE kernel evaluation is $\mathcal{O}(d\ell^2)$ on P_0, where d is the number of channels of the input time series and ℓ is their (maximum) length. Therefore, the complexity is quadratic in the length of the time series, which makes kernel evaluations computationally expensive for long time series. This also holds for the algorithm proposed in [1]. However, it is possible to parallelize the PDE solver by observing that instead of solving the PDE in row or column order, we can update the antidiagonals of the solution grid: each cell on an antidiagonal can be updated in parallel as there is no data dependency between them. This breaks the quadratic complexity, that becomes linear in the length ℓ, provided the number of threads in the `GPU` exceeds the size of the discretization. as shown in Figure 1.3. This parallelization is possible thanks to the "PDE structure" of the problem, representing a considerable computational gain of our algorithm compared to the one proposed by [1]. We also note that the linear dependency on the number of channels d of the input time series allows for the evaluation of the signature PDE kernel on time series with thousands of channels. Our library `sigkernel` offers the ability to evaluate kernels on a `CPU` using an optimized `cython` implementation as well as on `CUDA` if `GPU`s are available to the user.

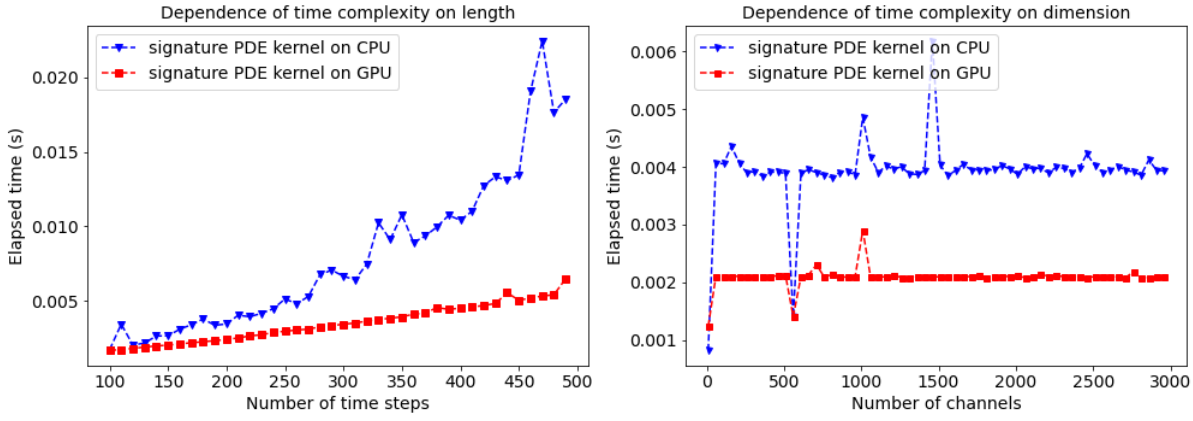


Figure 1.3: Comparison of the elapsed time (s) to reach an accuracy of 10^{-3} from a target value obtained by solving the signature kernel PDE on a fine discretization grid (P_5). We simulate $N = 5$ (piecewise linear interpolation of) Brownian paths at each run. On the **left** is the dependency on the length of two paths of dimension $d = 2$. Note the complexity reduction from quadratic on `CPU` to linear on `GPU` (P100). On the **right** is the dependency on the dimension of two paths of length $\ell = 10$.

In Section 1.5 we will present various applications of the signature kernel to time series classification and regression problems. But first we continue our theoretical analysis and drop the smoothness assumption on the input paths x, y and extending the definition of signature kernel to less regular classes of paths, namely geometric rough paths. The need to investigate the rough version of the signature kernel can be motivated also from several practical viewpoints. For example, this kernel can be used to derive an (unbiased) estimator for the *maximum mean discrepancy* (MMD) distance between distributions on path-space [17, sections 7, 8]. The MMD distance itself is useful to train models such as *neural SDEs* [24, 25, 26, 27], that is to fit neural SDEs to time series data. Since SDE strong solutions are geometric p-rough paths, Theorem 1.4.1 provides a candidate for the limiting kernel as the mesh size of the SDE discretization tends to zero. In particular, it guarantees that the signature kernel doesn't "blow-up" in the limit. Another area where the rough signature kernel could be relevant is quantitative finance, where *rough volatility models* [28, 29] try to calibrate differential equation driven by *fractional Brownian motion*, which is a rough path if the *Hurst exponent* $h \leq 2$.

1.3 Differentiable computations with the signature kernel

Next we introduce the notion of *reproducing kernel Hilbert space* (RKHS) associated to the signature kernel.

Definition 1.3.1 (**Signature RKHS**)**.** Let $\mathcal{X}$ be a compact set of continuous paths of bounded variation $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be the signature kernel. Let $\mathcal{H}_0 = \text{Span}\{k(\cdot, x) : x \in \mathcal{X}\}$ be a set of functions endowed with the following inner product

$$\langle f, g \rangle_{\mathcal{H}_0} = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j k(x_i, y_j)$$

where $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$ and $g(\cdot) = \sum_{j=1}^{n} \beta_j k(\cdot, y_j)$. Define the RKHS $\mathcal{H}$ as the set of functions $f : \mathcal{X} \to \mathbb{R}$ for which there exists a Cauchy sequence $\{f_n\}$ of functions in $\mathcal{H}_0$ converging pointwise to f. For any $f, g \in \mathcal{H}$ let $\{f_n\}$ and $\{g_n\}$ be two $\mathcal{H}_0$-Cauchy sequences converging pointwise to f and g respectively; we endow $\mathcal{H}$ with the following inner product:

$$\langle f, g \rangle_{\mathcal{H}} = \lim_{n \to \infty} \langle f_n, g_n \rangle_{\mathcal{H}_0}$$

Remark 1.3.1. The fact that the above limit exists is the content of [30, Lemma 5]

1.3.1 A distance between probability measures on paths

Definition 1.3.2 (**MMD distance**). Let $\mathcal{X}$ be a compact set of continuous paths of bounded variation with values on a Banach space V and consider two probability measures μ, ν supported on $\mathcal{X}$. The Maximum Mean Discrepancy (MMD) distance between μ, ν is defined as follows

$$d_{\mathcal{G}}(\mu, \nu) = \sup_{f \in \mathcal{G}} |\mathbb{E}_{x \sim \mu}[f(x)] - \mathbb{E}_{y \sim \nu}[f(y)]|, \tag{1.38}$$

where $\mathcal{G}$ is the following RKHS-unit-ball

$$\mathcal{G} = \{f : \mathcal{X} \to \mathbb{R} : ||f||_{\mathcal{H}} \leq 1\}, \tag{1.39}$$

where $\mathcal{H}$ is the RKHS associated to the signature kernel and $||f||_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$.

For simplicity of notation, in what follows we will denote the signature kernel of two paths $x, y : [0, T] \to V$ up to the pair of times (T, T) by $k(x, y) := k_{x,y}(T, T)$.

Following the findings in [31] and [17] one has the following expression for the MMD distance

$$\begin{aligned} d_{\mathcal{G}}(\mu, \nu)^2 &= \|\mathbb{E}S(\mu) - \mathbb{E}S(\nu)\|^2_{T(V)} && (1.40) \\ &= \mathbb{E}_{(x,x') \sim \mu \times \nu}[k(x, x')] - 2\mathbb{E}_{(x,y) \sim \mu \times \nu}[k(x, y)] + \mathbb{E}_{(y,y') \sim \nu \times \nu}[k(y, y')], && (1.41) \end{aligned}$$

where $\mathbb{E}S$ denotes the expected signature, which will be extensively discussed in the next chapter.

Given $m \in \mathbb{N}$ sample paths $\{x^i\}_{i=1}^m \sim \mu$ and $n \in \mathbb{N}$ sample paths $\{y^j\}_{j=1}^n \sim \nu$, an unbiased estimator of the MMD distance is given by [31]

$$d_{\mathcal{G}}(\mu, \nu)^2 \approx \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k(x^i, x^j) - \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k(x^i, y^j) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(y^i, y^j). \tag{1.42}$$

Remark 1.3.2 (**A two-sample hypothesis test**). From the above we can easily construct a two-sample hypothesis test procedure to test the null hypothesis

$$H_0 : \mu = \nu \text{ against the alternative } H_1 : \mu \neq \nu$$

In effect, a natural test statistic is given precisely by the MMD distance $d_{\mathcal{G}}(\mu, \nu)$ (see [17, section 8] for further details).

In the context of generative models for time series, one can combine a parametric generator, usually a seq2seq-type neural network G_θ, with the MMD distance above to discriminate. More precisely, let μ be an unknown target probability measure on path space and ν be some reference measure from which we can draw sample paths (for example the Wiener measure),

and let G_θ be a parametric seq2seq-type model transforming sample paths from the support of μ to sample paths from the support of ν. Training on a dataset of input-output pairs of paths allows one to find the best set of parameters θ such that by transporting μ into the pushforward measure $G_\theta \# \nu$, the latter is a good approximation of μ with respect to the MMD metric.

In order to optimize the MMD distance with respect to the parameters $\boldsymbol{\theta}$ of a parametric model G_θ, one is required to take derivatives of the signature kernel with respect to each of its input paths. Differentiating through the operations of the "forward pass" can be done via the chain rule and by leveraging the automatic differentiation capabilities of modern deep learning libraries (Tensorflow, PyTorch etc.). However, backpropagating through the operations of the PDE solver used to obtain the signature kernel can be inefficient.

In the next section we show how to compute such gradients without backpropagating through the operations of the PDE solver. The ability not to store all the intermediate values of $k(G_\theta(x), G_\theta(y))$ on the discretization grid of the PDE solver allows one to train generative models efficiently both in the terms of time complexity and memory cost. We treat the PDE solver as a "black box" and show that the gradients of the signature kernel are also the solutions of a second hyperbolic PDE.

We now introduce the concept of "directional derivative" along a path.

1.3.2 Differentiating along the direction of a path

Consider a time series $\mathbf{x}$ as a collection of points $x_i \in \mathbb{R}^d$ with corresponding time-stamps $s_i \in \mathbb{R}$ such that

$$\mathbf{x} = ((s_0, x_0), (s_1, x_1), ..., (s_\ell, x_\ell)) \tag{1.43}$$

with $s_0 < ... < s_\ell$. Every vector x_i in the sequence can be written with respect to the canonical basis of $\mathbb{R}^d$ as

$$x_i = \sum_{j=1}^{d} x_{i,j} \mathbf{e}_j. \tag{1.44}$$

Let $X : [0, T] \to \mathbb{R}^d$ be the piecewise linear interpolation of the data such that $X_{t_i} = (t_i, x_i)$. Similarly for a second time series $\mathbf{y}$ and resulting piecewise linear interpolation Y.

To formulate a backpropagation algorithm in a rigorous way compatible with automatic differentiation libraries (in this work we use PyTorch), we need to give meaning to the following gradients

$$\left\{ \frac{\partial}{\partial x_{i,j}} k(X, Y) \right\}_{i,j=1}^{\ell,d}. \tag{1.45}$$

The technical difficulty here consists in reconciling the continuous nature of the input path X and the discrete nature of the locations $x_{i,j}$ where one wants to compute the gradients, given by the knots of the time series $\mathbf{x}$.

Next we introduce a collection of *localised impulses* and define the concept of *directional derivative of the signature kernel along a path* in order to make sense of the gradients in Equation (1.45). These definitions will be followed by the main result of this section, namely that the directional derivative of k solves another PDE similar to Equation (1.19) for the signature kernel, for which we derive an explicit solution via the *technique of variation of parameters* (Theorem 1.3.1).

Definition 1.3.3. For any $i = 1, \dots, \ell$ and any $j = 1, \dots, d$ define the localised impulse $\gamma_{i,j} : [0, T] \to \mathbb{R}^d$ as the solution of the following ODE

$$\dot{\gamma}_{i,j} = \frac{1}{\ell} e_j \mathbf{1}_{\{t \in [(i-1)/\ell, i/\ell)]\}}, \quad \gamma_{i,j}(0) = 0. \tag{1.46}$$

Definition 1.3.4. For any path $\gamma \in \mathcal{X}$ the directional derivative of the signature kernel k along γ is defined as

$$k_\gamma(X, Y) := \frac{\partial}{\partial \epsilon} k\Big(X + \epsilon\gamma, Y\Big)\Big|_{\epsilon=0}. \tag{1.47}$$

Each gradient of the signature kernel at the knot $x_{i,j}$ reported in Equation (1.45) can be identified with the directional derivative of k along the localised impulse $\gamma_{i,j}$ of Definition 1.3.3

$$\frac{\partial}{\partial x_{i,j}} k(X, Y) := k_{\gamma_{i,j}}(X, Y). \tag{1.48}$$

1.3.3 A second PDE for the gradients of the signature kernel

Recall that the signature kernel solves the following PDE

$$\frac{\partial^2 U}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t) U. \tag{1.49}$$

Integrating both sides with respect to s and t one obtains

$$U(s, t) = 1 + \int_{u=0}^{s} \int_{v=0}^{t} U(u, v)(\dot{X}_u^T \dot{Y}_v) du dv. \tag{1.50}$$

Let's denote by $U_\gamma : [0, T] \times [0, T] \to \mathbb{R}$ the directional derivative k_γ evaluated at the restricted paths $X|_{[0,s]}, Y|_{[0,t]}$, i.e.

$$U_\gamma(s, t) := k_\gamma(X|_{[0,s]}, Y|_{[0,t]}). \tag{1.51}$$

The combination of Equations (1.50) and (1.51) yields the relation

$$\begin{aligned} U_\gamma(s,t) &= \frac{\partial}{\partial \epsilon} k\Big((X+\epsilon\gamma)|_{[0,s]}, Y|_{[0,t]}\Big)\Big|_{\epsilon=0} \\ &= \frac{\partial}{\partial \epsilon}\left(\int_0^s \int_0^t U(u,v)\left(\dot{X}_u + \epsilon\dot{\gamma}_u\right)^T \dot{Y}_v du dv\right)_{\epsilon=0} \\ &= \int_0^s \int_0^t \left(U_\gamma(u,v)\dot{X}_u^T\dot{Y}_v + U(u,v)\dot{\gamma}_u^T\dot{Y}_v\right) du dv. \end{aligned}$$

Hence, differentiating the last equation first with respect to t and then s we get that the directional derivative k_γ of the signature kernel along the path γ solves the following PDE

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = (\dot{X}_s^T\dot{Y}_t)U_\gamma + (\dot{\gamma}_s^T\dot{Y}_t)U \tag{1.52}$$

with boundary conditions

$$U_\gamma(0,\cdot) = 0, \quad U_\gamma(\cdot,0) = 0. \tag{1.53}$$

As a result, the gradients in Equation (1.45) can be computed in a single call to a PDE solver, which concatenates the original state and the partial derivatives (1.52) into a single vector. Each partial derivative follows the dynamics of (1.52) where one replaces the direction γ by the relevant localised impulse $\gamma_{i,j}$, $\tau_{i,j}$ for X and Y respectively.

1.3.4 An explicit solution by variation of parameters

From this second PDE (1.52) we derive the following theorem that allows to compute the directional derivative k_γ of the signature kernel directly from its evaluations at X, Y and at $\overleftarrow{X}, \overleftarrow{Y}$, where $\overleftarrow{X}, \overleftarrow{Y}$ are respectively the paths X, Y reversed in time.

Theorem 1.3.1. For any $\gamma \in \mathcal{X}$ the directional derivative $k_\gamma(X,Y)$ of the signature kernel along the path γ satisfies the following relation

$$k_\gamma(X,Y) = \int_0^T \int_0^T U(s,t)\widetilde{U}(T-s,T-t)(\dot{\gamma}_s^T\dot{Y}_t) ds dt$$

where $\widetilde{U}(s,t) = k(\overleftarrow{X}|_{[0,s]}, \overleftarrow{Y}|_{[0,t]})$.

Before proving Theorem 1.3.1 we need the following lemma.

Lemma 1.3.1.1. For any two continuous paths of bounded variation $X, Y \in \mathcal{X}$ the signature kernel satisfies the following relation

$$k(X,Y) = k(\overleftarrow{X}, \overleftarrow{Y}), \tag{1.54}$$

where $\overleftarrow{X}, \overleftarrow{Y}$ are the paths X, Y respectively reversed in time.

Proof. It is a standard result in rough path theory (see for example [2]) that $S(\overleftarrow{X}) = S(X)^{-1}$, where the inverse is taken in the set of grouplike elements, which is a group. The operator on grouplike elements $g : S(X) \mapsto S(X)^{-1}$ reverses the order of the letters in each word and multiplies the result by -1 if the length of the word is odd. Expanding out $k(\overleftarrow{X}, \overleftarrow{Y})$ coordinate-wise it is easy to see that the two -1's for words of odd length cancel as multiplied together, therefore the expansion of $k(X, Y)$ matches the one of $k(\overleftarrow{X}, \overleftarrow{Y})$. □

Recall the notation for the signature kernel and its directional derivative used in the statement of Theorem 1.3.1:

$$U(s,t) := k\left(X|_{[0,s]}, Y|_{[0,t]}\right),$$
$$U_\gamma(s,t) := k_\gamma\left(X|_{[0,s]}, Y|_{[0,t]}\right).$$

Proof of Theorem 1.3.1. Let $\gamma : [0,T] \to \mathbb{R}^d$ be a continuous path of bounded variation along which we wish to differentiate k. Let's assume that for any $s,t \in [0,T]$ there exists a function $A_{s,t} : [0,T] \times [0,T] \to \mathbb{R}$ such that

$$U_\gamma(s,t) = \int_0^s \int_0^t A_{s,t}(u,v) U(u,v)(\dot{\gamma}_u^T \dot{Y}_v) du dv. \tag{1.55}$$

Differentiating U_γ with respect to s and t we get

$$\begin{aligned}\frac{\partial^2 U_\gamma}{\partial s \partial t} &= \int_0^s \int_0^t \frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} U(u,v)(\dot{\gamma}_u^T \dot{Y}_v) du dv \\ &\quad + A_{s,t}(s,t) U(s,t)(\dot{\gamma}_s^T \dot{Y}_t).\end{aligned} \tag{1.56}$$

By Equation (1.52) in the main paper we know that the directional derivative of the signature kernel along the path γ solves the following PDE

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t) U_\gamma(s,t) + (\dot{\gamma}_s^T \dot{Y}_t) U(s,t). \tag{1.57}$$

Equating the right hand sides of Equation (1.56) and Equation (1.57) we deduce that $A_{s,t}(s,t) = 1$ and

$$\begin{aligned}&\int_0^s \int_0^t \frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} U(u,v)(\dot{\gamma}_u^T \dot{Y}_v) du dv \\ &= U_\gamma(s,t)(\dot{X}_s^T \dot{Y}_t) \\ &= (\dot{X}_s^T \dot{Y}_t) \int_0^s \int_0^t A_{s,t}(u,v) U(u,v)(\dot{\gamma}_u^T \dot{Y}_v) du dv,\end{aligned}$$

which implies that

$$\frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t) A_{s,t}(u,v). \tag{1.58}$$

Equivalently, by integrating with respect to s and t

$$A_{s,t}(u,v) = 1 + \int_u^s \int_v^t A_{s',t'}(u,v)(\dot{X}_{s'}^T \dot{Y}_{t'}) ds' dt'. \tag{1.59}$$

Hence

$$A_{T,T}(u,v) = \langle S(X)_{[u,T]}, S(Y)_{[v,T]} \rangle \tag{1.60}$$

$$= k(\overleftarrow{X}|_{[0,T-u]}, \overleftarrow{Y}|_{[0,T-v]}), \tag{1.61}$$

where the last equality is a consequence of Lemma 1.3.1.1. Pluging back this result into Equation (1.55) concludes the proof. □

We now continue our theoretical analysis by dropping the smoothness assumption on the input paths x, y and extending the definition of the signature kernel to less regular classes of paths, namely geometric rough paths. In section 1.4 we will show our second main result, i.e. that the rough version of the signature kernel satisfies a rough integral equation analogous to the Goursat problem presented above.

1.4 The signature kernel for geometric rough paths

Here we extend the notion of signature kernel developed in Section 1.1 to the broader class of geometric rough paths. To follow the material presented in this section we assume that the reader has some level of familiarity with basic concepts of rough path theory. We provide a brief summary of this theory in Appendix A. We begin by clarifying what we mean by signature of a geometric p-rough path.

1.4.1 The signature of a geometric rough path

Definition 1.4.1. The signature $S(X)$ of a geometric p-rough path $X \in G\Omega_p(V)$ (Definition A.2.6) controlled by a control (Definition A.2.2) ω is its unique extension to a multiplicative functional (Definition A.2.1) on $T(V)$ as given by the Extension Theorem (A.2.1).

From now on, we will denote by $G\Omega_p(V)$ the space of geometric p-rough paths over V. Because all the sums in $T(V)$ are finite, $(T(V), \langle \cdot, \cdot \rangle)$ is an inner product space. Hence, denoting by $\overline{T(V)}$ the completion of $T(V)$, $(\overline{T(V)}, \langle \cdot, \cdot \rangle)$ is a Hilbert space. Let $\|\cdot\|$ be the norm on $\overline{T(V)}$ induced by the inner product $\langle \cdot, \cdot \rangle$, i.e. defined for any $A = (a_0, a_1, ...) \in \overline{T(V)}$ as $\|A\| = \sqrt{\sum_{k \geq 0} \|a_k\|^2_{V^{\otimes k}}}$, where $\|\cdot\|_{V^{\otimes k}}$ is the norm on $V^{\otimes k}$ induced by $\langle \cdot, \cdot \rangle_{V^{\otimes k}}$, for any $k \geq 0$. In summary, we have the following chain of inclusions

$$T(V) \hookrightarrow \overline{T(V)} \hookrightarrow T((V)). \tag{1.62}$$

Note that $\overline{T(V)} = \{x \in T((V)) : ||x|| < \infty\}$.

Lemma 1.4.0.1. Let X be a geometric p-rough path $X \in G\Omega_p(V)$ defined over the simplex Δ_T. Then, for any $(s,t) \in \Delta_T$ one has $S(X)_{s,t} \in \overline{T(V)}$.

Proof. To prove the statement of the lemma it suffices to find a sequence of tensors $\{X^{(n)}_{s,t} \in T^k(V)\}_{n\in\mathbb{N}}$ that convergences to $S(X_{s,t})$ in the $\|\cdot\|$-topology. Setting $X^{(n)}_{s,t} = (1, X^1_{s,t}, ..., X^n_{s,t}, 0, ...)$, and using the bounds from the Extension Theorem A.2.1 we have

$$\|S(X)_{s,t}\| = \sqrt{\sum_{k=0}^{\infty} \|X^k_{s,t}\|^2_{V^{\otimes k}}} \leq \sqrt{\sum_{k=0}^{\infty} \frac{\omega(s,t)^{2k/p}}{(\beta_p(k/p)!)^2}} \leq \sum_{k=0}^{\infty} \frac{\omega(s,t)^{k/p}}{\beta_p(k/p)!} \tag{1.63}$$

which is clearly a convergent series because of the terms decay factorially, and $\forall (s,t) \in \Delta_I$

$$\|X^{(n)}_{s,t} - S(X)_{s,t}\| = \sqrt{\sum_{k\geq n+1}^{\infty} \|X^k_{s,t}\|^2_{V^{\otimes k}}} \longrightarrow 0 \text{ as } n \to \infty. \tag{1.64}$$

□

Next we present our second main result, that is we extend the definition of signature kernel to the space of geometric p-rough paths (Definition A.2.6) and show that this rough version of the signature kernel solves an iterated double integral equation of two one-forms analogous to the Goursat PDE (1.19) presented in Section 1.1.

1.4.2 The signature kernel for geometric rough paths

In what follows Δ_I, Δ_J will denote the following two simplices

$$\Delta_I = \{(s,t) \in [i_-, i_+]^2 : i_- \leq s \leq t \leq i_+\}, \tag{1.65}$$

$$\Delta_J = \{(s,t) \in [j_-, j_+]^2 : j_- \leq s \leq t \leq j_+\}, \tag{1.66}$$

where $i_-, i_+, j_-, j_+ \geq 0$ are positive scalars such that $i_- < i_+$ and $j_- < j_+$.

Definition 1.4.2. Let $p, q \geq 1$ be two scalars. Let $X \in G\Omega_p(V)$ and $Y \in G\Omega_q(V)$ be two geometric p- and q-rough paths respectively which are controlled by two controls ω_X and ω_Y respectively. The (rough) signature kernel $K_{(s_1,s_2),(t_1,t_2)} : G\Omega_p(V) \times G\Omega_q(V) \to \mathbb{R}$ is defined for any $(s_1, s_2) \in \Delta_I$ and $(t_1, t_2) \in \Delta_J$ as follows

$$K_{(s_1,s_2),(t_1,t_2)}(X,Y) = \langle S(X)_{s_1,s_2}, S(Y)_{t_1,t_2} \rangle \tag{1.67}$$

where the inner product is taken in $\overline{T(V)}$.

Remark 1.4.1. On the one hand, the signature kernel of Definition 1.1.2 is configured to act on two time indices and is indexed by two paths. This choice was made in order to differentiate with respect to these and obtain the PDE (1.19). On the other hand, the rough signature kernel of Definition 1.4.2 acts on two (rough) paths and is indexed by time indices. When dealing with highly oscillatory objects like rough paths studied in this section, one can't expect to obtain a PDE, as these paths are far from being differentiable (even locally). However, we will nonetheless be able to use a density argument to prove our claim (Theorem 1.4.1).

Next we show that the rough signature kernel is bounded and continuous.

Lemma 1.4.0.2. For any $(X, Y) \in G\Omega_p(V) \times G\Omega_q(V)$ and any $(s_1, s_2) \in \Delta_I, (t_1, t_2) \in \Delta_J$

$$\langle S(X)_{s_1,s_2}, S(Y)_{t_1,t_2} \rangle < +\infty. \tag{1.68}$$

Furthermore, the rough signature kernel $K_{(s_1,s_2),(t_1,t_2)}$ is continuous with respect to the product p, q-variation topology.

Proof. For any $(s_1, s_2) \in \Delta_I, (t_1, t_2) \in \Delta_J$ and by definition of the inner product $\langle \cdot, \cdot \rangle$ on $\overline{T(V)}$ we immediately have

$$\begin{aligned}
\langle S(X_{s_1,s_2}), S(Y_{t_1,t_2}) \rangle &= \sum_{k=0}^{\infty} \langle X^k_{s_1,s_2}, Y^k_{t_1,t_2} \rangle_{V^{\otimes k}} \\
&\leq \sum_{k=0}^{\infty} \|X^k_{s_1,s_2}\|_{V^{\otimes k}} \|Y^k_{t_1,t_2}\|_{V^{\otimes k}} && \text{(Cauchy-Schwarz)} \\
&\leq \sum_{k=0}^{\infty} \frac{\omega_X(s_1, s_2)^{k/p} \cdot \omega_Y(t_1, t_2)^{k/q}}{\beta_p(k/p)! \cdot \beta_q(k/q)!} && \text{(Ext. Theorem)} \\
&< +\infty.
\end{aligned}$$

Consider now the function $f_{(s_1,s_2),(t_1,t_2)} : G\Omega_p(V) \times G\Omega_q(V) \to \overline{T(V)} \times \overline{T(V)}$ defined as follows

$$f_{(s_1,s_2),(t_1,t_2)}(X, Y) = (S(X)_{s_1,s_2}, S(Y)_{t_1,t_2}) \tag{1.69}$$

and the function $g : \overline{T(V)} \times \overline{T(V)} \to \mathbb{R}$ defined as follows

$$g(T_1, T_2) = \langle T_1, T_2 \rangle. \tag{1.70}$$

The map g is clearly continuous in both variables in the sense of $\|\cdot\|$. By the Extension Theorem A.2.1 we know that the two maps that extend (uniquely) X and Y respectively to multiplicative functionals on the full tensor algebra $\overline{T(V)}$ are continuous in the p- and q-variation topologies respectively. Therefore $f_{(s_1,s_2),(t_1,t_2)}$ is also continuous in both of its variables. Hence, $K_{(s_1,s_2),(t_1,t_2)} = g \circ f_{(s_1,s_2),(t_1,t_2)}$ is also continuous in both variables as it is the composition of two continuous functions. □

In the next section we present our second main result. The core technical tool we use in the proof is the notion of *integral of a one-form along a rough path*, discussed in Appendix A.3.

1.4.3 A rough integral equation

To prove our main result we ought to give a meaning to the following double integral

$$\text{“}\mathcal{I}(X,Y) = \int\int K(X,Y)\langle dX, dY\rangle''. \tag{1.71}$$

We do so by constructing a double rough integral defined as the composition of two *one-forms* (Definition A.1.1) as we shall explain next. In what follows we let $W := V \oplus \overline{T(V)}$.

Remark 1.4.2. In the following construction, the spaces V, W are swapped compared to the notation used in Appendix A.3.

For a fixed tensor $A \in \overline{T(V)}$, consider the linear one-form $\alpha_A : W \to L(W, V)$ defined as follows[2]: for any $(b, B) \in W$

$$\alpha_A(b, B) = \begin{pmatrix} \langle A, B\rangle I_V & 0 \\ 0 & 0 \end{pmatrix} \tag{1.72}$$

where $I_V : V \to V$ is the identity on V and where the inner product is taken in $\overline{T(V)}$.

Remark 1.4.3. Note that a linear one-form is $Lip(\gamma)$ for all $\gamma \geq 0$ (Definition A.1.1). Hence, by Definition A.3.2 of *integral of a one-form along a rough path*, we can integrate α_A along any geometric p-rough path with $p \geq 1$.

For any $p \geq 1$ and for any fixed geometric p-rough path $Z \in G\Omega_p(V)$, consider now a second linear one-form $\beta_Z : W \to L(W, \mathbb{R})$ defined as follows: for any $(a, A) \in W$ and for any $s, t \in \Delta_I$

$$\beta_{Z_{s,t}}(a, A) = \begin{pmatrix} \left\langle \left(\int_s^t \alpha_A(Z_u) dZ_u \right)^1, I_V \right\rangle & 0 \\ 0 & 0 \end{pmatrix} \tag{1.73}$$

where the inner product is taken in V.

Remark 1.4.4. Note that the rough integral $\int \alpha_A(Z)dZ$ is a p-rough path with values in $T^{\lfloor p \rfloor}(V)$ (and that, by the Extension Theorem A.2.1, its values in $T(V), T((V))$ are also uniquely determined). Here, with the notation $(\int \alpha_A(Z_u)dZ_u)^1$ we mean the canonical projection of the rough path $\int \alpha_A(Z)dZ$ onto V.

Remark 1.4.5. We note that for any $(b, B) \in W$, the data in $b \in V$ is ignored by both one-forms α_A and β_Z when acting on (b, B). We preferred to keep this notation as we find it more in line with the standard notation used in rough integration.

[2]Perhaps more explicitly: for any $(b, B), (b', B') \in W$, $\alpha_A(b, B)(b', B') = \langle A, B\rangle b'$.

As the one-form β_Z is $Lip(\gamma)$ for all $\gamma \geq 0$, we can integrate β_Z along any q-rough path $\widetilde{Z}$ with $q \geq 1$ and use this integral as definition for the double integral $\mathcal{I}$ of Equation (1.71).

Definition 1.4.3. Let Δ_I, Δ_J be two simplices and $p, q \geq 1$ be two scalars. Let $X \in G\Omega_p(V)$ and $Y \in G\Omega_q(V)$ be two geometric p- and q-rough paths respectively. For any $(s_1, s_2) \in \Delta_I$ and any $(t_1, t_2) \in \Delta_J$, define the double rough integral $\mathcal{I}_{(s_1,s_2),(t_1,t_2)}(X, Y)$ as

$$\mathcal{I}_{(s_1,s_2),(t_1,t_2)}(X, Y) = \left(\int_{u=t_1}^{t_2} \beta_{X_{s_1,s_2}}(Y_u) dY_u \right)^1. \tag{1.74}$$

Note that this definition doesn't depend on the order of integration of X and Y. Next is our second main result, an analogue of Theorem 1.1.2 for the case of geometric rough paths.

Theorem 1.4.1. Let Δ_I, Δ_J be two simplices, $p, q \geq 1$ be two scalars, and let $X \in G\Omega_p(V)$ and $Y \in G\Omega_q(V)$ be two geometric p- and q-rough paths respectively. For any $(s_1, s_2) \in \Delta_I$ and any $(t_1, t_2) \in \Delta_J$ the rough signature kernel of Definition 1.4.2 satisfies the following equation

$$K_{(s_1,s_2),(t_1,t_2)}(X, Y) = 1 + \mathcal{I}_{(s_1,s_2),(t_1,t_2)}(X, Y) \tag{1.75}$$

where $\mathcal{I}$ is the double rough integral of Definition 1.4.3.

Proof. By [2, Theorem 4.12] if $Z \in G\Omega_p(V)$ is a geometric p-rough path and $\alpha : V \to L(V, W)$ is a $Lip(\gamma)$ one-form for some $\gamma > p$, then the mapping $Z \mapsto \int \alpha(Z) dZ$ is continuous from $G\Omega_p(V)$ to $G\Omega_p(W)$ in the p-variation topology.

For any $A \in \overline{T(V)}$ and any $\widetilde{Z} \in G\Omega_p(V)$ both α_A and $\beta_{\widetilde{Z}}$ defined in Equation (1.72) and (1.73) respectively are linear one-forms, hence $Lip(\gamma)$ for any $\gamma \geq 1$. Similarly if $\widetilde{Z} \in G\Omega_q(V)$. Thus, for any $(s_1, s_2) \in \Delta_I$ and any $(t_1, t_2) \in \Delta_J$, the map $\mathcal{I}_{(s_1,s_2),(t_1,t_2)} : G\Omega_p(V) \times G\Omega_q(V) \to \mathbb{R}$ is continuous in the p, q-variation product topology.

By Lemma 1.4.0.2, the rough signature kernel $K_{(s_1,s_2),(t_1,t_2)} : G\Omega_p(E) \times G\Omega_q(E) \to \mathbb{R}$ is also continuous with respect to the p, q-variation product topology.

Following the exact same steps as in the proof of Theorem 1.1.2, if $X \in G\Omega_1(V)$ and $Y \in G\Omega_1(V)$ are both of bounded variation then the following double integral equation holds

$$K_{(s_1,s_2),(t_1,t_2)}(X, Y) = 1 + \int_{s=s_1}^{s_2} \int_{t=t_1}^{t_2} K_{(s_1,s),(t_1,t)}(X, Y) \langle dX_s, dX_t \rangle \tag{1.76}$$

which is equivalent to the equality $K_{(s_1,s_2),(t_1,t_2)}(X, Y) = 1 + \mathcal{I}_{(s_1,s_2),(t_1,t_2)}(X, Y)$.

By Definition A.2.6 of a geometric p- (respectively q-) rough path as the limit of 1-rough paths in the p- (respectively q-) variation topology, the space of continuous paths of bounded variation $G\Omega_1(V)$ is dense $G\Omega_p(V)$ (respectively $G\Omega_q(V)$). Two continuous functions that are equal on a dense subset of a set are also equal on the whole set. The functional equation $K_{(s_1,s_2),(t_1,t_2)}(\cdot,\cdot) = 1 + \mathcal{I}_{(s_1,s_2),(t_1,t_2)}(\cdot,\cdot)$ holds on $\Omega^1 G(V) \times \Omega^1 G(V)$, which concludes the proof by the previous density argument. □

1.4.4 Cross-integrals of the rough signature kernel

In this final section, we provide another interpretation for the double integral of Equation (1.71). We begin by recalling a generalization of the Extension Theorem A.2.1.

Definition 1.4.4. Let $N \in \mathbb{N}$ be an integer. We denote by $G^N(V) \subset T^N(V)$ the free nilpotent Lie group of step N over V.

Theorem 1.4.2. [32, Theorem 14] Let $p > 1$. Let K be a closed normal subgroup of $G^{\lfloor p \rfloor}(V)$. If x is a $(G^{\lfloor p \rfloor}(V)/K)$-valued continuous path of finite p-variation, with $p \notin \mathbb{N} \setminus \{0, 1\}$, then there exists a continuous $G^{\lfloor p \rfloor}(V)$-valued geometric p-rough path y such that

$$\pi_{G^{\lfloor p \rfloor}(V), G^{\lfloor p \rfloor}(V)/K}(y) = x$$

where $\pi_{G^{\lfloor p \rfloor}(V), G^{\lfloor p \rfloor}(V)/K}$ is the canonical homomorphism (projection) from $G^{\lfloor p \rfloor}(V)$ to $G^{\lfloor p \rfloor}(V)/K$.

Corollary 1.4.2.1. If $p \in \mathbb{R}_{\geq 1} \setminus \{2, 3, ...\}$, then a continuous V-valued smooth path of finite p-variation can be lifted to a geometric p-rough path

Proof. It suffices to apply Theorem 1.4.2 to $K = \exp\{\bigoplus_{i=2}^{\lfloor p \rfloor} V_i\}$, where $v_0 = V$ and $V_{i+1} = [V, V_i]$, with $[\cdot, \cdot]$ being the Lie bracket. □

Without loss of generality let's assume $q \geq p$. $\mathbb{X}$ is a geometric p-rough path, therefore by the Extension Theorem $\mathbb{X}$ can be lifted uniquely to a geometric q-rough path $\mathbb{X}'$. Let R be any compact time interval such that such that there exists two continuous and increasing surjections $\psi_1 : R \to I$ and $\psi_2 : R \to J$. Let $\widetilde{\mathbb{X}} = \mathbb{X}' \circ \psi_1$ and $\widetilde{\mathbb{Y}} = \mathbb{Y} \circ \psi_2$.

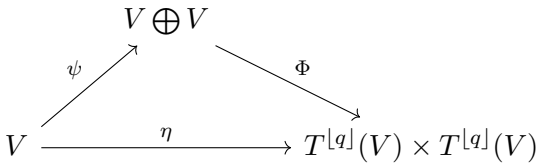
Consider the path $\mathbb{Z} : R \to G^{\lfloor q \rfloor}(V) \times G^{\lfloor q \rfloor}(V)$ defined as

$$\mathbb{Z} : t \mapsto (\widetilde{\mathbb{X}}_t, \widetilde{\mathbb{Y}}_t) \tag{1.77}$$

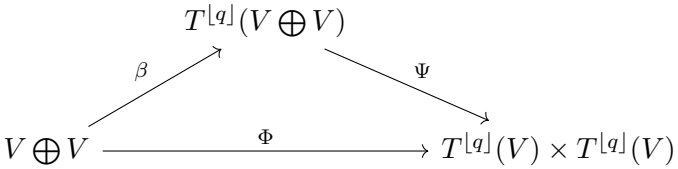
$\mathbb{Z}$ is a continuous, $(G^{\lfloor q \rfloor}(V) \times G^{\lfloor q \rfloor}(V))$-valued path of finite q-variation. Firstly, we consider the *product of algebras* $T^{\lfloor q \rfloor}(V) \times T^{\lfloor q \rfloor}(V)$, where the product of elements is defined by the following operation: $(f_1, g_1)(f_2, g_2) = (f_1 \otimes f_2, g_1 \otimes g_2)$.

Now consider the free tensor algebra $T^{\lfloor q \rfloor}(V \bigoplus V)$ over the vector space $V \bigoplus V$. Let $\phi : V \to T^{\lfloor q \rfloor}(V)$ be the canonical inclusion of V into $T^{\lfloor q \rfloor}(V)$ and let $\psi : T^{\lfloor q \rfloor}(V) \to T^{\lfloor q \rfloor}(V) \times T^{\lfloor q \rfloor}(V)$ be the linear map defined as $\psi(T) = (T, T), \forall T \in T(V)$.

Now let's consider the map $\eta = \psi \circ \phi : V \to T^{\lfloor q \rfloor}(V) \times T^{\lfloor q \rfloor}(V)$. By the universal property of $\bigoplus$ there exists a unique algebra homomorphism $\Phi : V \bigoplus V \to T^{\lfloor q \rfloor}(V) \times T^{\lfloor q \rfloor}(V)$ such that $\Phi \circ \psi = \eta$.



But now $T^{\lfloor q \rfloor}(V \bigoplus V)$ has also the universal property, therefore there exists a unique algebra homomorphism $\Psi : T^{\lfloor q \rfloor}(V \bigoplus V) \to T^{\lfloor q \rfloor}(V) \times T^{\lfloor q \rfloor}(V)$ such that $\Psi \circ \beta = \Phi$, where β is the canonical inclusion of $V \bigoplus V$ into $T^{\lfloor q \rfloor}(V \bigoplus V)$.



Note that $G^{\lfloor q \rfloor}(V) \times G^{\lfloor q \rfloor}(V)$ is a group embedded in the product algebra $T^{\lfloor q \rfloor}(V) \times T^{\lfloor q \rfloor}(V)$ and $G^{\lfloor q \rfloor}(V \bigoplus V)$ is a group embedded in the tensor algebra $T^{\lfloor q \rfloor}(V \bigoplus V)$. Let π be the map Ψ restricted to $G^{\lfloor q \rfloor}(V \bigoplus V)$. Given that $G^{\lfloor q \rfloor}(V) \times G^{\lfloor q \rfloor}(V) \subset \pi(G^{\lfloor q \rfloor}(V \oplus V))$, this map is a surjective group-homomorphism. Therefore, by the *First Group Isomorphism Theorem* we have that $Ker(\pi) \triangleleft G^{\lfloor q \rfloor}(V \oplus V)$, and

$$G^{\lfloor q \rfloor}(V \oplus V)/Ker(\pi) \simeq G^{\lfloor q \rfloor}(V) \times G^{\lfloor q \rfloor}(V)$$

By Lemma 1.4.2 there exists a continuous $G^{\lfloor q \rfloor}(V \oplus V)$-valued geometric q-rough path $\widetilde{\mathbb{Z}}$ such that $\pi(\widetilde{\mathbb{Z}}) = \mathbb{Z}$. Expanding out coordinate-wise we obtain

$$\begin{aligned}
\int_{s=u}^{u'}\int_{t=v}^{v'} k(\mathbb{X}_{u,s},\mathbb{Y}_{v,t})\langle d\mathbb{X}_s, d\mathbb{Y}_t\rangle &= \sum_{n=0}^{\lfloor q\rfloor}\sum_{\tau\in\{1,\dots,d\}^n}\int_{s=u}^{u'}\int_{t=v}^{v'} k(\mathbb{X}_s,\mathbb{Y}_t)d\mathbb{X}_s^\tau d\mathbb{Y}_t^\tau \\
&= \sum_{n=0}^{\lfloor q\rfloor}\sum_{\tau\in\{1,\dots,d\}^n}\int_{s=u}^{u'}\int_{t=v}^{v'} \langle S(\mathbb{X}_{u,s}), S(\mathbb{Y}_{v,t})\rangle d\mathbb{X}_s^\tau d\mathbb{Y}_t^\tau \\
&= \sum_{m=0}^{\infty}\sum_{\omega\in\{1,\dots,d\}^m}\sum_{n=0}^{\lfloor q\rfloor}\sum_{\tau\in\{1,\dots,d\}^n}\int_{s=u}^{u'}\int_{t=v}^{v'} S(\mathbb{X}_{u,s})^\omega S(\mathbb{Y}_{v,t})^\omega d\mathbb{X}_s^\tau d\mathbb{Y}_t^\tau \\
&= \sum_{m=0}^{\infty}\sum_{\omega\in\{1,\dots,d\}^m}\sum_{n=0}^{\lfloor q\rfloor}\sum_{\tau\in\{1,\dots,d\}^n}\Big(\int_{s=u}^{u'} S(\mathbb{X}_{u,s})^\omega d\mathbb{X}_s^\tau\Big)\Big(\int_{t=v}^{v'} S(\mathbb{Y}_{v,t})^\omega d\mathbb{Y}_t^\tau\Big)
\end{aligned} \tag{1.78}$$

Note that all the cross-integrals of $S(\mathbb{X})$ and $S(\mathbb{Y})$ do not contribute in the above expression, which nicely factors into two separate integrals: expression (1.78) tells us that the rough path $\widetilde{\mathbb{Z}}$ does not depend on the lift used in the extension (from the joint path $\mathbb{Z}$ to the rough path $\widetilde{\mathbb{Z}}$). The terms involved in the infinite sum on the right-hand-side of the equation (1.78) are all $\mathbb{R}$-projections of the signature of the rough paths $\mathbb{X}$ and $\mathbb{Y}$.

1.5 Machine learning applications

In this section we evaluate our signature PDE kernel on three different learning tasks. Firstly, we consider the task of multivariate time series classification on UEA[3] datasets [33] with a *support vector classifier* (SVC) and compare the performance obtained by equipping the same SVC configuration with various kernel functions, including ours. Secondly, we run a regression task to predict future (average) bitcoin prices from previously observed prices by means of a *support vector regressor* (SVR) and similarly to the previous experiment, we compare the performance produced by a variety of kernels. Lastly, we show how the signature kernel can be incorporated within a simple optimization procedure to represent the distribution of a large ensemble of paths as a weighted average of a small number of selected paths from the ensemble whilst maintaining certain statistical properties [34].

In the presence of sequential inputs, well-designed kernels must be chosen with care [35]. In the case where all the time series inputs are of the same length, standard kernels on $\mathbb{R}^d$ can be deployed by stacking each dimension of the time series into one single vector. Standard choices of kernels include the linear and Gaussian (a.k.a. RBF) kernels. When the series are not of the same length, other kernels specifically designed for time series can be used to address this issue. Other than the signature PDE kernel introduced in this chapter, to our

[3]Data available at `https://timeseriesclassification.com`

knowledge only two other kernels for sequential data have been proposed in the literature: the truncated signature kernel [1] (Sig(n) - where n denotes the truncation level) and the *global alignment kernel* (GAK) [14]. For the classification and regression experiments we made use of the SVC and SVR estimators respectively, from the popular python library `tslearn` [36].

Hyperparameter selection The hyperparameters of the SVC and SVR estimators were selected by cross-validation via a grid search on the training set. For the classification we used the train-test split as provided by UEA, whilst for the regression we used a 80-20 split. Both the SVC and SVR estimators depend on a kernel k and on two scalar parameters C and γ. The range of values for C was chosen to be $\{1, 10, ..., 10^4\}$ and the one for γ to be $\{10^{-4}, ..., 10^4\}$ for all kernel functions included in the comparison. We benchmark our Sig-PDE kernel against the Linear, RBF and GAK [14] kernels as well as the truncated signature kernel Sig(n) from [1]. We found that the algorithm provided to us by [37] was in general slower than directly computing the truncated signatures with `iisignature` [38]. This is because the former was implemented as pure python code, whilst `iisignature` is highly optimized and uses a `C++` backend. For this reason, we ended up computing Sig(n) without kernel trick for all the experiments. The truncation n is chosen from the range $\{2, ..., 6\}$. Furthermore, we added a variety of additional hyperparameters to Sig(n) consisting of: 1) scaling the paths by different scalar scales, 2) normalizing the truncated signatures by multiplying (or not) each level $\ell \in \{1, ..., n\}$ by $\ell!$, 3) equipping the SVC/SVR with a Linear or RBF kernel (indexed on truncated signatures). For our Sig-PDE we used the RBF-lifted version with parameter σ taken in the range $\{10^{-3}, ..., 10^1\}$. All the experiments are reproducible using the code in `https://github.com/crispitagorico/sigkernel` and following the instructions thereafter.

1.5.1 Multivariate time series classification

The support vector classifier (SVC) [39] is one of the simplest yet widely used supervised learning models for classification. It has been successfully used in the fields of text classification [40], image retrieval [41], mathematical finance [42], medicine [43] etc. We considered various UEA datasets [33] of input-output pairs $\{(x_i, y_i)\}_{i=1}^n$ where each x_i is a multivariate time series and each y_i is the corresponding class. In Table 1.1 we display the performance of the same SVC equipped with different kernels (including ours). As the results show, our Sig-PDE kernel is systematically among the top 2 classifiers across all the datasets (except for FingerMovements and UWaveGestureLibrary) and always outperforms its truncated counterpart, that often overfits during training.

Datasets/Kernels	Linear	RBF	GAK	Sig(n)	Sig-PDE
ArticularyWordRecognition	98.0	98.0	98.0	92.3	**98.3**
BasicMotions	87.5	97.5	97.5	97.5	**100.0**
Cricket	91.7	91.7	**97.2**	86.1	**97.2**
ERing	92.2	92.2	**93.7**	84.1	93.3
Libras	73.9	77.2	79.0	**81.7**	**81.7**
NATOPS	90.0	92.2	90.6	88.3	**93.3**
RacketSports	76.9	78.3	84.2	80.2	**84.9**
FingerMovements	57.0	60.0	**61.0**	51.0	58.0
Heartbeat	70.2	73.2	70.2	72.2	**73.6**
SelfRegulationSCP1	86.7	87.3	**92.4**	75.4	88.7
UWaveGestureLibrary	80.0	**87.5**	**87.5**	83.4	87.0

Table 1.1: Test set classification accuracy (in %) on UEA multivariate time series datasets.

1.5.2 Predicting Bitcoin prices

In the last few years, there has a remarkable rise of cryptocurrency trading where the most popular currency, Bitcoin, reached its peak at almost $20,000$ USD/BTC at the end of the year 2017 followed by a big crash in November 2018, when the price dropped to around 3000 USD/BTC. In this section, we will use daily Bitcoin to USD prices data[4] from Gemini which is one of the biggest cryptocurrency trading platforms in the US. Our goal is to use a window of size 36 days to predict the mean price of the next 2 days. As shown in Table 1.2, SVR equipped with the Sig-PDE is able to generalise better to unseen prices and produces better predictions on the test set in terms of MAPE compared to all other benchmarks. We also note that the truncated signature kernel doesn't seem to generalize well to unseen observation for this regression example. In Figure 1.4 we plot the predictions obtained with SVR-Sig-PDE on the train and test sets.

Kernel	RBF	GAK	Sig(n)	Sig-PDE
MAPE	4.094	4.458	13.420	**3.253**

Table 1.2: Test set *mean absolute percentage error* (MAPE) (in %) to predict the average Bitcoin price over the next 2 days given prices over the previous 36 days.

1.5.3 Reducing the support of a discrete measure on paths

As described in [44], *herding* refers to any procedure to approximate integrals of functions in a *reproducing kernel Hilbert space* (RKHS). In particular, such procedure can be useful to estimate *kernel mean embeddings* (KMEs) as we shall explain next. Consider a set $\mathcal{X}$ and a feature map Φ from $\mathcal{X}$ to an RKHS H with k being the associated positive definite kernel. All elements of H may be identified with real functions f on $\mathcal{X}$ defined by $f(x) = \langle f, \Phi(x) \rangle$

[4]Data is from `https://www.cryptodatadownload.com/`

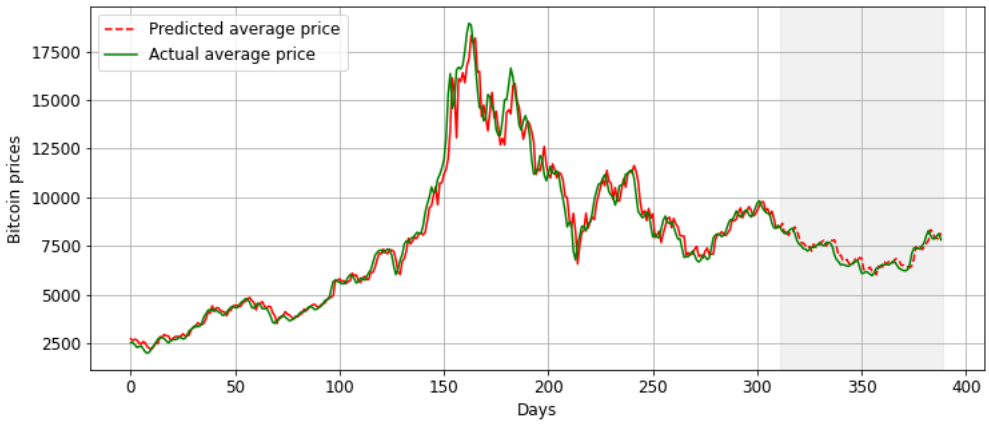


Figure 1.4: SVR-Sig-PDE predictions of average Bitcoin prices over the next 2 days from Bitcoin prices over the previous 36 days. In white are the predicted prices in the training set; in grey are the predicted prices in the test set. Trading days range from '2017-06-01' to '2018-08-01'.

for $x \in \mathcal{X}$. Following [45] for a fixed probability measure μ on $\mathcal{X}$ we seek to approximate the KME $\mathbb{E}_\mu \Phi := \int_{\mathcal{X}} \Phi(x) d\mu(x)$, that belongs to the convex hull of $\{\Phi(x)\}_{x \in \mathcal{X}}$ [46]. To approximate $\mathbb{E}_\mu \Phi$, we consider n points $x_1, ..., x_n \in \mathcal{X}$ combined linearly with positive weights $w_1, ..., w_n$ that sum to 1. We then consider the discrete measure $\nu = \sum_{i=1}^n w_i \delta_{x_i}$ and as shown in [46] we have that

$$\sup_{f \in H, ||f|| \leq 1} |\langle \mathbb{E}_\nu \Phi, f \rangle - \langle \mathbb{E}_\mu \Phi, f \rangle| = ||\mathbb{E}_\nu \Phi - \mathbb{E}_\mu \Phi||_H \tag{1.79}$$

which means that controlling $\mathbb{E}_\nu \Phi - \mathbb{E}_\mu \Phi$ is enough to control the error in computing the expectation for all $f \in H$ with norm bounded by 1. We are interested in the setting where $\mathcal{X}$ is a set of paths of bounded variation taking values on a d-dimensional space E (or in practice a set of multivariate time series for example). The signature being a natural feature map for sequential data we set $\Phi = S$, k to be the untruncated signature kernel and $H = \overline{T(E)}^*$. Following [47, 34], we consider the problem of reducing the size of the support in $\mathcal{X}$ of a discrete measure μ whilst preserving some of its statistical properties. Suppose $\#supp(\mu) = N$, where N is large, and $\mu = \sum_{i=1}^N \alpha_i \delta_{x_i}, x_i \in \mathcal{X}$. We call a measure ν on $\mathcal{X}$ a *reduced measure* with respect to μ if

$$1)\ supp(\nu) \subset supp(\mu) \quad \text{and} \quad 2)\ \|\mathbb{E}_\nu S - \mathbb{E}_\mu S\| < \epsilon \quad \text{(for some small } \epsilon > 0).$$

We fix the size of the support of the reduced measure ν to be $\#supp(\nu) = n$, so that $n << N$. Because of condition 1) we have that ν is of the form $\mu = \sum_{i=1}^N \beta_i \delta_{x_i}$, where all but n of the weights β_i's are equal to 0. Therefore the vector of weights $\beta = (\beta_1, ..., \beta_N) \in \mathbb{R}^N$ is sparse.

We are interested in the following optimization problem

$$\begin{aligned}\min_{\beta\in\mathbb{R}^N} ||\mathbb{E}_\nu S - \mathbb{E}_\mu S||^2_{T(E)} &= \min_{\beta\in\mathbb{R}^N} ||\sum_{i=1}^N (\alpha_i - \beta_i)S(x_i)||^2_{T(E)} \\ &= \min_{\beta\in\mathbb{R}^N} \Big\langle \sum_{i=1}^N (\alpha_i - \beta_i)S(x_i), \sum_{j=1}^N (\alpha_j - \beta_j)S(x_i) \Big\rangle_{T(E)} \\ &= \min_{\beta\in\mathbb{R}^N} \underbrace{\sum_{i,j=1}^N (\alpha_i - \beta_i)(\alpha_j - \beta_j)k(x_i, x_j)}_{:=L(\beta)}\end{aligned}$$

where k is the signature kernel. This minimisation will not yield a sparse vector β. To induce sparsity we use an l_1 penalisation on the weights β as in LASSO, which amounts to the following Lagrangian minimisation

$$\min_{\beta\in\mathbb{R}^N} L(\beta) + \lambda||\beta||_1 \tag{1.80}$$

where λ is a penalty parameter determined by the size n of the support of ν. Equation (1.80) minimises a function $f : \mathbb{R}^N \to \mathbb{R}$ that can be decomposed as $f = L + h$, where L is differentiable and $h = \lambda||\cdot||_1$ is convex but non-differentiable, so a gradient descent algorithm can't be directly applied. *Subgradient descent methods* are classical algorithms that address this issue but have poor convergence rates [48]. A better choice of algorithms for this particular problem are called *proximal gradient methods* [49]. Define the soft-thresholding operator $A_\gamma : \mathbb{R}^N \to \mathbb{R}^N$ as follows

$$A_\gamma(\beta)_i = \begin{cases} \beta_i - \gamma, & \text{if } \beta_i > \gamma \\ 0, & \text{if } |\beta_i| \leq \gamma \\ \beta_i + \gamma, & \text{if } \beta_i < -\gamma. \end{cases} \tag{1.81}$$

Then, it can be shown [49] that β^* is a minimiser of the optimisation (1.80) if and only if β^* solves the following fixed point problem

$$\beta^* = A_\gamma(\beta^* - \gamma\nabla_\beta L(\beta^*)). \tag{1.82}$$

The fixed point problem Equation (1.82) can be solved iteratively fixing $\beta^0 \in \mathbb{R}^N$ and for $k \geq 1$

$$\beta^{k+1} = A_\gamma(\beta^k - \gamma\nabla_\beta L(\beta^k)). \tag{1.83}$$

Proximal gradient descent methods convergence with rate $O(1/\epsilon)$ which is an order of magnitude better that the $O(1/\epsilon^2)$ convergence rate of subgradient methods [49].

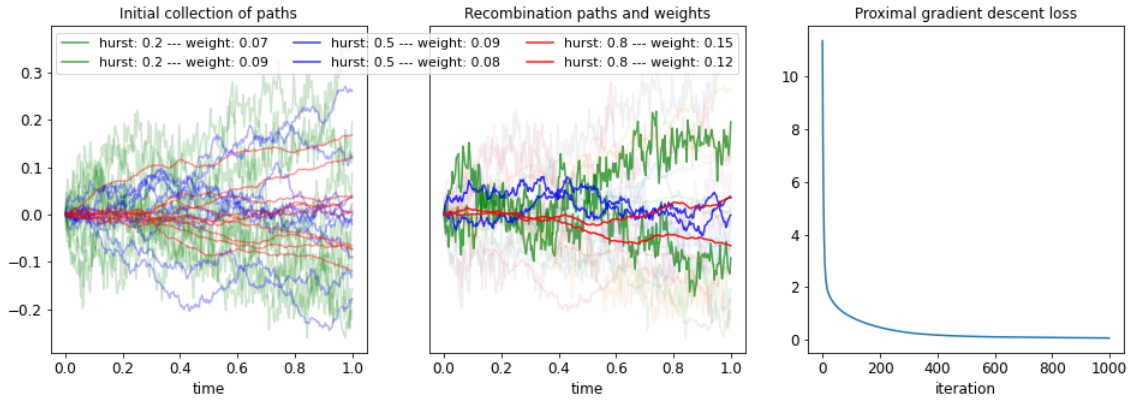


Figure 1.5: On the **left** are 30 fBM sample paths. In the **middle** are the results obtained by solving the optimisation Equation (1.80). On the **right** is the loss as a function of the proximal gradient descent iteration.

We apply the above proximal gradient descent algorithm to an example of a set of 30 sample paths of fractional Brownian Motion (fBM) with Hurst exponent drawn uniformly at random from $\{0.2, 0.5, 0.8\}$ (note that fBM(0.5) corresponds to Brownian motion). The goal is to compute a reduced measure with smaller support size. We choose a value of the penalisation constant λ in Equation (1.80) so that the new support is of size $= 6$. The selected paths with corresponding weights are displayed in Figure 1.5. This selection is clearly well-balanced across the samples (2 paths per exponent) so more likely to well-represent the initial ensemble.

1.6 Conclusion

In this chapter we introduce the signature PDE kernel and show that when paths are continuously differentiable the latter solves a hyperbolic PDE. We recognize the connection with a well known class of differential equations known in the literature as Goursat problems. Our Goursat PDE can be solved numerically using state-of-the-art hyperbolic PDE solvers; we propose an efficient finite different scheme to do so that has linear time complexity when implemented on `GPU` and analyse its convergence properties. We extend the previous analysis to the case of geometric rough paths and establish a rough integral equation analogous to the aforementioned Goursat problem. Finally we demonstrated the effectiveness of our kernel in various data science applications dealing sequential data.

Chapter 2

Distribution Regression for Sequential Data

Distribution regression (DR) refers to the supervised learning problem where labels are only available for groups of inputs rather than for individual inputs. In this chapter, we develop a rigorous mathematical framework for distribution regression on sequential data, i.e. where inputs are groups of complex data streams.

For instance, in thermodynamics (see Figure 2.1) one may be interested in determining the temperature of a gas from the set of trajectories described by its particles [50]. Similarly, in quantitative finance practitioners may wish to estimate mean-reversion parameters from observed market dynamics [51, 28]. Another example of distribution regression arises in agricultural science where the challenge consists in predicting the overall end-of-year crop yield from high-resolution climatic data across a field [52, 53].

DR techniques have been successfully applied to handle Euclidean inputs. Recently, there has been an increased interest in extending these techniques to non-standard inputs such as images. However DR for sequential data, such as multivariate time-series, has been largely ignored.

In this chapter we propose a framework for DR that addresses precisely the setting where the inputs within each group are data streams, mathematically thought of as continuous paths of bounded variation over E (recall the notation $G\Omega_1(E)$). We formulate two distinct approaches, one feature-based and the other kernel-based, both relying on the expected signature [17, 54, 55, 56]. Firstly, we construct a new set of features that are universal in the sense that any continuous real-valued function acting on distributions on path space can be uniformly well-approximated by a linear combination of these features (Section 2.2.1). Secondly, we introduce a universal kernel on distributions on paths given by the composition

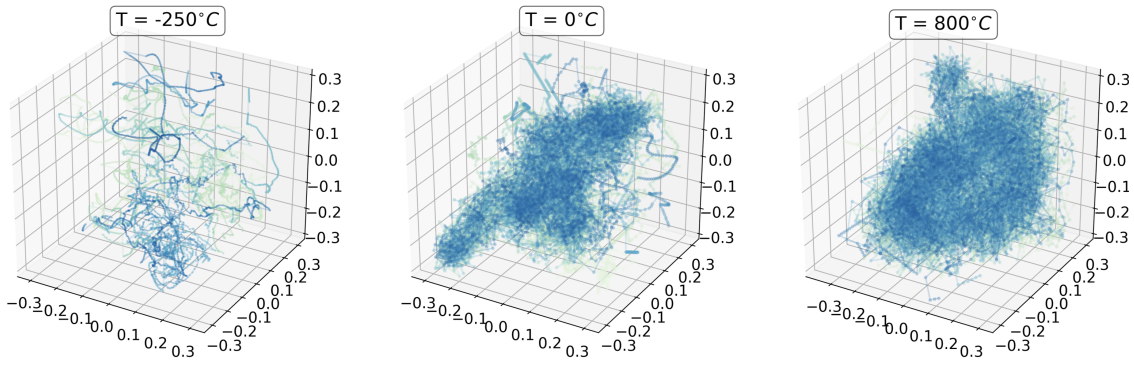


Figure 2.1: Simulation of the trajectories traced by 20 particles of an ideal gas in a 3-d box under different thermodynamic conditions. Higher temperatures equate to a higher internal energy in the system which increases the number of collisions resulting in different large-scale dynamics of the gas.

of the expected signature and a Gaussian kernel (Section 2.2.2), which can be evaluated thanks to the PDE kernel trick developed in the previous chapter. The former method is more suitable to datasets containing a large number of low dimensional streams, whilst the latter is more suitable for datasets with a low number of high dimensional streams. We demonstrate the versatility of our methods to handle interacting trajectories like the ones depicted in Figure 2.1. We show how these two methods can be used to provide practical DR algorithms for multivariate time-series, which are robust to irregular sampling and achieve state-of-the-art performance on synthetic and real-world examples (Section 2.3).

Consider M input-output pairs $\{(\{\mathbf{x}^{i,p}\}_{p=1}^{N_i}, y^i)\}_{i=1}^{M}$, where each pair is given by a scalar target $y^i \in \mathbb{R}$ and a group of N_i d-dimensional time-series of the form

$$\mathbf{x}^{i,p} = \{(t_1, \mathbf{x}_1^{i,p}), \dots, (t_{\ell_{i,p}}, \mathbf{x}_{\ell_{i,p}}^{i,p})\}, \tag{2.1}$$

of possibly unequal lengths $\ell_{i,p} \in \mathbb{N}$, with time-stamps $t_1 \leq \ldots \leq t_{\ell_{i,p}}$ and values $\mathbf{x}_k^{i,p} \in \mathbb{R}^d$. Every time-series $\mathbf{x}^{i,p}$ can be naturally embedded into a continuous path

$$x^{i,p} : [t_1, t_{l_{i,p}}] \to \mathbb{R}^d, \tag{2.2}$$

by piecewise linear interpolation with knots at $t_1, \dots, t_{l_{i,p}}$ such that $x_{t_k}^{i,p} = \mathbf{x}_k^{i,p}$. After having formally introduced a set of probability measures on this class of paths, we will summarize the information on each set $\{x^{i,p}\}_{p=1}^{N_i}$ by the empirical measure $\delta^i = \frac{1}{N_i}\sum_{p=1}^{N_i} \delta_{x^{i,p}}$ where $\delta_{x^{i,p}}$ is the Dirac measure centred at the path $x^{i,p}$. The supervised learning problem we propose to solve consists in learning an unknown function $F : \delta^i \mapsto y^i$.

Given a compact subset of paths $\mathcal{X} \subset G\Omega_1(E)$, we denote by $\mathcal{P}(\mathcal{X})$ the set of (Borel) probability measures on $\mathcal{X}$.

2.1 The expected signature of a measure on paths

The sequence of moments $(\mathbb{E}[Z^{\otimes m}])_{m\geq 0}$ is classically known to characterize the law $\mu_Z = \mathbb{P}\circ Z^{-1}$ of any finite-dimensional random variable Z (provided the sequence does not grow too fast). It turns out that in the infinite dimensional case of laws of paths-valued random variables (or equivalently of probability measures on paths) an analogous result holds [17]. It says that one can fully characterise a probability measure on paths (provided it has compact support) by replacing monomials of a vector by iterated integrals of a path (i.e. signatures). At the core of this result is a recent tool from stochastic analysis that we introduce next.

Definition 2.1.1 (**Expected Signature**)**.** The expected signature is the map $\mathbb{E}S : \mathcal{P}(\mathcal{X}) \to T(E)$ defined element-wise for any $k \geq 0$ and any $w = e_{i_1} \otimes \ldots \otimes e_{i_k}$ as

$$\mathbb{E}S(\mu)^w = \int_{x\in\mathcal{X}} S(x)^w \mu(dx), \tag{2.3}$$

where we omitted the indexing time-interval to make the notation lighter.

We will rely on the following important theorem in order to prove the universality of the proposed techniques for DR on sequential data presented in the next two sections.

Definition 2.1.2. A sequence of probability measures $\mu_n \in \mathcal{P}(\mathcal{X})$ *converges weakly* to μ if for every $f \in C_b(\mathcal{X}, \mathbb{R})$ we have $\int_{\mathcal{X}} f d\mu_n \to \int_{\mathcal{X}} f d\mu$ as $n \to \infty$, where $C_b(\mathcal{X}, \mathbb{R})$ is the space of real-valued continuous bounded functions on $\mathcal{X}$.

Remark 2.1.1. Since $\mathcal{X}$ is a compact metric space, we can drop the word "bounded" in Definition 2.1.2.

Definition 2.1.3. Given two probability measures $\mu, \nu \in \mathcal{P}(\mathcal{X})$, the Wasserstein-1 distance is defined as follows

$$W_1(\mu,\nu) = \inf_{\gamma\in\Pi(\mu,\nu)} \int_{x,y\in\mathcal{X}} \|x-y\|_{Lip}\, d\gamma(x,y) \tag{2.4}$$

where the infimum is taken over all possible couplings of μ and ν.

Lemma 2.1.0.1. [17, Theorem 5.3] The signature $S : \mathcal{C}(I,E) \to T(E)$ is injective.[1]

Lemma 2.1.0.2. [54, Corollary 5.5] The signature $S : \mathcal{C}(I,E) \to T(E)$ is continuous w.r.t. $\|\cdot\|_{Lip}$.

Lemma 2.1.0.3. [17, Theorem 5.6] The expected signature $\mathbb{E}S : \mathcal{P}(\mathcal{X}) \to T(E)$ is injective.[2]

[1] Up to tree-like equivalence (see [17, appendix B] for a definition and detailed discussion).

[2] This result was firstly proved in [57] for probability measures supported on compact subsets of $\mathcal{C}(I,E)$, which is enough for this paper. It was also proved in a more abstract setting in [54]. The authors of [17] introduce a normalization that is not needed in case of compact supports, as they mention in [17, (I) - page 2]

Theorem 2.1.1. The expected signature $\mathbb{E}S : \mathcal{P}(\mathcal{X}) \to T(E)$ is weakly continuous.

Proof. Consider a sequence $\{\mu_n\}_{n\in\mathbb{N}}$ of probability measures on $\mathcal{P}(\mathcal{X})$ converging weakly to a measure $\mu \in \mathcal{P}(\mathcal{X})$. By Lemma 2.1.0.2 the signature $S : x \mapsto S(x)$ is continuous w.r.t. $\|\cdot\|_{Lip}$. Hence, by definition of weak-convergence (and because $\mathcal{X}$ is compact), for any $k > 0$ and any multi-index $(i_1, \dots, i_k) \in \{1, \dots, d\}^k$ it follows that

$$\int_{x\in\mathcal{X}} S(x)^{(i_1,\dots,i_k)}\mu_n(dx) \to \int_{x\in\mathcal{X}} S(x)^{(i_1,\dots,i_k)}\mu(dx).$$

The factorial decay from Equation (1.9) yields

$$\int_{x\in\mathcal{X}} S(x)\mu_n(dx) \to \int_{x\in\mathcal{X}} S(x)\mu(dx)$$

in the topology induced by $\langle\cdot,\cdot\rangle_{T(E)}$. □

2.2 Methods for DR on sequential data

The *distribution regression* (DR) setting for sequential data we have set up so far consists of M groups of input-output pairs of the form

$$\left\{\left(\{x^{i,p} \in G\Omega_1(E)\}_{p=1}^{N_i}, y^i \in \mathbb{R}\right)\right\}_{i=1}^{M}, \tag{2.5}$$

such that the finite set of paths $\mathcal{X} = \bigcup_{i=1}^{M}\{x^{i,p}\}_{p=1}^{N_i}$ is a compact subset of $G\Omega_1(E)$. As mentioned above, we can summarize the information carried by the collection of paths $\{x^{i,p}\}_{p=1}^{N_i}$ in group i by considering the empirical measure

$$\delta^i = \frac{1}{N_i}\sum_{p=1}^{N_i}\delta_{x^{i,p}} \in \mathcal{P}(\mathcal{X}) \tag{2.6}$$

where $\delta_{x^{i,p}}$ is the Dirac measure centred at the path $x^{i,p}$. In this way the input-output pairs in (2.5) can be represented as follows

$$\left\{\left(\delta^i \in \mathcal{P}(\mathcal{X}), y^i \in \mathbb{R}\right)\right\}_{i=1}^{M}. \tag{2.7}$$

2.2.1 Method 1: a feature-based approach (SES)

As stated in Theorem 1.0.1, linear combinations of path-iterated-integrals are universal approximators for continuous functions f on compact sets of paths. In this section we prove the analogous density result for continuous functions F on probability measures on paths. We do so by reformulating the problem of DR on paths as a linear regression on the iterated integrals of an object that we will refer to as the pathwise expected signature. We start with the definition of this term followed by the density result. Ultimately, we show that our DR algorithm materializes as extracting signatures of (pathwise expected) signatures.

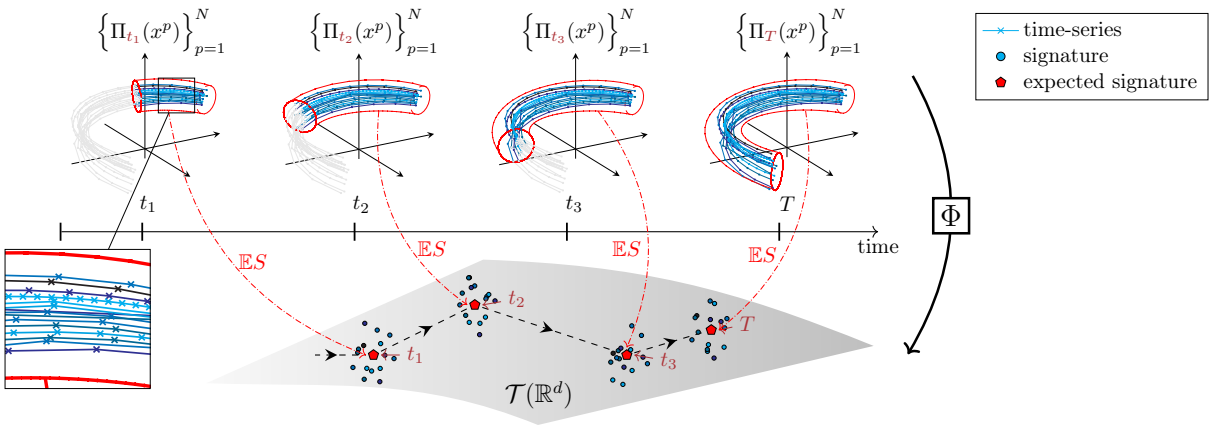


Figure 2.2: Schematic overview of the action of pathwise expected signature Φ on a group of time-series $\{x^p\}_{p=1}^N$. (top) Representation of the information about the group of time-series available from start up to time t_k. (bottom) At each time t_k this information gets embedded into a single point in $T(\mathbb{R}^d)$.

Definition 2.2.1. For any $t \in I = [a, T]$ define the projection

$$\Pi_t : G\Omega_1(E) \to G\Omega_1(E) \tag{2.8}$$

that maps any path x to its restriction to the sub-interval $[a, t] \subset I$, such that $\Pi_t(x) = x_{|_{[a,t]}}$.

Definition 2.2.2. The pathwise expected signature is the function $\Phi : \mathcal{P}(\mathcal{X}) \to G\Omega_1(T(E))$ that to a probability measure $\mu \in \mathcal{P}(\mathcal{X})$ associates the path $\Phi(\mu) : I \to T(E)$ defined as

$$\Phi(\mu) : t \mapsto \mathbb{E}_{x\sim\mu}\left[S \circ \Pi_t(x)\right]. \tag{2.9}$$

Theorem 2.2.1. [58, Theorem 3.7] Let $x \in \mathcal{C}(I, E)$ and recall the definition of the projection $\Pi_t : x \mapsto x_{|_{[a,t]}}$. Then, the $T(E)$-valued path defined by

$$S_{path}(x) : t \mapsto S \circ \Pi_t(x) \tag{2.10}$$

is Lipschitz continuous. Furthermore the map $x \mapsto S_{path}(x)$ is continuous w.r.t. $\|\cdot\|_{Lip}$.

Theorem 2.2.2. The pathwise expected signature $\Phi : \mathcal{P}(\mathcal{X}) \to \mathcal{C}(I, T(E))$ is injective.[3]

Proof. Let $\mu, \nu \in \mathcal{P}(\mathcal{X})$ be two probability measures. If $\Phi(\mu) = \Phi(\nu)$, then for any $t \in I$, $\mathbb{E}_{x\sim\mu}[S \circ \Pi_t(x)] = \mathbb{E}_{y\sim\nu}[S \circ \Pi_t(y)]$. In particular, for $t = T$, $\mathbb{E}S(\mu) = \mathbb{E}_{x\sim\mu}[S(x)] = \mathbb{E}_{y\sim\nu}[S(y)] = \mathbb{E}S(\nu)$. The result follows from the injectivity of the expected signature $\mathbb{E}S$ (Lemma 2.1.0.3). □

[3]For any $\mu \in \mathcal{P}(\mathcal{X})$ the path $\Phi(\mu) \in \mathcal{C}(I, T(E))$. Indeed $\Phi(\mu)$ is a continuous path because x, Π_t, S and Φ are all continuous and the composition of continuous functions is continuous. The Lipschitzianity comes from the fact that $\|\Phi(\mu)\|_{Lip} \leq \mu(\mathcal{X}) \sup_{x\in\mathcal{X}} \|S_{path}(x)\|_{Lip} < +\infty$ by Theorem 2.2.1.

Theorem 2.2.3. The pathwise expected signature $\Phi : \mathcal{P}(\mathcal{X}) \to \mathcal{C}(I, T(E))$ is weakly continuous.

Proof. Let $\{\mu_n\}_{n\in\mathbb{N}}$ be a sequence in $\mathcal{P}(\mathcal{X})$ converging weakly to $\mu \in \mathcal{P}(\mathcal{X})$. As S_{path} is continuous (Theorem 2.2.1), it follows, by the *continuous mapping theorem*, that $S_{path}\#\mu_n \to S_{path}\#\mu$ weakly, where $S_{path}\#\mu$ is the pushforward measure of μ by S_{path}. Given that S_{path} is continuous and $\mathcal{X}$ is compact, it follows that the image $S_{path}(\mathcal{X})$ is a compact subset of the Banach space $\mathcal{C}(I, T(E))$. By [59, Theorem 6.8] weak convergence of probability measures on compact supports is equivalent to convergence in *Wasserstein-1 distance*. By *Jensen's inequality* $\|\mathbb{E}[S_{path}\#\mu_n] - \mathbb{E}[S_{path}\#\mu]\|_{Lip} \leq \mathbb{E}[\|S_{path}\#\mu_n - S_{path}\#\mu\|_{Lip}]$. Taking the infimum over all couplings $\gamma \in \Pi(S_{path}\#\mu_n, S_{path}\#\mu)$ on the right-hand-side of the previous equation we obtain $\|\mathbb{E}[S_{path}\#\mu_n] - \mathbb{E}[S_{path}\#\mu]\|_{Lip} \leq W_1(S_{path}\#\mu_n, S_{path}\#\mu) \to 0$, which yields the convergence $\mathbb{E}[S_{path}\#\mu_n] \to \mathbb{E}[S_{path}\#\mu]$ in $\|\cdot\|_{Lip}$ over $\mathcal{C}(I, T(E))$. Noting that $\mathbb{E}[S_{path}\#\mu] = \Phi(\mu)$ concludes the proof. □

The action of the function Φ is illustrated on Figure 2.2, and its implementation is outlined in Algorithm 1.[4]

Algorithm 1 Pathwise Expected Signature (PES)

Input: N streams $\{\mathbf{x}^p\}_{p=1}^N$ each of length ℓ
Create array Φ to store the PES
Create array S to store the signatures
Initialize $S[p] \leftarrow 1$ for $p \in \{1, \dots, N\}$
for each time-step $k \in \{2, \dots, \ell\}$ **do**
    $S[p] \leftarrow S[p] \otimes \exp(\mathbf{x}_k^p - \mathbf{x}_{k-1}^p)$ for $p \in \{1, \dots, N\}$
    $\Phi[k] \leftarrow \text{avg}(S)$
Output: The pathwise expected signature Φ

The next theorem states the universality of Method 1 (SES), i.e. that any weakly continuous function on $\mathcal{P}(\mathcal{X})$ can be uniformly well approximated by a linear combination of terms in the signature of the pathwise expected signature.

Theorem 2.2.4. Let $\mathcal{X} \subset G\Omega_1(E)$ be a compact set of paths and consider a weakly continuous function $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$. Then for any $\epsilon > 0$ there exists a truncation level $m \geq 0$ and a collection of scalars $\{\alpha_w\}$ such that for any probability measure $\mu \in \mathcal{P}(\mathcal{X})$

$$\left| F(\mu) - \sum_{k=0}^{m} \sum_{\substack{w = e_{i_1} \otimes \dots \otimes e_{i_k} \\ (i_1, \dots, i_k) \in \{1, \dots, d\}^k}} \alpha_w S(\Phi(\mu))_I^w \right| < \epsilon. \tag{2.11}$$

[4] $\Phi(\mu) = \mathbb{E}S(\Pi_t\#\mu)$ where $\Pi_t\#\mu$ is the push-forward measure of μ by the mesurable map Π_t.

Proof. $\mathcal{P}(\mathcal{X})$ is compact (see proof of Theorem 2.2.5) and the image of a compact set by a continuous function is compact. Therefore, the image $K = \Phi(\mathcal{P}(\mathcal{X}))$ is a compact subset of $G\Omega_1(T(E))$. Consider a weakly continuous function $F : \mathcal{P}(\mathcal{X}) \to \mathbb{R}$. Given that Φ is injective (Theorem 2.2.2), Φ is a bijection when restricted to its range K. Because Φ is (weakly) continuous there exists a continuous function $f : K \to \mathbb{R}$ (w.r.t $\|\cdot\|_1$) such that $F = f \circ \Phi$. By Theorem 1.0.1 we know that for any $\epsilon > 0$, there exists a linear functional $\mathcal{L} : T(E) \to \mathbb{R}$ such that $\|f - \mathcal{L} \circ S\|_\infty < \epsilon$. Thus $\|F \circ \Phi^{-1} - \mathcal{L} \circ S\|_\infty < \epsilon$, implying $\|F - \mathcal{L} \circ S \circ \Phi\|_\infty < \epsilon$. □

The practical consequence of this theorem is that the complex task of learning a highly non-linear regression function $F : P(\mathcal{X}) \to \mathbb{R}$ can be reformulated as a linear regression on the signature (truncated at level m) of the pathwise expected signature (truncated at level n). The resulting SES algorithm is outlined in Alg.2, and has time complexity $\mathcal{O}(M\ell d^n(N + d^m))$, where M is the total number of groups, ℓ is the largest length across all time-series, d is the state space dimension, N is the maximum number of input time series in a single group and n, m are the truncation levels of the signature and pathwise expected signature respectively.

Algorithm 2 DR on sequential data with SES

Input: $\{(\{\mathbf{x}^{i,p}\}_{p=1}^{N_i}, y^i)\}_{i=1}^{M}$
Create array A to store M signatures of the PES.
for each group $i \in \{1, ..., M\}$ **do**
    $\Phi = \text{PES}(\{\mathbf{x}^{i,p}\}_{p=1}^{N_i})$ ▷ Using Algorithm 1
    for each time-step $k \in \{2, \dots, \ell_i\}$ **do** ▷ Compute the signature of the PES
        $A[:, i] \leftarrow A[:, i] \otimes \exp(\Phi_k - \Phi_{k-1})$ ▷ via Chen's relation
$(\alpha_0, \dots, \alpha_c) \leftarrow \text{LinearRegression}(A, (y^i)_{i=1}^{M})$
Output: Regression coefficients $(\alpha_0, \dots, \alpha_c)$

2.2.2 Method 2: a kernel-based approach (KES)

The SES algorithm is well suited to datasets containing a possibly large number MN of relatively low dimensional paths. If instead the input paths are high dimensional, it would be prohibitive to deploy SES since the number of terms in the signature increases exponentially in the dimension d of the path. To address this, in this section we construct a new kernel function $k : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ combining the expected signature with a Gaussian kernel and prove its universality to approximate weakly continuous function on probability measures on path space. The resulting kernel-based algorithm (KES) for DR on sequential data is well-adapted to the opposite data regime to the one above, i.e. when the dataset consists of few number (MN) of high dimensional data streams.

Theorem 2.2.5. Let $\mathcal{X} \subset G\Omega_1(E)$ be a compact set of paths and $\sigma > 0$. The kernel $k : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}$ defined by

$$\begin{aligned} k(\mu, \nu) &= \exp\Big(- \sigma^2 \left\| \mathbb{E}S(\mu) - \mathbb{E}S(\nu) \right\|^2_{T(E)} \Big) \\ &= \exp\Big(- \sigma^2 d_{\mathcal{G}}(\mu, \nu)^2 \Big) \end{aligned} \tag{2.12}$$

is universal, in the sense that the associated RKHS is dense in the space of weakly continuous functions from $\mathcal{P}(\mathcal{X})$ to $\mathbb{R}$, and where $d_{\mathcal{G}}(\mu, \nu)$ denotes the MMD distance between μ and ν defined in the previous chapter.

Proof. By [60, Thm. 2.2] if K is a compact metric space and H is a separable Hilbert space such that there exists a continuous and injective map $\rho : K \to H$, then for $\sigma > 0$ the Gaussian-type kernel $k_\sigma : K \times K \to \mathbb{R}$ is a universal kernel, where $k_\sigma(z, z') = \exp\Big(- \sigma^2 \left\| \rho(z) - \rho(z') \right\|^2_H \Big)$. With the metric induced by $\|\cdot\|_1$, $\mathcal{X}$ is a compact metric space. Hence the set $\mathcal{P}(\mathcal{X})$ is weakly-compact [61, Thm. 10.2]. As argued in [60, Example 1], given that $(\mathcal{X}, d_{\mathcal{X}})$—where $d_{\mathcal{X}}$ is the topology induced by $|| \cdot ||_1$—is a compact metric space, the topology describing weak convergence of (Borel) probability measures can be metrized (e.g. by the Prohorov metric $d_{P(\mathcal{X})}$). Therefore $(P(\mathcal{X}), d_{P(\mathcal{X})})$ is also a compact metric space. By Theorem 2.1.1, the expected signature $\mathbb{E}S : \mathcal{P}(\mathcal{X}) \to T(E)$ is injective and weakly continuous. Furthermore $T(E)$ is a Hilbert space with a countable basis, hence it is separable. Setting $K = \mathcal{P}(\mathcal{X}), H = T(E)$ and $\rho = \mathbb{E}S$ concludes the proof. □

When the input distributions on path space are two empirical measures

$$\delta^1 = \frac{1}{N_1} \sum_{p=1}^{N_1} \delta_{x^{1,p}} \quad \text{and} \quad \delta^2 = \frac{1}{N_2} \sum_{q=1}^{N_2} \delta_{x^{2,q}} \tag{2.13}$$

the evaluation of the kernel k in Equation (2.12) requires the ability to compute the MMD

$$d_{\mathcal{G}}(\delta^1, \delta^2)^2 = E_{11} + E_{22} - 2E_{12} \tag{2.14}$$

where

$$E_{ij} = \frac{1}{N_i N_j} \sum_{p,q=1}^{N_i, N_j} k(x^{i,p}, x^{j,q}) \tag{2.15}$$

for $i, j \in \{1, 2\}$, and where k is the PDE kernel introduced in the previous chapter, computable in practice via any numerical PDE solver.

In light of Theorem 2.2.5, DR on paths with KES can be performed via any kernel method [62, 63] available within popular machine learning libraries [64, 65, 66] using the Gram matrix computed via Algorithm 3 and leveraging the aforementioned kernel trick. When using a finite difference scheme (referred to as PDESolve in Algorithm 3) to approximate the solution of the PDE, the resulting time complexity of KES is $\mathcal{O}(M^3 + M^2 N^2 \ell^2 d)$.

Algorithm 3 Gram matrix for KES

Input: $\{x^{i,p}\}_{p=1}^{N_i},\ i = 1, \dots, M$ and $\sigma > 0$.
Initialize 0-array $G \in \mathbb{R}^{M \times M}$
for each pair of groups (i, j) such that $i \leq j$ **do**
    Initialize 0-array $K_{ij} \in \mathbb{R}^{N_i \times N_j}$
    Similarly initialize $K_{ii}, K_{jj} \in \mathbb{R}^{N_i \times N_i}, \mathbb{R}^{N_j \times N_j}$
    for p, p' in group i and q, q' in group j **do**
        $K_{ii}[p, p'] \leftarrow \text{PDESolve}(x^{i,p}, x^{i,p'})$
        $K_{jj}[q, q'] \leftarrow \text{PDESolve}(x^{j,q}, x^{j,q'})$
        $K_{ij}[p, q] \leftarrow \text{PDESolve}(x^{i,p}, x^{j,q})$
    $G[i, j] \leftarrow \text{avg}(K_{ii}) + \text{avg}(K_{jj}) - 2 \times \text{avg}(K_{ij})$
    $G[j, i] \leftarrow G[i, j]$
$G \leftarrow \exp(-\sigma^2 G)$
Output: The gram matrix G.

Remark In the case where the observed paths are assumed to be i.i.d. samples $\{x^p\}_{p=1}^N \sim \mu$ from the law of an underlying random process one would expect the bigger the sample size N, the better the approximation of μ, and therefore of its expected signature $\mathbb{E}S(\mu)$. Indeed, for any basis element w, the Central Limit Theorem yields the convergence (in distribution)

$$\sqrt{N}\Big(\mathbb{E}_{x\sim\mu}[S(x)^w] - \frac{1}{N}\sum_{p=1}^{N} S(x^p)^w\Big) \xrightarrow{\mathcal{D}} \mathcal{N}(0, \sigma_w^2) \tag{2.16}$$

as the variance $\sigma_w^2 = \mathbb{E}_{x\sim\mu}[S(x)^w S(x)^w] - (\mathbb{E}_{x\sim\mu}[S(x)^w])^2$ is always finite; in effect recall that for any path x its signature $S(x)$ is a grouplike element. Hence, $S(x)$ satisfies the shuffle-identity; in particular one has

$$S(x)^w S(x)^w = S(x)^{w \sqcup\!\sqcup w}. \tag{2.17}$$

2.2.3 Related work

Recently, there has been an increased interest in extending regression algorithms to the case where inputs are sets of numerical arrays [67, 68, 69, 70]. Here we highlight the previous work most closely related to our approach.

Deep learning techniques DeepSets [71] are examples of neural networks designed to process each item of a set individually, aggregate the outputs by means of well-designed operations (similar to pooling functions) and feed the aggregated output to a second neural network to carry out the regression. However, these models depend on a large number of parameters and results may largely vary with the choice of architecture and activation functions [72].

Kernel-based techniques In the setting of DR, elements of a set are viewed as samples from an underlying probability distribution [73, 74, 75, 45]. This framework can be intuitively summarized as a two-step procedure. Firstly, a probability measure μ is mapped to a point in an RKHS $\mathcal{H}_1$ by means of a kernel mean embedding $\Phi : \mu \to \int_{x\in\mathcal{X}} k_1(\cdot, x)\mu(dx)$, where $k_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the associated reproducing kernel. Secondly, the regression is finalized by approximating a function $F : \mathcal{H}_1 \to \mathbb{R}$ via a minimization of the form $F \approx \arg\min_{g\in\mathcal{H}_2} \sum_{i=1}^{M} \mathcal{L}(y^i, g \circ \Phi(\mu^i))$, where $\mathcal{L}$ is a loss function, resulting in a procedure involving a second kernel $k_2 : \mathcal{H}_1 \times \mathcal{H}_1 \to \mathbb{R}$. Despite the theoretical guarantees of these methods [73], the feature map $k_1(\cdot, x)$ acting on the support $\mathcal{X}$ is rarely provided explicitly, especially in the setting of non-standard input spaces $\mathcal{X} \not\subset \mathbb{R}^d$, requiring manual adaptations to make the data compatible with standard kernels. In Section 2.3 we denote by DR-k_1 the models produced by choosing k_2 to be a Gaussian-type kernel.

2.3 Experiments

We benchmark our feature-based (SES) and kernel-based (KES) methods against DeepSets and the existing kernel-based DR techniques discussed in Section 2.2.3 on various simulated and real-world examples from physics, mathematical finance, agricultural science and cybersecurity. With these examples, we show the ability of our methods to handle challenging situations where only a small number of labelled groups of multivariate time-series are available. We consider the kernel-based techniques DR-k_1 with $k_1 \in \{\text{RBF}, \text{Matern32}, \text{GA}\}$, where GA refers to the Global Alignment kernel for time-series from [76]. For KES and DR-k_1 we perform Kernel Ridge Regression, whilst for SES we use Lasso Regression. All models are run 5 times and we report the mean and standard deviation of the predictive mean squared error (MSE). The hyperparameters of KES, SES and DR-k_1 are selected by cross-validation via a grid search on the training set of each run. All the experiments can be reproduced at `https://github.com/crispitagorico/Distribution_Regression_Streams`.

For the kernel-based baselines DR-k_1, we perform Kernel Ridge regression with the kernel defined by $k(\delta^i, \delta^j) = \exp\left(-\sigma^2 \left\|\rho(\delta^i) - \rho(\delta^j)\right\|^2_{\mathcal{H}_1}\right)$, where $\rho(\delta^i) = N_i^{-1} \sum_{p=1}^{N_i} k_1(\cdot, x^{i,p})$. For $k_1 \in \{\text{RBF}, \text{Matern32}\}$, if the time-series are multi-dimensional, the dimensions are stacked to form one large vector $x \in \mathbb{R}^{d\ell}$. In Table 2.1 we report the expressions of the kernels k_1 used as baselines in all the following experiments.

RBF	$\exp(-\gamma^2 \left\|x - x'\right\|^2)$
Matern32	$(1 + \sqrt{3}\gamma^2 \left\|x - x'\right\|) \exp(-\sqrt{3}\gamma^2 \left\|x - x'\right\|)$
GA	$\exp(-\gamma \operatorname{dtw}_{1/\gamma}(x, x'))$

Table 2.1: Kernels k_1 for the kernel-based baselines. See [77] for the definition of $\operatorname{dtw}_{1/\gamma}$.

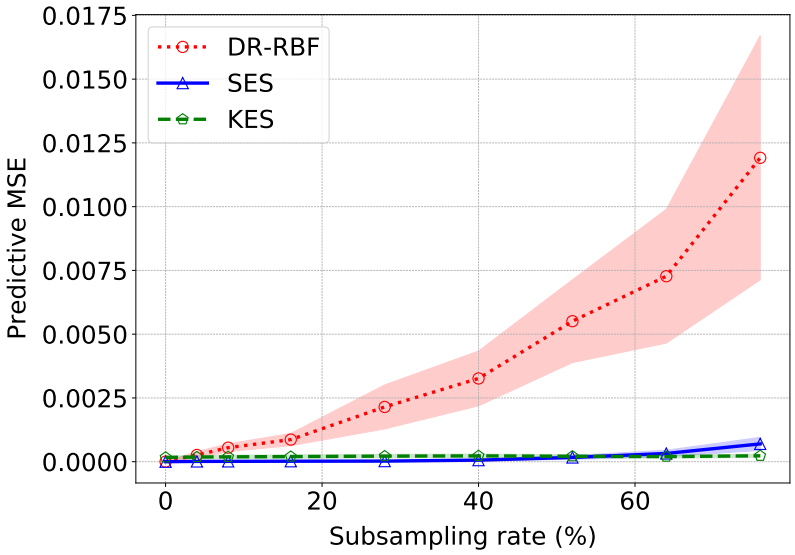


Figure 2.3: Predictive MSE at various subsampling rates for $M = 50$ circuits and $N = 15$ devices. The shaded area indicates the standard deviation.

2.3.1 A defective electronic device

We start with a toy example to show the robustness of our methods to irregularly sampled time-series. For this, we propose to infer the phase φ of an electronic circuit from multiple recordings of its voltage $v^{\varphi}(t) = \sin(\omega t)$ and current $i^{\varphi}(t) = \sin(\omega t - \varphi)$. The data consists of M simulated circuits with phases $\{\varphi^i\}_{i=1}^M$ selected uniformly at random from $[\pi/8, \pi/2]$. Each circuit is attached to N measuring devices recording the two sine waves over 20 periods at a frequency 25 points per period. We then randomly subsample the data at rates ranging from 0% to 75% independently for each defective device. As shown in Figure 2.3, the predictive performances of DR-RBF drastically deteriorate when the subsampling rate increases, whilst results for KES and SES remain roughly unchanged.

2.3.2 Inferring the temperature of an ideal gas

The thermodynamic properties of an ideal gas of N particles inside a 3-d box of volume V (cm^3) can be described in terms of the temperature T (K), the pressure P (Pa) and the total energy U (J) via the two equations of state $PV = Nk_BT$ and $U = c_V Nk_BT$, where k_B is the Boltzmann constant [78], and c_V the heat capacity.

The large-scale behaviour of the gas can be related to the trajectories of the individual particles (through their momentum = mass × velocity) by the equation $U = \frac{1}{2}\sum_{p=1}^{N} m_p|v_p|^2$. The complexity of the large-scale dynamics of the gas depends on T (see Figure 2.1) as well as on the radius of the particles. For a fixed T, the larger the radius the higher the chance of collision between the particles. We simulate $M = 50$ different gases of $N = 20$ particles each by randomly initialising all velocities and letting particles evolve at constant speed.[5] The task is to learn T (sampled uniformly at random from $[1, 1\,000]$) from the set of trajectories traced by the particles in the gas. In Figure 2.4 we report the results of two experiments, one where particles have a small radius (few collisions) and another where they have a bigger radius (many collisions).

Model	Predictive MSE [$\times 10^{-2}$]	
	r_1	$r_2 > r_1$
DeepSets	8.69 (3.74)	5.61 (0.91)
DR-RBF	3.08 (0.39)	4.36 (0.64)
DR-Matern32	3.54 (0.48)	4.12 (0.39)
DR-GA	2.85 (0.43)	3.69 (0.36)
KES	**1.31** (0.34)	**0.08** (0.02)
SES	**1.26** (0.23)	**0.09** (0.03)

Figure 2.4: Ideal gas dataset. Radii of all particles composing the simulated gases: $r_1 = 3.5 \cdot 10^{-1}(V/N)^3$ (few collisions) and $r_2 = 6.5 \cdot 10^{-1}(V/N)^3$ (many collisions).

The performance of DR-k_1 is comparable to that of KES and SES in the simpler setting. However, in the presence of a high number of collisions our models become more informative to retrieve the global temperature from local trajectories, whilst the performance of DR-k_1 drops with the increase in system-complexity. With a total number of $MN = 1\,000$ time-series of dimension $d = 7$ (after having applied time-augmentation and lead-lag transformations - see [80] for a definition of lead-lag), KES runs in 50 seconds, three times faster than SES on a 128 cores CPU.

2.3.3 Estimating parameters in a rough volatility model

Financial practitioners often model asset prices via an SDE of the form $dP_t = \mu_t dt + \sigma_t dW_t$, where μ_t is a drift term, W_t is a 1-d Brownian motion (BM) and σ_t is the volatility process [81]. This setting is often too simple to match the volatility observed in the market, especially since the advent of electronic trading [28].

Instead, we model the (rough) volatility process as $\sigma_t = \exp\{P_t\}$ where $dP_t = -a(P_t - m)dt + \nu dW_t^H$ is a fractional Ornstein-Uhlenbeck (fOU) process, with $a, \nu, m \geq 0$. The fOU is driven by a fractional Brownian Motion (fBM) W_t^H of Hurst exponent $H \in (0, 1)$, governing the

[5] We assume [79] that the environment is frictionless, and that particles are not subject to other forces such as gravity. We make use of python code from `https://github.com/labay11/ideal-gas-simulation`.

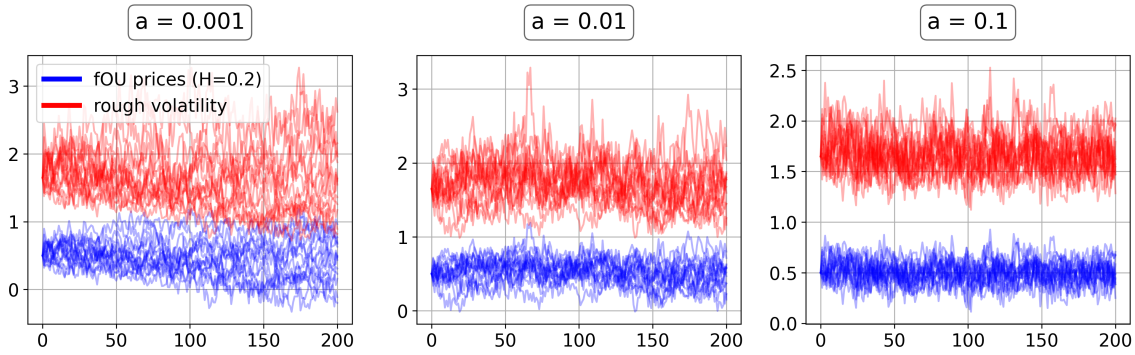


Figure 2.5: fOU sample-paths and corresponding volatilities for 3 mean-reversion parameters.

regularity of the trajectories [82].[6] In line with the findings in [28] we choose $H = 0.2$ and tackle the task of estimating the mean-reversion parameter a from simulated sample-paths of σ_t. We consider 50 mean-reversion values $\{a^i\}_{i=1}^{50}$ chosen uniformly at random from $[10^{-6}, 1]$. Examples are shown in Figure 2.5. Each a^i is regressed on a collection of $N = 20, 50, 100$ (time-augmented) trajectories $\{\hat{\sigma}_t^{i,p}\}_{p=1}^{N}$ of length 200.

Model	Predictive MSE $[\times 10^{-3}]$		
	N=20	N=50	N=100
DeepSets	74.43 (47.57)	74.07 (49.15)	74.03 (47.12)
DR-RBF	52.25 (11.20)	58.71 (19.05)	44.30 (7.12)
DR-Matern32	48.62 (10.30)	54.91 (12.02)	32.99 (5.08)
DR-GA	3.17 (1.59)	2.45 (2.73)	0.70 (0.42)
KES	**1.41** (0.40)	**0.30** (0.07)	**0.16** (0.03)
SES	**1.49** (0.39)	**0.33** (0.12)	**0.21** (0.05)

Table 2.2: Predictive MSE (standard deviation) on the rough volatility dataset. N is the number of rough volatility trajectories and $(M, d, \ell) = (50, 2, 200)$.

As shown in Table 2.2 KES and SES systematically yield the best MSE among all compared models. Moreover, the performance of KES and SES progressively improves with the number of time series in each group, in accordance with the remark at the end of Section 2.2, whilst this pattern is not observed for DR-RBF, DR-Matern32, and DeepSets. Both KES and SES yield comparable performances. However, whilst the running time of SES remains stable ($\approx$ 1 min) when MN increases from 1 000 to 5 000, the running time of KES increases from $\approx$ 1 min to 15 min (on 128 cores).

[6] We note that sample-paths of fBM are not of bounded variation, but we can assume that the interpolations obtained from market high-frequency data provide a sufficiently refined approximation of the underlying process.

2.3.4 Crop yield prediction

We also evaluate our methods on a crop yield prediction task. The challenge consists in predicting the yield of wheat crops over a region from the longitudinal measurements of climatic variables recorded across different locations of the region. We use the publicly available Eurostat dataset containing the total annual regional yield of wheat crops in mainland France—divided in 22 administrative regions—from 2015 to 2017. The climatic measurements (temperature, soil humidity and precipitation) are extracted from the GLDAS database [83], are recorded every 6 hours at a spatial resolution of $0.25° \times 0.25°$, and their number varies across regions.[7] We further subsample at random 50% of the measurements. As reported in Figure 2.6, SES and KES are the two methods which improve the most against the baseline which consists in predicting the average yield on the train set.

Model	MSE	MAPE
Baseline	2.38 (0.60)	23.31 (4.42)
DeepSets	2.67 (1.02)	22.88 (4.99)
DR-RBF	0.82 (0.22)	13.18 (2.52)
DR-Matern32	0.82 (0.23)	13.18 (2.53)
DR-GA	0.72 (0.19)	12.55 (1.74)
KES	0.65 (0.18)	12.34 (2.32)
SES	**0.62** (0.10)	**10.98** (1.12)

Figure 2.6: MSE and MAPE (mean absolute percentage error) on the Eurostat/GLDAS dataset

Interpretability of the signature features When dealing with complex data-streams, cause-effect relations between the different path-coordinates might be an essential feature that one wishes to extract from the signal. Intrinsic in the definition of the signature is the concept of iterated integral of a path over an ordered set of time indices $a < u_1 < \ldots < u_k < T$. This ordering of the domain of integration, naturally captures causal dependencies between the coordinate-paths $x^{(i_1)}, \ldots, x^{(i_k)}$.

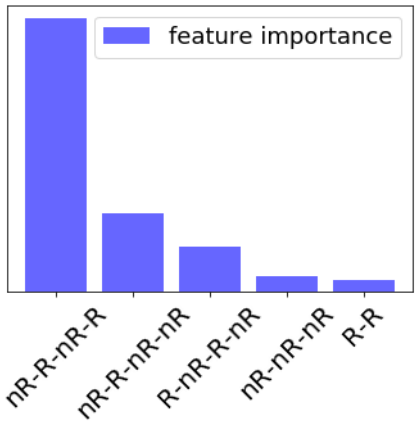


Figure 2.8: The 5 most predictive features provided by (Lasso) SES for the task of crop yield prediction.

Taking this property into account, we revisit the crop yield prediction example (see Section 2.3.4 in the main paper, and Figure 2.7) to show how the iterated integrals from the signature (of the

[7] `http://ec.europa.eu/eurostat/data/database`

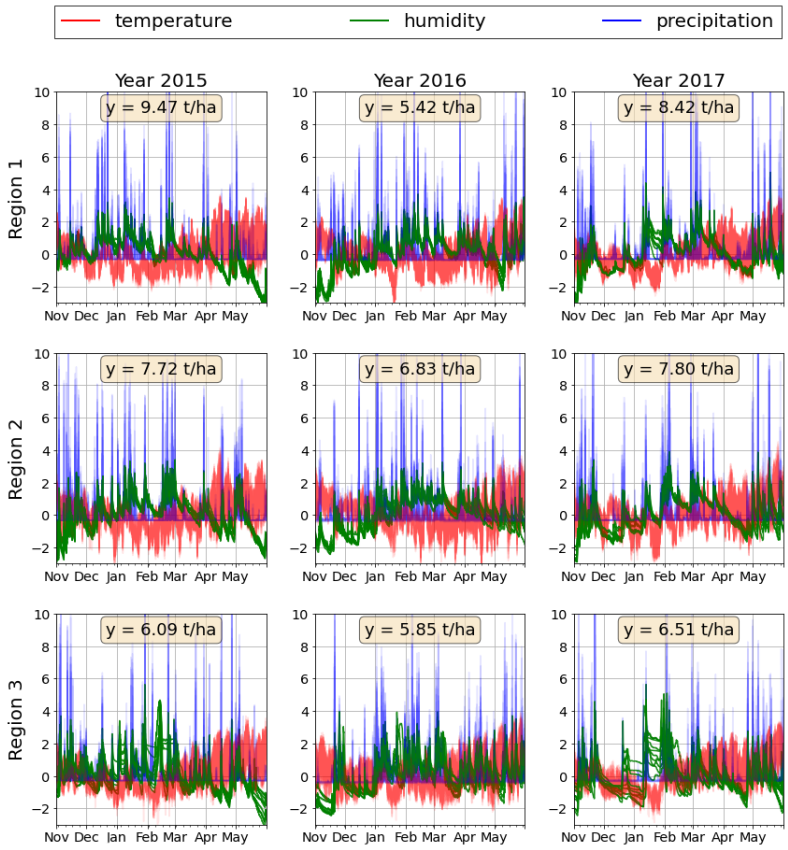


Figure 2.7: GLDAS/Eurostat dataset. Each plot shows the normalized time-series of temperature, humidity and precipitation, measured over 10 different locations across a region within a year.

pathwise expected signature) provide interpretable predictive features, in the context of DR with SES. For this, we replace the climatic variables by two distinct multi-spectral reflectance signals: 1) near-infrared (nR) spectral band; 2) red (R) spectral band [84]. These two signals are recorded at a much lower temporal resolution than the climatic variables, and are typically used to assess the health-status of a plant or crop, classically summarized by the "normalized difference vegetation index" (NDVI) [84]. To carry out this experiment, we use a publicly available dataset [85] which contains multi-spectral time-series corresponding to geo-referenced French wheat fields from 2006 to 2017, and consider these field-level longitudinal observations to predict regional yields (still obtained from the Eurostat database).[8]Instead of relying on a predefined vegetation index signal, such as the aforementionned NDVI : $t \mapsto (x_t^{nR} - x_t^R)/(x_t^{nR} + x_t^R)$, we use the raw signals in the form of 2-dimensional paths $x : t \mapsto x_t = (x_t^{nR}, x_t^R)$ to perform a Lasso DR with SES.

Chlorophyll strongly absorbs light at wavelengths around $0.67\mu m$ (red) and reflects strongly

[8]http://ec.europa.eu/eurostat/data/database

in green light, therefore our eyes perceive healthy vegetation as green. Healthy plants have a high reflectance in the near-infrared between 0.7 and $1.3\mu m$. This is primarily due to healthy internal structure of plant leaves [86]. Therefore, this absorption-reflection cycle can be seen as a good indicator of the health of crops. Intuitively, the healthier the crops, the higher the crop-yield will be at the end of the season. It is clear from Figure 2.8 that the feature in the signature that gets selected by the Lasso penalization mechanism corresponds to a double red-infrared cycle, as described above. This simple example shows how the terms of the signature are not only good predictors, but also carry a natural interpretability that can help getting a better understanding of the underlying physical phenomena.

2.3.5 Malware detection

The design and deployment of sophisticated cyber-attacks such as advanced persistent threats [87] has grown dramatically over the last few years. This has been facilitated by the appearance of new varieties of malware, and new adversary tactics and techniques, which are designed to evade existing defensive products used by enterprises such as anti-virus, firewalls and intrusion detection systems. Modern cyber security systems are generally rule-based and rely on a team of security analysts monitoring network activities and manually investigating suspected malicious activities, to determine the scope of the potential threats. This investigation phase is particularly labour-intensive. In order to defend against the influx of new malware variants and increasingly sophisticated attacks, it is imperative to develop systematic mechanisms to detect them.

One of the main challenges in creating effective and systematic malware detection systems is the complex nature of the data. The relevant datasets consist of *multimodal streams* of information, i.e. sequences of data generated by a large set of heterogeneous sources (servers, routers, workstations etc.), and representing various types of activity such as logons, file accesses, and network connections. Such data streams are often recorded at irregular intervals, span different time periods and exhibit missing observations, all aspects that make the design of systematic methods even more complicated.

A common characteristic of this data that cannot be ignored in the analysis is their *hierarchical structure*. For example, a given process may set off child processes, which themselves may spawn more children, producing a *process tree*. Here we account for how activity occurs within this tree-based structure over time. Specifically we represent the behaviour of computer processes, as observed in host-based event logs, as *streaming trees* evolving in continuous time. A representation of some selected channels of a single streaming tree is depicted in Figure 2.9. The notion of a streaming tree we introduce is a fairly generic data structure

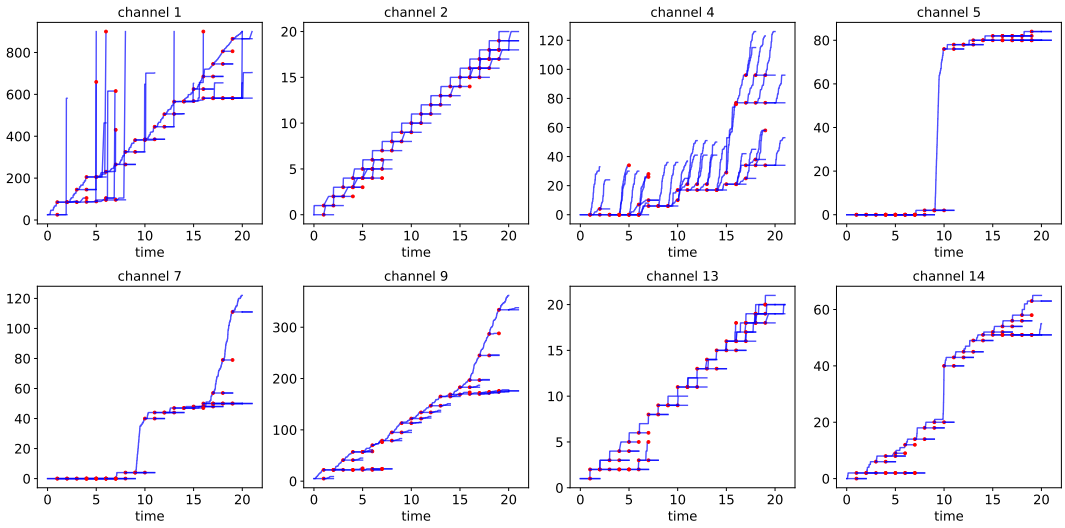


Figure 2.9: Visual representation of selected channels of one single streaming tree. Each plot represents the evolution in time of the value of a given channel of the streaming tree, on its various branches. A red dot indicates a point where the currently-tracked process sets off a child process, causing the tree to branch.

encompassing a large portion of real-world sequential data encountered in the cyber security domain. The structure of a streaming tree is significantly more complex than that of a multivariate time series, as it accounts for the hierarchy of the data; to our knowledge no systematic method has been proposed in the literature to deal with such a data structure.

Related work Many machine learning methods developed in the cyber security literature have focused on the processing of network data, with only few techniques having been designed to consume host event data as we do in this work. Among publicly-available datasets, two released by Los Alamos National Laboratory (LANL) are the most widely used [88, 89]. As far as we are aware, our work is the first to make use of the DARPA OpTC dataset [90]. Compared to earlier datasets, events in OpTC give a much more detailed summary of host activity, and contain more varied red-team behaviour. The heterogeneity of the data is one of the major bottlenecks.

On the one hand, some of the existing techniques aim to detect malicious activity from streams of events without taking into account any hierarchical structure. There has been a proliferation of work in anomaly detection for the purposes of finding malicious activity [91, 92, 93, 94, 95, 96, 97]. In particular [93] took a host-based approach and considered the authentication events in the 2015 LANL dataset using a RNN language model. Auto-encoders were used in [98] to recognise lateral movement through a network manually engineer features based on conditional probabilities defined by event counts.

On the other hand, another line of thoughts tries to detect malware or suspicious behaviour using the graph-like structure of the data, but without taking into account the sequential nature of the events. Utilised features include connections between computers on a network or relations of processes to those that they spawn. Tree and graph kernels were first introduced for malware analysis in [99], comparing the similarity of subtrees and subgraphs respectively, and achieving promising results on data obtained from a honeypot. Process trees were further studied in [100], where structures from process trees were compared against reference trees known to be non-malicious. A similar technique was used by [101], comparing star structures with templates extracted from uninfected trees.

However, none of the aforementioned approaches are designed to process complex data structures that are hierarchical and sequential simultaneously.

Streaming trees A *multivariate time series* $\mathbf{x}$ *of dimension* $d \in \mathbb{N}$ is a collection of points $x_i \in \mathbb{R}^{d-1}$ with corresponding time-stamps $t_i \in \mathbb{R}$ such that $t_0 < ... < t_n$ and defined as follows

$$\mathbf{x} = ((t_0, x_0), (t_1, x_1), ..., (t_n, x_n)) \tag{2.18}$$

A *tree* τ is defined recursively as a tuple of the form $\tau = (\mathbf{x}_\tau, F_\tau)$, where $\mathbf{x}_\tau$ is a multivariate time series and $F_\tau = (\tau_1, ..., \tau_k)$ is a (possibly empty) collection of trees. Concatenating to the time series $\mathbf{x}_\tau$ of a tree τ the values of the time series $\mathbf{x}_{\tau_1}, ..., \mathbf{x}_{\tau_k}$ of its children subtrees $\tau_1, ..., \tau_k$, and repeating this operation recursively for each children subtree τ_i, we end up with an equivalent representation of the tree τ as a collection of branches $\tau = (\mathbf{x}_\tau^1, ..., \mathbf{x}_\tau^n)$, where each branch $\mathbf{x}_\tau^i$ is a multivariate time series starting at t_0. For the sequel, it is important to keep both representations in mind.

Example 2.3.1. Consider the tree $\tau = (\mathbf{x}_\tau, ((\mathbf{x}_{\tau_1}, \varnothing), (\mathbf{x}_{\tau_2}, \varnothing)))$, where $\varnothing$ denotes the empty list and where $\mathbf{x}_\tau, \mathbf{x}_{\tau_1}, \mathbf{x}_{\tau_2}$ are the following multivariate time series of dimension d

$$\begin{aligned}
\mathbf{x}_\tau &= ((t_0, x_0), (t_1, x_1), ..., (t_n, x_n)) \\
\mathbf{x}_{\tau_1} &= ((t_{n+1}, y_0), (t_{n+2}, y_1), ..., (t_{n+i}, y_{i-1})) \\
\mathbf{x}_{\tau_2} &= ((t_{n+1}, z_0), (t_{n+2}, z_1), ..., (t_{n+j}, z_{j-1}))
\end{aligned}$$

It is easy to see that τ has two branches. Indeed, consider a first multivariate time series of dimension d

$$\mathbf{x}_\tau^1 = ((t_0, x_0), (t_1, x_1), ..., (t_{n+i}, x_{n+i}))$$

where

$$x_k = \begin{cases} x_k, & \text{if } k \leq n, \\ y_{k-n-1}, & \text{if } n+1 \leq k \leq n+i \end{cases}$$

and a second multivariate time series of dimension d

$$\mathbf{x}_\tau^2 = ((t_0, x_0), (t_1, x_1), ..., (t_{n+j}, x_{n+j}))$$

where

$$x_k = \begin{cases} x_k, & \text{if } k \leq n, \\ z_{k-n-1}, & \text{if } n+1 \leq k \leq n+j \end{cases}$$

Then, τ has the equivalent representation $\tau = (\mathbf{x}_\tau^1, \mathbf{x}_\tau^2)$ in terms of its two branches $\mathbf{x}_\tau^1, \mathbf{x}_\tau^2$.

Given a multivariate time series $\mathbf{x}$ of the form of eq. (2.18), define the *path* $X : [t_0, t_n] \to \mathbb{R}^d$ as the the continous piecewise linear interpolation of the data such that $X_{t_i} = (t_i, x_i)$. A *streaming tree* $\mathcal{T}$ is the data structure obtained by replacing all the time series appearing in the definition of a tree τ (and in all its children subtrees) by their continuous piecewise linear interpolation. Analogously to the equivalent representation of a tree in terms of its branches that we discussed above, a streaming tree $\mathcal{T}$ can also be represented as a collection of branches $\mathcal{T} = (X_\mathcal{T}^1, ..., X_\mathcal{T}^n)$ where each branch $X_\mathcal{T}^i$ is the continuous piecewise linear interpolation of the i^{th} branch $\mathbf{x}_\tau^i$ of the tree τ. The resulting branches $X_\mathcal{T}^1, ..., X_\mathcal{T}^n$ form a collection of paths with common history.

Example 2.3.2. Consider again the same tree as in Example 2.3.1, i.e. $\tau = (\mathbf{x}_\tau, ((\mathbf{x}_{\tau_1}, \varnothing), (\mathbf{x}_{\tau_2}, \varnothing)))$. We saw that τ can be represented in terms of its two branches as $\tau = (\mathbf{x}_\tau^1, \mathbf{x}_\tau^2)$. Let $X_\mathcal{T}^1 : [t_0, t_{n+i}] \to \mathbb{R}^d$ and $X_\mathcal{T}^2 : [t_0, t_{n+j}] \to \mathbb{R}^d$ be the continuous piecewise linear interpolation of $\mathbf{x}_\tau^1, \mathbf{x}_\tau^2$ respectively. Then the streaming tree $\mathcal{T}$ can be represented in terms of its two branches as $\mathcal{T} = (X_\mathcal{T}^1, X_\mathcal{T}^2)$.

Data and experiments The Operationally Transparent Cyber (OpTC) dataset [90] was released by DARPA in June 2020. It consists of data collected from an isolated network of 1000 hosts over a multi-day period. We use OpTC's *ecar* data, which records endpoint activity in an extended CAR format based on MITRE's CAR data model [102]. For example, a process creation event appears as follows (some fields omitted for clarity):


```
{"action":"CREATE",
"actorID":"437acfc7-d9ef-4c60-a108-...",
"hostname":"SysClient0201.systemia.com",
"object":"PROCESS",
"objectID":"b9d06a48-0968-4bda-b743-...",
"properties":...,
"timestamp":1569245579591}
```

Here `actorID` and `objectID` uniquely identify the parent and child processes. In events of other types, such as file and network activity, the `actorID` similarly provides a unique identifier of the process responsible for the event.

These events can be interpreted as trees in a natural way, where each branch follows the events that are associated with a particular process. At a process creation event, the tree splits into two branches: one following the continuing parent process, the other following the new child process. Specifically, we produce a tree in 23 dimensions:

- The first dimension represents time.
- The next two dimensions encode the structure of the process tree: they represent the depth in the tree, and the number of processes that this branch has set off. At a process creation event, the child branch and parent branch immediately increment in these dimensions respectively.
- We take 20 other types of events defined by (object, action) pairs, eg (MODULE, LOAD) or (FILE, DELETE). Each event type is associated a dimension. Whenever an event of this type is observed along a branch, the branch increments in the appropriate dimension.

Given the relatively high dimensionality of the paths we only test our method KES on this example, denoted `SK-Tree`. For more details see our paper [10]. The method easily extends to other sources of host data: the only requirement is that process creation events are logged (including the identity of the parent and child processes), enabling a process tree to be built. Even in the OpTC *ecar* data there is much more information available in the `properties` of each event: e.g. the identities of processes and files; source and destination ports of network connections. We currently ignore these properties, but they could be used to select a set of event types that better summarises the data.

Results The vast majority of activity in the OpTC dataset is benign: this activity continues throughout the period. However on three days, red-team attackers are active on the network, providing a source of known malicious activity that we aim to detect. The attackers' actions are detailed in a PDF document that accompanies the OpTC dataset. We take all processes that correspond to activity listed in the document, together with their descendants, and mark these processes as malicious. All the results are reproducible at `https://github.com/crispitagorico/SK-Tree`

To form the trees, we take all activity on the first day of malicious activity from host 0201 (since this host has the greatest proportion of malicious activity). We split each process into

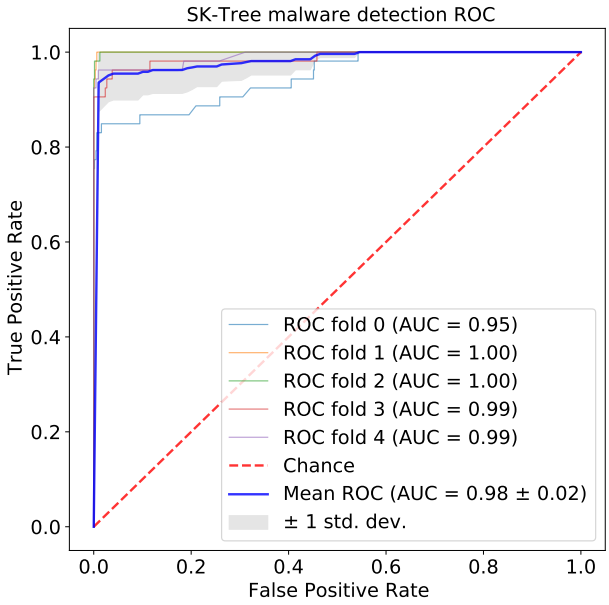


Figure 2.10: ROC evaluation of the SK-Tree binary classifier on the OpTC data

15-minute windows of activity and create a streaming tree for each. The associated label is 1 if the root process is malicious, 0 otherwise. All branches are scaled to have mean 0, sd 1. We discard any tree with fewer than 2 or more than 200 events. As a result, the final dataset contains 4 199 streaming trees. The SK-Tree binary classifier is run on random 5 fold train-test partitions and we report the mean and standard deviation of the AUROC score on Figure 2.10. The hyperparameters of SK-Tree are selected by cross-validation via a grid search on the training set of each fold. As it can be observed, we achieve a 98% AUROC score.

2.4 Conclusion

We have developed two novel techniques for distribution regression on sequential data, a task largely ignored in the previous literature. In the first technique, we introduce the pathwise expected signature and construct a universal feature map for probability measures on paths. In the second technique, we define a universal kernel based on the expected signature. We have shown the robustness of our proposed methodologies to irregularly sampled multivariate time-series, achieving state-of-the-art performances on various DR problems for sequential data. Future work will focus on developing algorithms to handle simultaneously a large number of groups of high-dimensional time-series.

Chapter 3

Neural Differential Equations

At the heart of rough path theory is the challenge of modelling the effect that a highly oscillatory signal x might generate when interacting with a control system of the form $dy_t = f(y_t)dx_t$ so as to effectively predict the trajectory of a response y. Such control systems are known in the literature as *controlled differential equations* (CDEs). For example, a CDE can model the effect the weather might have on the electricity consumption of a local community or the effect a news stream might have on stock prices traded on a stock exchange. In a simplified setting where everything is differentiable a CDE is a differential equation of the form

$$\frac{dy_t}{dt} = f\left(\frac{dx_t}{dt}, y_t\right), \quad y_0 = a \tag{3.1}$$

where $x : [0, T] \to \mathbb{R}^d$ is a given path, $a \in \mathbb{R}^n$ is an initial condition, and $y : [0, T] \to \mathbb{R}^n$ is the unknown solution.

3.1 Neural ordinary differential equations

If f did not depend on its first variable, Equation (3.1) would be a first order time-homogeneous *ordinary differential equation* (ODE), the function f would be a vector field, and y would be the integral curve of f starting at a. If instead we had $\frac{dy_t}{dt} = f(t, y_t)$, then Equation (3.1) would describe a first order time-inhomogeneous ODE, with solution y being an integral curve of a time-dependent vector field f.

Models such as residual neural networks (ResNets) match the Euler discretization of some ODE. The idea introduced in [103] consists in parametrizing the vector fields f of an ODE by a neural network f_θ so that $\frac{dy_t}{dt} = f_\theta(t, y_t)$. The main technical difficulty in training continuous-depth networks is performing backpropagation through the ODE solver. Differentiating through the operations of the forward pass is straightforward, but incurs a high memory cost and introduces additional numerical error.

The authors of [103] use the *adjoint sensitivity method* [104] to compute gradients of a loss function by solving a second augmented ODE backward in time. This approach can be used with any ODE numerical solver and scales linearly with problem size, has low memory cost, and explicitly controls numerical error.

3.1.1 The adjoint sensitivity method for backpropagation

Consider optimizing a scalar-valued loss function $L(\cdot)$, whose input is the result of an ODE solver:

$$L(y_{t_1}) = L\left(y_{t_0} + \int_{t_0}^{t_1} f_\theta(t, y_t)dt\right) = L\left(\text{ODESolve}(y_{t_0}, f_\theta, t_0, t_1)\right).$$

To optimize L we require gradients with respect to θ. By the results in [104], the quantity $a_t := \frac{\partial L}{\partial y_t}$ also satisfies another ODE:

$$\frac{da_t}{dt} = -a_t^T \frac{\partial f_\theta(t, y_t)}{\partial y_t}.$$

We can compute $\frac{\partial L}{\partial y_{t_0}}$ by another call to the ODE solver, which must be run backwards, starting from the initial value of $\frac{\partial L}{\partial y_{t_1}}$. Computing the gradients with respect to the parameters θ requires evaluating a third integral, which depends on both y_t, a_t:

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} a_t^T \frac{\partial f_\theta(t, y_t)}{\partial y_t}.$$

For a modern proof of these statements see [103, Appendix].

3.2 Controlled differential equations

In Equation (3.1) the time-inhomogeneity depends on the trajectory of x, which is said to control the problem. A controlled differential equation (CDE) provides a fairly generic way to describe how a signal x interacts with a control system of the form Equation (3.1) to produce a response y. For example, a CDE can model the interaction of an electricity signal (current, voltage) with a domestic appliance (e.g. a washing machine) to produce a response (e.g. the rotation speed or the water temperature).

In this section we present a framework allowing to learn the unknown control-response mechanism governing the interaction between a control signal and a response whose dynamics are modelled via a CDE. In other words, we are interested in learning the unknown vector field f of a CDE from a dataset of input-output pairs of vector-valued data streams $\{x(i), y(i)\}$ where the interaction between the input signal $x(i)$ and the output signal $y(i)$ is modelled by a CDE of the form

$$dy(i)_t = f(y(i)_t)dx(i)_t \tag{3.2}$$

In view of practical applications in data science, the Banach spaces V, W of the previous chapters will be replaced by $\mathbb{R}^m, \mathbb{R}^n$ respectively. Consequently, we will think of an input as a path $x : [0, T] \to \mathbb{R}^m$, of an output as a path $y : [0, T] \to \mathbb{R}^n$ and of the unknown vector field as a continuous function $f : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ to be learned.

Given an initial condition $a \in \mathbb{R}^m$, we say that x, y and f satisfy the following controlled differential equation (CDE)

$$dy_t = f(y_t)dx_t, \quad y_0 = a \tag{3.3}$$

if the following equality holds for all $t \in I$

$$y_t = a + \int_0^t f(y_s)dx_s \tag{3.4}$$

Here f is called a *vector field*, x the *control* driving the equation and y the *response* solution.

Remark 3.2.1. If y is a solution of Equation (3.3) driven by x and if $\psi : I \to I$ is reparametrization, then $y \circ \psi$ is also a solution to the same equation driven by $x \circ \psi$. In other words, solutions of CDEs are invariant to time-reparametrization.

Remark 3.2.2. The definition of f as a continuous function from $\mathbb{R}^n$ to $L(\mathbb{R}^m, \mathbb{R}^n) \simeq \mathbb{R}^{n\times m}$ makes the notation $f(y_s)dx_s$ in Equation (3.3) clear. However, there is another equivalent interpretation of f which is important to keep in mind. Let $C(\mathbb{R}^n, \mathbb{R}^n)$ be the space of continuous functions from $\mathbb{R}^n$ to $\mathbb{R}^n$. Then f can be thought of as an element of $L(\mathbb{R}^m, C(\mathbb{R}^n, \mathbb{R}^n))$, which is the view taken in rough integration (see Appendix). Adopting this point of view, f is a linear function on $\mathbb{R}^m$ with values in the space of vector fields on $\mathbb{R}^n$. To make this view compatible with Equation (3.3) it is perhaps easier to think of the CDE (3.3) using the slightly different notation $dy_t = f(dx_t)y_t$. This alternative point of view has a deep physical interpretation: at each time t the solution y describes the state of a complex system that evolves as a function of its present state y_t and of an infinitesimal external parameter dx_t controlling it. The function f transforms the infinitesimal displacement dx_t into a vector field $f(dx_t)$ that defines a direction for the trajectory of y to follow.

If one assumes that x is of bounded variation and that y and f are continuous, then the integral $\int_0^t f(y_s)dx_s$ exists for all $t \in I$ (as a classical Riemann-Stieltjes integral). If f is Lipschitz continuous then the solution is unique [2, Theorem 1.3 & 1.4].

3.3 Neural controlled differential equations

The class of data science problems which can be most naturally modelled via a CDE falls into the category known as "sequence-to-sequence" modelling, where the input x is a multivariate time series, the output y is also a multivariate time series, and the objective is to learn

the unknown functional relationship between x and y. Sometimes though, one might be interested in a simplified setting where instead of being a data stream, the output y is a single point in $\mathbb{R}^n$. In time series regression or classification for instance, the input x is a time series whilst the output y is a single vector in $\mathbb{R}^n$ (a probability vector in case of classification). We would like to emphasize that this simpler supervised learning setting can be tackled by means of a CDE model by simply considering the output $y = y_T$ as the final point of the response trajectory in equation (3.2).

Remark 3.3.1. Stochastic differential equations (SDEs) can be seen as a special class of CDEs where the control x is of the form $x_t = (t, W_t)$, and where W_t is a sample path from a d-dimensional Brownian motion.

Consider again a time series $\mathbf{x}$ as a collection of points $x_i \in \mathbb{R}^{m-1}$ with corresponding time-stamps $t_i \in \mathbb{R}$ such that $\mathbf{x} = ((t_0, x_0), (t_1, x_1), ..., (t_n, x_n))$, and $t_0 < ... < t_n$. Let $x : [t_0, t_n] \to \mathbb{R}^m$ be some interpolation of the data such that $x_{t_i} = (t_i, x_i)$. In [105] the authors use natural cubic splines. Here we will actually end up finding piecewise linear interpolation to be a more convenient choice. Let $\xi_\theta : \mathbb{R}^m \to \mathbb{R}^n$ and $f_\theta : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ be two neural networks. Let $\ell_\theta : \mathbb{R}^n \to \mathbb{R}^q$ be a linear map, for some output dimension $q \in \mathbb{N}$. Here θ is used to denote dependence on learnable parameters.

We define Z as the hidden state and y as the output of a neural CDE driven by x if

$$Z_{t_0} = \xi_\theta(t_0, x_0), \quad \text{with} \quad Z_t = Z_{t_0} + \int_{t_0}^{t} f_\theta(Z_s)dx_s \quad \text{and} \quad y_t = \ell_\theta(Z_t) \quad \text{for } t \in (t_0, t_n]. \tag{3.5}$$

That is – just like an RNN – we have an evolving hidden state Z, which is fed into a linear map to produce an output. This formulation is a universal approximator [105, Appendix B] in the sense that it can approximate uniformly well continuous functions on compact set of continuous paths of bounded variation. As mentioned in the above remark, the output may be either the time-evolving y_t or just the final y_{t_n}. This is then fed into a loss function (L^2, cross entropy, ...) and trained via stochastic gradient descent in the usual way. In this way, neural CDEs are the continuous-time analogue of an RNN, just as Neural ODEs [103], are analogous to ResNets, and provide a natural method for modelling temporal dynamics with neural networks.

Remark 3.3.2. Neural CDEs are similar to neural ordinary differential equations (Neural ODEs), as popularised by [103]. A Neural ODE is determined by its initial condition, without a direct way to modify the trajectory given subsequent observations. In contrast the vector field of a Neural CDE depends upon the time-varying data, so that the trajectory of the system is driven by a sequence of observations.

In [105] the authors assume that the driving path x in (3.5) is differentiable and let

$$g_{\theta,x}(Z,s) = f_\theta(Z)\frac{dx}{ds}(s), \tag{3.6}$$

where the product on the right hand side denotes matrix multiplication, and then note that the integral can be written as

$$Z_t = Z_{t_0} + \int_{t_0}^{t} g_{\theta,x}(Z_s, s)ds. \tag{3.7}$$

This reduces the CDE to an ODE, so that adjoint method tools for Neural ODEs may be used to evaluate and backpropagate.

Remark 3.3.3. By moving from the discrete-time formulation of an RNN to the continuous-time formulation of a Neural CDE, then every kind of time series data is put on the same footing, whether it is regularly or irregularly sampled, whether or not it has missing values, and whether or not the input sequences are of consistent length.

However, the regularity assumption of differentiability on the control x of (3.7) is very strong. It is clearly violated, for instance, for piecewise linear paths. This is not an issue if the concerned application allows to take a smooth approximation (cubic splines etc.) of the underlying data. In many applications though, the fine oscillations in the data are indicative of important information that needs to be captured. Rough volatility models from quantitative finance for instance fall in this category. Practitioners have found that sample paths from fractional Brownian motion with Hurst exponent $h < 0.5$ needs to be considered as controls in order to match the volatility observed in the market, especially since the advent of electronic trading [28].

In these situations smoothing is not appropriate, as doing so would completely remove the information carried by the fine oscillations. Furthermore, neural CDEs, as with RNNs, begin to break down for long time series. Training loss/accuracy worsens, and training time becomes prohibitive due to the sheer number of forward operations within each training epoch. This is where rough path theory becomes useful.

3.4 Neural rough differential equations

The theory of rough differential equations (RDEs), i.e. CDEs driven by rough paths, is discussed in the Appendix. Here we present directly the analogue deep learning model. The central numerial technique will be the *log-ODE method* that allows to express the solution at final time of an RDE to the solution of an ODE dependent on the log-signature of the driving path. The conversion to an ODE allows to leverage the numerical tools for neural

ODEs discussed above. In particular, experiments show that this method is particularly beneficial for long time series, that both training time and model performance of Neural CDEs are improved, and memory requirements are reduced [106].

Consider again a CDE of the form

$$dy_t = f(y_t)dx_t, \quad y_0 = a \tag{3.8}$$

where $x : [0, T] \to \mathbb{R}^m$ has bounded variation, $a \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to L(\mathbb{R}^m, \mathbb{R}^n)$ is a function with certain smoothness assumptions so that the CDE (3.8) is well posed. In order to define the log-ODE method we need to introduce the concept of log-signature of a path.

Definition 3.4.1. For $A = (a_0, a_1, \cdots) \in T(\mathbb{R}^m)$ with $a_0 > 0$, define the tensor logarithm $\log(A)$ to be the element of $T(\mathbb{R}^m)$ given by the following series:

$$\log(A) = \log(a_0) + \sum_{n=1}^{\infty} \frac{(-1)^n}{n} \left(\mathbf{1} - \frac{A}{a_0} \right)^{\otimes n} \tag{3.9}$$

where $\mathbf{1} = (1, 0, \cdots)$ is the unit element of $T(\mathbb{R}^m)$ and $\log(a_0)$ is viewed as $\log(a_0)\mathbf{1}$. We denote by $\log_N(A) \in T^N(\mathbb{R}^m)$ the truncated (at level N) tensor logarithm defined as

$$\log_N(A) = \log(a_0) + \sum_{n=1}^{N} \frac{(-1)^n}{n} \left(\mathbf{1} - \frac{A}{a_0} \right)^{\otimes n} \tag{3.10}$$

Definition 3.4.2 (**Log-signature**)**.** The log-signature of a path of bounded variation $x : [0, T] \to \mathbb{R}^m$ over the interval $[s, t]$ is defined as the logarithm of its signature over $[s, t]$

$$\log S(x)_{s,t} := \log(S(x)_{s,t}) \tag{3.11}$$

We denote by $\log S^N(x)_{s,t} = \log^N(S(x)_{s,t})$ the log-signature of x truncated at level N.

Remark 3.4.1. It turns out that the range of the truncated log-signature (for any level $N \geq 1$) has some very special algebraic structure (see Chapter 4). What is relevant in this section is that it is a vector space of dimension $\beta(m, N)$ where

$$\beta(m, N) = \sum_{k=1}^{N} \frac{1}{k} \sum_{i|k} \mu\left(\frac{k}{i}\right) m^i \tag{3.12}$$

where μ the Möbius function. Bases of this space are known as Hall bases [107], which will be extensively studied in Chapter 4. What is important to note is that $\beta(m, N) < N$, or in other words that the truncated log-signature can store the same amount of information as the truncated signature but it requires less storage memory. This is because the action of the log-signature on a path removes some redundancy that are present in the iterated integrals defining the signature. In practice, the log-signature truncated at level N can be thought of as an embedding of path $x : [0, T] \to \mathbb{R}^m$ into $\mathbb{R}^{\beta(m,N)}$.

We now describe two closely related numerical methods for solving the CDE (3.8): the Taylor method and the log-ODE method.

3.4.1 The Taylor method

It turns out that the signature appears naturally when one wants to solve CDEs. In what follows, we define the numerical scheme using iterated integrals of a path to encode information and its construction via a practical example of a linear CDE.

Let $A : \mathbb{R}^m \to L(\mathbb{R}^n, \mathbb{R}^n)$ be a bounded linear map. Here A will play the role of (linear) vector field defined with the alternative but equivalent point of view expressed in Remark 3.2.2. Let $x : [0, T] \to V$ be a continuous path of bounded variation. Consider the following system of linear CDEs

$$dy_t = Ay_t dx_t, \quad y_0 \in \mathbb{R}^n \tag{3.13}$$

$$d\phi_t = A\phi_t dx_t, \quad \phi_0 \in L(\mathbb{R}^n, \mathbb{R}^n) \tag{3.14}$$

where in the first equation the expression $Ay_t dx_t$ is to be read as $A(dx_t)(y_t)$ and in the second expression $A\phi_t dx_t$ means $A(dx_t) \circ \phi$. The solution $t \mapsto \phi_t$ to Equation (3.14) is called the flow associated to the linear equation Equation (3.13). From the flow ϕ_t and the initial condition y_0 one gets the solution y_t in the usual way

$$y_t = \phi_t(y_0) \tag{3.15}$$

The map $I_A : \Omega_1(\mathbb{R}^m) \times \mathbb{R}^n \to \Omega_1(\mathbb{R}^n)$ that to the pair (x, y_0) associates the solution y is generally called the Itô-map. We will now run an iterative procedure analogous to the Picard's iteration for ODEs in order to obtain a sequence $\{\phi_t^n : [0, T] \to L(\mathbb{R}^n, \mathbb{R}^n)\}_{n \geq 0}$ of flows, i.e. paths with values in the space of linear vector fields on $\mathbb{R}^n$. Firstly, we start by setting the first element of the sequence $\phi_t^0 = I_n$ equal to the identity on $L(\mathbb{R}^n, \mathbb{R}^n)$ and then we integrate Equation (3.14) to get

$$\phi_t^1 = I_n + \int_0^t A\phi_s^0 dx_s = I_n + \int_0^t A(dx_s) \tag{3.16}$$

Doing it again we obtain the following expression for the next element in the sequence

$$\phi_t^2 = I_n + \int_0^t A\phi_s^1 dx_s = I_n + \int_0^t A(dx_s) + \int\int_{0<s_1<s_2<t} A(dx_{s_1})A(dx_{s_2}) \tag{3.17}$$

Iterating this process we get the following expression for the n^{th} term in the sequence

$$\phi_t^n = I_n + \sum_{k=1}^{n} \int ... \int_{0<s_1<...<s_k<t} A(dx_{s_1})...A(dx_{s_k}) \tag{3.18}$$

Now, for any $k \geq 1$ the linear operator $A : \mathbb{R}^m \to L(\mathbb{R}^n, \mathbb{R}^n)$ can be extended to the operator $A^{\otimes k} : (\mathbb{R}^m)^{\otimes k} \to L(\mathbb{R}^n, \mathbb{R}^n)$ as follows

$$A^{\otimes k}(v_1 \otimes ... \otimes v_k) = A(v_k)...A(v_1) \tag{3.19}$$

which allows us to rewrite equation Equation (3.18) as

$$\phi_t^n = I_n + \sum_{k=1}^{n} A^{\otimes k} \int_{0<s_1<...<s_k<t} ... \int dx_{s_1} \otimes ... \otimes dx_{s_k} \tag{3.20}$$

where we can finally recognise the various terms of the signature of the path x (see Definition 1.0.3) and how the latter fundamentally encode the information to solve linear CDEs. The sequence of approximations $\{\phi^n\}$ for the flow ϕ provides a sequence of approximations for the solution of the CDE Equation (3.13):

$$y_t^n = y_0 + \sum_{k=1}^{n} \left(A^{\otimes k} \int_{0<s_1<...<s_k<t} ... \int dx_{s_1} \otimes ... \otimes dx_{s_k} \right) y_0 \tag{3.21}$$

It turns out that the speed at which the sequence of approximations $\{y^n\}$ converges to the true solution y of the CDE Equation (3.13) is extremely high.

Lemma 3.4.0.1. [2, Proposition 2.2] For each $k \geq 1$ we have

$$\Big| \int_{0<s_1<...<s_k<t} ... \int dx_{s_1} \otimes ... \otimes dx_{s_k} \Big| \leq \frac{||x||_{1,[0,t]}^k}{k!} \tag{3.22}$$

In particular, taking any norm $|| \cdot ||$ on bounded linear operators, the error decays factorially

$$|y_t - y_t^n| \leq \sum_{k=n+1}^{\infty} \frac{(||A|| \cdot ||x||_{1,[0,t]})^k}{k!} \tag{3.23}$$

This shows how the signature of a control x provides an extremely efficient sequence of statistics to solve any linear CDE driven by x (provided the norm of A is not too large).

This procedure can be extended to non-linear vector fields f, provided it is bounded and with bounded derivatives.

Definition 3.4.3. Let $U \subset \mathbb{R}^m$ be an open set. A function $g : U \to \mathbb{R}^n$ is called Fréchet differentiable at $x \in U$ if there exists a linear operator $A : \mathbb{R}^m \to \mathbb{R}^n$ such that

$$\lim_{\|h\| \to 0} \frac{\|g(x+h) - g(x) - Ah\|_{\mathbb{R}^n}}{\|h\|_{\mathbb{R}^m}} = 0 \tag{3.24}$$

Or equivalently such that $f(x+h) = f(x) + Ah + o(h)$. We call such an operator A the Fréchet derivative of f at x and denote it by $Df(x) = A$.

Definition 3.4.4. Let $f : \mathbb{R}^n \to L(\mathbb{R}^m, \mathbb{R}^n)$ be a bounded vector field with N bounded derivatives. We define $f^{\circ k} : \mathbb{R}^n \to L((\mathbb{R}^m)^{\otimes k}, \mathbb{R}^n)$ recursively by

$$\begin{aligned} f^{\circ(0)}(y) &:= y, \\ f^{\circ(1)}(y) &:= f(y), \\ f^{\circ(k+1)}(y) &:= D(f^{\circ k})(y)f(y), \end{aligned}$$

for $y \in \mathbb{R}^n$, where $D(f^{\circ k})$ denotes the Fréchet derivative of $f^{\circ k}$.

Given the CDE (3.8), we can use the signature of x to approximate the solution y on an interval $[s, t]$ via its truncated Taylor expansion. That is, we use

$$\text{Taylor}(y_s, f, S^N_{s,t}(x)) := \sum_{k=0}^{N} f^{\circ k}(y_s)\pi_k(S^N_{s,t}(x)), \tag{3.25}$$

as an approximation for y_t where each $\pi_k : T^N(\mathbb{R}^m) \to (\mathbb{R}^m)^{\otimes k}$ is the canonical projection.

3.4.2 The log-ODE method

Using the Taylor method (3.25), we can define the function $\widehat{f} : \mathbb{R}^n \to L(T^N(\mathbb{R}^m), \mathbb{R}^n)$ by $\widehat{f}(z) := \text{Taylor}(z, f, \cdot)$. By applying $\widehat{f}$ to the truncated log-signature of the path x over an interval $[s, t]$, we can define the following ODE on the interval $[0, 1]$

$$\frac{dz}{du} = \widehat{f}(z)\log S^N(x)_{s,t}, \qquad z(0) = y_s \tag{3.26}$$

Then the log-ODE approximation of y_t is defined as

$$\text{LogODE}(y_s, f, \log S^N(x)_{s,t}) := z(1). \tag{3.27}$$

We assume that f is of class $Lip(\gamma)$ for some $\gamma \geq 1$ (see Appendix for a definition). By [108, Lemma 15] there exists a universal constant $C_{p,\gamma}$ depending only on the p-variation of x and γ such that

$$\|y_t - z_1\| \leq C_{p,\gamma}\|f\|^{\gamma}_{\text{Lip}(\gamma)}\|x\|^{\gamma}_{p\text{-var};[s,t]}, \tag{3.28}$$

where $\|\cdot\|_{p\text{-var};[s,t]}$ is the p-variation norm.

Remark 3.4.2. Our assumptions on f ensure that $z \mapsto \widehat{f}(z)\log S^N(x)_{s,t}$ is either globally bounded and Lipschitz continuous or linear. Hence both the Taylor and log-ODE methods are well defined. At the moment of writing, the vector field f of a neural CDE is modelled via a generic neural network as outlined above. However, in theory, extra constraints on f should be imposed in order to ensure that the CDE is well-posed. This is a research area that is currently being explored.

Example 3.4.1 (**The "increment-only" log-ODE method**). When $N = 1$, the ODE (3.26) becomes

$$\frac{dz}{du} = f(z)(x_t - x_s), \qquad z(0) = y_s. \tag{3.29}$$

Therefore we see that this "increment-only" log-ODE method is equivalent to driving the original CDE (3.8) by a piecewise linear approximation of the control path x.

We now explain how the log-ODE method can be used to train a Neural CDE. Recall that we observe some time series $\mathbf{x} = ((t_0, x_0), (t_1, x_1), ..., (t_k, x_k))$, and have constructed a piecewise linear interpolation $x\colon [t_0, t_k] \to \mathbb{R}^m$ such that $x_{t_i} = (t_i, x_i)$.

We now pick points r_i such that $t_0 = r_0 < r_1 < \cdots < r_\ell = t_k$. In principle these can be variably spaced but in practice we will typically space them equally far apart. The total number of points ℓ should be much smaller than k.

We now replace (3.6) with the piecewise

$$\widehat{g}_{\theta,x}(Z, s) = \widehat{f}_\theta(Z)\frac{\log S^N(x)_{r_i, r_{i+1}}}{r_{i+1} - r_i} \quad \text{for } s \in [r_i, r_{i+1}), \tag{3.30}$$

where $\widehat{f}_\theta : \mathbb{R}^n \to \mathbb{R}^{n \times \beta(m,N)}$ is an arbitrary neural network, and the right hand side denotes a matrix-vector product between $\widehat{f}_\theta$ and the log-signature. Equation (3.7) then becomes

$$Z_t = Z_{t_0} + \int_{t_0}^{t} \widehat{g}_{\theta,x}(Z_s, s)ds \tag{3.31}$$

This may now be solved as a (neural) ODE using standard ODE solvers [103].

Remark 3.4.3. Suppose we chose $r_i = t_i$ and $r_{i+1} = t_{i+1}$. Then the log-signature term is

$$\frac{\log S^N(x)_{t_i, t_{i+1}}}{t_{i+1} - t_i} \tag{3.32}$$

The log-signature truncated at level 1 is just the increment of the path over the interval, and so this becomes

$$\frac{\Delta x_{[t_i, t_{i+1}]}}{t_{i+1} - t_i} = \frac{dx}{dt}\Big|_{t=t_i} \quad \text{for } s \in [t_i, t_{i+1}), \tag{3.33}$$

that is to say the same as obtained via the simple method in [105] using linear interpolation.

Training a neural CDE via the log-ODE method is straightforward to implement using pre-existing tools [103]. There are standard libraries available for computing the log-signature transform; we use signatory [109]. Then, as Equation (3.31) is an ODE, it may be solved directly using tools such as `torchdiffeq` [110].

For an extensive discussion of the computational details and various experiments on time series modelling (with series of length up to 17k) see our paper [106]. The experiments have not been included in this thesis because they have been mainly carried out by my collaborator James Morrill.

3.5 Conclusion

We introduced neural RDEs as an approach to continuous-time time series modelling. These extend Neural CDEs, driving the hidden state not by point evaluations but by log-signatures over short intervals of the underlying time series or control path. Our Neural RDEs may still be solved via ODE methods, and thus retain both adjoint backpropagation and continuous dynamics. As they additionally reduce the effective length of the control path, we observe substantial practical benefits in applying Neural RDEs to long time series

Chapter 4

Structure theorems for streamed information

4.1 Introduction

This final chapter is algebraic; it is about the structure of the shuffle algebra, that is the algebra of linear functionals on grouplike elements with point-wise multiplication. In a different world it is also about the space of continuous functions acting on streams. The core results are structure theorems that provide, in different ways and via different algebraic operations, a (possibly unique) factorisation for streamed information loosely analogous to the unique prime factorization for integers. Before delving into the details, we provide a road map to clarify our high-level objectives.

4.1.1 The connection with streamed information

It is not too much to accept that a) on some very fine scale, most instances of streamed information can be conceptually modelled via a path $\gamma : [s,t] \to V$ with values in some Banach space V and b) that γ can be better understood by looking at a potential response $\tau : [s,t] \to W$, with values in some Banach space W, caused by its interaction with some control system of the form[1] $d\tau_u = \psi(\tau_u)d\gamma_u$, rather than by looking at its values at any given time [111]. In this case, γ is said to "drive" the system.

Decades of advances in the study of differential equations have provided tools to solve such control systems (as long as the vector field ψ is smooth enough) driven by less and less regular paths, starting from the classical case of paths of bounded variation, and culminating with the case of extremely irregular geometric rough paths [112]. In all cases, a solution is usually

[1] Here $\psi(\cdot)$ is a continuous linear map from W into a space of vector fields on V sufficiently regular, when contrasted to γ, that the equation $\tau_t - \tau_s = \int_s^t \psi(\tau_u)d\gamma_u$ is well defined with a unique solution.

found by Picard iteration which yields a quickly converging sequence of approximations defined in terms of the iterated integrals of the driving path.

As extensively discussed in previous chapters, the sequence of iterated integrals of a path has a compact representation as a sequence of tensors within the algebra $T((V))$ of tensor series over V, and is known as the signature of a path. This non-commutative exponential is obtained by evaluating the solution of the controlled differential equation $dS_u = S_u \otimes d\gamma_u$ with $S_s = (1, 0, 0, \ldots) \in T((V))$ and where $\otimes$ is the tensor product [2].

If two continuous paths have the same signature then they are the same path up to a very specific form of generalised re-parametrization known as treelike equivalence [113, 114]. We call the equivalence classes of this equivalence relation the set of unparametrized paths and denote it by $\Omega(V)$.

The image of unparametrized paths $\Omega(V)$ by the signature forms a Lie group $(G, \otimes)$ embedded in $T((V))$; its Lie algebra is the free Lie algebra [2]. Elements of the shuffle algebra (the algebric dual of $T((V))$ equipped with the shuffle product $\sqcup\!\sqcup$), when restricted to act on G, form a unital algebra that separates points [2]. As discussed in Chapter 1, by the Stone-Weierstrass theorem, for any compact set C of unparametrized paths, the set of linear functionals on signatures of paths from C is dense in the set of continuous real-valued functions on C.

Our goal is to identify explicit algebraic bases for this space of functions. It is clear that evaluating a function on a particular stream has a cost. After all, such evaluation involves "measuring" the stream in some way. However, once we have evaluated $f(\gamma)$ and $g(\gamma)$, then the cost of additionally evaluating $f(\gamma)g(\gamma)$ is a simple scalar multiplication that can be carried out without referring back to γ.

In this way we see that identifying a basis for this space of functions splits the evaluation process into two parts: a) evaluating the basis elements on the underlying data γ and then b) evaluating a unique polynomial function in these expensive but informative precomputed basis elements in order to deliver the desired function evaluation without further reference to the original stream γ.

4.1.2 The main results

Assume that V is a d-dimensional Banach space with $d < +\infty$ with a canonical basis. Then a stream γ with values in V is made of d real-valued streams $(\gamma^1, \ldots, \gamma^d)$ called its coordinate channels. If we consider two of these channels γ^i, γ^j, then we can compute the integral of

one against the other $\int \gamma_u^i d\gamma_u^j$, to form a new real-valued channel of information about our original stream γ.

We can repeat this process following specific integration patterns outlined by binary planar rooted trees with leaves labelled in a set of letters $\{1, \ldots, d\}$ indexing the channels of γ. Doing so generates what we call a pure integral. Repeating this process along the structure of a Hall tree gives rise to what we call a Hall integral. A Hall area is similarly defined by replacing the integral by its antisymmetrization $\int \gamma_u^i d\gamma_u^j - \int \gamma_u^j d\gamma_u^i$. If the tree is not Hall but generic, then we call the resulting object a pure area. These definitions are intentionally kept informal for now, and will be made formal in the following section.

The two main contributions of this chapter are:

1. to provide a more direct proof of the main result in [115] stating that polynomials in pure areas generate the shuffle algebra (Theorem 4.4.1)
2. to provide a more direct proof of the main result in [116] reported without proof also in [117, 118] stating that that polynomials in Hall integrals uniquely generate the shuffle algebra (Theorem 4.5.1)

These results allow to split functions on streams into two components: a) a first that engages with the underlying stream, systematically extracts and packages the relevant information into atomic objects (pure areas, Hall integrals) whilst removing what's irrelevant; b) a second that evaluates polynomials in these features without reference to the original stream. As a consequence of the density argument about linear functions on the signature mentioned above, polynomial functions in these atomic objects (pure areas, Hall integrals) approximate uniformly well any continuous function on a compact set of unparametrized paths.

4.2 Background

First we remind the reader in very terse form of the general collection of objects about which we write. The reader can find much more by looking in Bourbaki [119] or (and we will follow this for the results we need) Reutenauer [120]. We hope the chapter is self contained, and cites what is needed.

4.2.1 Words, tensors and series

The starting point will be a finite alphabet A of d letters.

Definition 4.2.1. A word on the alphabet A is a finite sequence of letters from A, including the empty sequence, called the empty word and denoted by e. We denote by W_A the set of

all words, including the empty word. W_A with the concatenation product is a monoid, called the free monoid over A. The length $|w|$ of a word $w \in W_A$ is the number of letters in w.

Definition 4.2.2 (**The tensor algebra** $T(A)$)**.** We denote by $T(A)$ the free associative $\mathbb{R}$-algebra over A with the concatenation product.

Note that W_A is contained in $T(A)$ as a submonoid. $T(A)$ has W_A as a canonical basis. The words of length greater than n span an ideal, and the quotient of $T(A)$ by this ideal is often referred to as the truncated tensor algebra $T^{(n)}(A)$. $T(A)$ is also a Lie algebra with Lie bracket $[x, y] = xy - yx$ for $x, y \in T(A)$.

Definition 4.2.3 (**The free Lie algebra** $L(A)$)**.** Denote by $L(A)$ the Lie algebra generated by A in $T(A)$, i.e. the intersection of all Lie algebras in $T(A)$ containing A.

Lemma 4.2.0.1. [120, Theorem 0.5] $L(A)$ is the free Lie algebra over A.

Definition 4.2.4. A series on A over $\mathbb{R}$ is a infinite linear combination of words. The associative graded algebra of all series is denoted by $T((A))$.

Note that $T(A)$ is a subalgebra of $T((A))$. Those elements in $T(A)$ or $T((A))$ that are, at each truncated level n, in $L(A)$ are known as Lie elements (and they are a Lie algebra). Those elements in $T(A)$ or $T((A))$ that are, at each truncated level, exponentials of Lie elements, or equivalently, whose truncated logarithm is in L(A), are known as grouplike elements (and they are a group); log and exp provide a one to one correspondence between group-like elements and Lie elements.

4.2.2 The shuffle algebra as the dual space of the tensor algebra

There is a natural duality between $T(A)$ and $T((A))$ given by the pairing $T(A) \times T((A)) \to \mathbb{R}$ defined as follows:

$$(P, S) \mapsto \sum_{w \in W_A} P^w S^w \tag{4.1}$$

where P^w, S^w denote the coefficients in front of the word w in P, S respectively. Note that this sum is finite because P is a finite linear combination of words.

With this pairing $T((A))$ can be identified with the dual space of $T(A)$. When restricted to $T(A)$ this pairing gives a scalar product on $T(A)$ with W_A as orthonormal basis.

Definition 4.2.5 (**The shuffle algebra** $\mathrm{Sh}(A)$)**.** We denote the space dual to $T((A))$ by $\mathrm{Sh}(A)$ and refer to it as the shuffle algebra. The basis for $\mathrm{Sh}(A)$ dual to W_A is denoted by W_A^*.

$\mathrm{Sh}(A)$ is a commutative algebra, called the shuffle algebra, when equipped with the shuffle product $\sqcup\!\sqcup$ that we define next.

Definition 4.2.6 (**The shuffle product** $\sqcup\!\sqcup$)**.** Let u and v be non-empty words in W_A; the shuffle product $u \sqcup\!\sqcup v$ is defined by the following relations

1. $u \sqcup\!\sqcup e = e \sqcup\!\sqcup u = u$,
2. $au \sqcup\!\sqcup bv = a(u \sqcup\!\sqcup bv) + b(au \sqcup\!\sqcup v)$.

Remark 4.2.1. Although we do not equate W_A and W_A^*, we do allow implicit and free conversion of letters and words according to context. If u, v are words with letters in A then in the expression (u, v), the word u is in W_A^*, the word v is in W_A, and the action of u on v is zero unless u and v are the same word.

4.2.3 Magmas of trees

The final ingredient we need to introduce is $M(A)$, the free magma over A.

Definition 4.2.7 (**The free magma** $M(A)$)**.** The free magma $M(A)$ is the minimal set of trees satisfying: i) $A \subset M(A)$ and ii) if $t', t'' \in M(A)$ then $(t', t'') \in M(A)$.

There is a natural correspondence between the basis for $T(A)$ comprising words W_A and its PBW basis. This correspondence is realised through the foliage map that we define next.

Definition 4.2.8. The foliage map $f : M(A) \to W_A$ defined on a letter $a \in A$ as $f(a) = a$ and on a tree $t = (t_1, t_2) \in M(A)$ as $f(t) = f(t_1)f(t_2)$ where the product is concatenation.

Remark 4.2.2. As noted in [120], $M(A)$ can be equivalently identified with the set of binary, planar, rooted trees with leaves labelled in A. For a given element $t \in M(A)$ we will refer to the collection of letters appearing in its leaves as its foliage.

Definition 4.2.9. Given a tree $t \in M(A)$ we its degree $d(t)$ as: $d(t) = 1$ if $t \in A$, otherwise if $t = (t_1, t_2) \in M(A)$ then $d(t) = d(t_1) + d(t_2)$.

This concludes the presentation of the background material.

4.3 Operators on the shuffle algebra: half shuffle and area

Before presenting the three structure theorems announced in the introduction we will discuss another basic operator on the shuffle algebra and associated quantities which will serve as building blocks for the results that follow.

4.3.1 The half shuffle product and connections with calculus

Definition 4.3.1 (**half shuffle**)**.** The (left) half shuffle product $\prec : \mathrm{Sh}(A) \times \mathrm{Sh}(A) \to \mathrm{Sh}(A)$ is a bi-linear form initially defined on basis elements. Let u and v be two words in W_A^*. If u is the empty word e then $u \prec v := 0$. If u is nonempty, then let the letters in u be given by $u = u_1 ... u_n$ and define

$$u \prec v := u_1((u_2 \dots u_n) \sqcup\!\sqcup v). \tag{4.2}$$

In particular, if u is a nonempty word then $u \prec e = u$.

We note that any binary operator defined on words over A automatically extends to the magma $M(A)$. In particular this leads to the next definition.

Definition 4.3.2. The extension of the half shuffle product to the operator $\prec : M(A) \to \mathrm{Sh}(A)$ is defined recursively as follows

1. if $t \in A \subset M(A)$ is a letter, then $\prec(t) = t \in \mathrm{Sh}(A)$,
2. otherwise, if $t = (t_1, t_2) \in M(A)$, then $\prec(t) = (\prec(t_1)) \prec (\prec(t_2))$.

The half shuffle product satisfies many algebraic relations that correspond to fundamental calculus operations.

Remark 4.3.1 (**fundamental theorem of calculus**)**.** Let $e \in \mathrm{Sh}(A)$ be the empty word, then for any $x \in \mathrm{Sh}(A)$

$$x \prec e = x - (x, e)e. \tag{4.3}$$

This is an algebraic version of the statement

$$\int_0^{\cdot} 1 \; df = f(.) - f(0). \tag{4.4}$$

Remark 4.3.2 (**product rule**)**.** It is easy to check from the definition of half shuffle that for any $x, y \in \mathrm{Sh}(A)$ the following relation is satisfied

$$(x \sqcup\!\sqcup y) \prec e = x \prec y + y \prec x. \tag{4.5}$$

This is an algebraic version of the statement

$$\int_0^{\cdot} 1 \; d(fg) = \int_0^{\cdot} g df + \int_0^{\cdot} f dg. \tag{4.6}$$

Remark 4.3.3 (**integration by parts**)**.** Rewriting the left hand side in the product rule gives that for any $x, y \in \mathrm{Sh}(A)$

$$x \sqcup\!\sqcup y - (x, e)(y, e)e = x \prec y + y \prec x. \tag{4.7}$$

This is an algebraic version of the statement

$$f(\cdot)g(\cdot) - f(0)g(0) = \int_0^{\cdot} g df + \int_0^{\cdot} f dg. \tag{4.8}$$

Remark 4.3.4 (**chain rule**)**.** It follows directly from the definition of half shuffle that for any $x, y, z \in \mathrm{Sh}(A)$

$$x \prec (y \sqcup\!\sqcup z) = (x \prec y) \prec z. \tag{4.9}$$

This is an algebraic version of the following statement

$$\int_0^{\cdot} f g dh = \int_0^{\cdot} f d(\int_0^{\cdot} g dh). \tag{4.10}$$

The identity of the next lemma, known in the literature as the Zinbiel or Chronological identity, is a direct application of the chain rule and integration by parts.

Lemma 4.3.0.1 (**Zinbiel identity**)**.** For any $x, y, z \in \mathrm{Sh}(A)$ such that $(x, e) = (y, e) = (z, e) = 0$ the following identity holds

$$(x \prec y) \prec z = x \prec (y \prec z) + x \prec (z \prec y). \tag{4.11}$$

Proof. By the chain rule and integration by parts we deduce that

$$\begin{aligned}(x \prec y) \prec z &= x \prec (y \sqcup\!\sqcup z) \\ &= x \prec (y \prec z + z \prec y + (y, e)(z, e)e) \\ &= x \prec (y \prec z) + x \prec (z \prec y)\end{aligned}$$

as required. □

We now turn to the second fundamental product considered in this chapter.

4.3.2 Areas and identities

The area operator is defined as the antisymmetrisation of the half shuffe product.

Definition 4.3.3 (**area**)**.** The operator area : $\mathrm{Sh}(A) \times \mathrm{Sh}(A) \to \mathrm{Sh}(A)$ is the bi-linear form defined on any two elements x, y of $\mathrm{Sh}(A)$ by

$$\mathrm{area}(x, y) := x \prec y - y \prec x. \tag{4.12}$$

Remark 4.3.5 (**left/right areas**)**.** In this chapter area is defined as the antisymmetrization of the left half shuffle. In [115], the right half shuffle is introduced and area is defined as the as the antisymmetrization of the right half shuffle. Although closely connected, these are

not identical. The left half shuffle is consistent with [120] and matches the conventions for Hall basis used there. The *left* half shuffle occurs in the dual PBW basis defined using this convention for defining Hall bases. The right half shuffle is more easily consistent with the convention used in integration that the integrand is on the left and the integrator is on the right. The reversed order of terms within equations (4.9) and (4.10) reflect this dissonance. The proofs of our main results imply equivalent results with the other definition of area, by reversing everything.

Definition 4.3.4. The extension of area to the operator $\mathsf{area} : M(A) \to \mathrm{Sh}(A)$ is defined recursively as follows

1. if $t \in A \subset M(A)$ is a letter, then $\mathsf{area}(t) = t \in \mathrm{Sh}(A)$
2. Otherwise, if $t = (t_1, t_2) \in M(A)$, then $\mathsf{area}(t) = \mathrm{area}(\mathsf{area}(t_1), \mathsf{area}(t_2))$

Contrary to the Lie bracket on $T(A)$, area doesn't satisfy the Jacobi identity. However, it satisfies the following two non-trivial identities that will be leveraged to prove the first structure theorem in the next section. The proofs directly follow from definition and the above lemmas.

Lemma 4.3.0.2 (**shuffle-pullout identity**)**.** For any triple $x, y, z \in \mathrm{Sh}(A)$ the following relation is satisfied

$$\begin{aligned} 3\,\mathrm{area}(z, x \sqcup\!\sqcup y) = x \sqcup\!\sqcup \mathrm{area}(z, y) + y \sqcup\!\sqcup \mathrm{area}(z, x) - x \sqcup\!\sqcup y \sqcup\!\sqcup z \\ + \mathrm{area}(\mathrm{area}(z, y), x) + \mathrm{area}(\mathrm{area}(z, x), y). \end{aligned}$$

Proof. Let $x, y, z \in \mathrm{Sh}(A)$. We first note that

$$\begin{aligned} x \sqcup\!\sqcup y \sqcup\!\sqcup z &= x \prec (y \sqcup\!\sqcup z) + (y \sqcup\!\sqcup z) \prec x \\ &= x \prec (y \sqcup\!\sqcup z) + (y \prec z) \prec x + (z \prec y) \prec x \\ &= x \prec (y \sqcup\!\sqcup z) + y \prec (x \sqcup\!\sqcup z) + z \prec (x \sqcup\!\sqcup y) \end{aligned}$$

Using this relation to establish the 4^{th} equality below respectively, we have

$$\begin{aligned}
3\,\mathrm{area}(z, x \sqcup y) + x \sqcup y \sqcup z &= 3z \prec (x \sqcup y) - 3(x \sqcup y) \prec z + x \sqcup y \sqcup z \\
&= 3z \prec (x \sqcup y) - 3(x \prec y) \prec z - 3(y \prec x) \prec z + x \sqcup y \sqcup z \\
&= 3z \prec (x \sqcup y) - 3x \prec (y \sqcup z) - 3y \prec (x \sqcup z) + x \sqcup y \sqcup z \\
&= 4z \prec (x \sqcup y) - 2x \prec (y \sqcup z) - 2y \prec (x \sqcup z) \\
&= 2z \prec (x \sqcup y) - 2x \prec (y \sqcup z) + 2z \prec (x \sqcup y) - 2y \prec (x \sqcup z) \\
&= 2(z \prec x) \prec y - 2(x \prec z) \prec y + 2(z \prec y) \prec x - 2(y \prec z) \prec x \\
&= 2\,\mathrm{area}(z, x) \prec y + 2\,\mathrm{area}(z, y) \prec x \\
&= x \sqcup \mathrm{area}(z, y) + y \sqcup \mathrm{area}(z, x) + \mathrm{area}(\mathrm{area}(z, y), x) \\
&\quad + \mathrm{area}(\mathrm{area}(z, x), y),
\end{aligned}$$

where the last equality is derived from the relation $x \prec y = 1/2(\mathrm{area}(x, y) + x \sqcup y)$. □

Lemma 4.3.0.3 (**area-Jacobi identity**)**.** For any triple $x, y, z \in \mathrm{Sh}(A)$ the following relation is satisfied

$$\begin{aligned}
&\mathrm{area}(\mathrm{area}(x, y), z) + \mathrm{area}(\mathrm{area}(y, z), x) + \mathrm{area}(\mathrm{area}(z, x), y) \\
&= -x \sqcup \mathrm{area}(y, z) - y \sqcup \mathrm{area}(z, x) - z \sqcup \mathrm{area}(x, y)
\end{aligned}$$

Proof. For any $x, y, z \in \mathrm{Sh}(A)$

$$\begin{aligned}
&x \sqcup \mathrm{area}(y, z) + y \sqcup \mathrm{area}(z, x) + z \sqcup \mathrm{area}(x, y) \\
&= x \prec \mathrm{area}(y, z) + y \prec \mathrm{area}(z, x) + z \prec \mathrm{area}(x, y) \\
&= \frac{1}{2}\,\mathrm{area}(x, \mathrm{area}(y, z)) + \frac{1}{2} x \sqcup \mathrm{area}(y, z) + \frac{1}{2}\,\mathrm{area}(y, \mathrm{area}(z, x)) \\
&+ \frac{1}{2} y \sqcup \mathrm{area}(z, x) + \frac{1}{2}\,\mathrm{area}(z, \mathrm{area}(x, y)) + \frac{1}{2} z \sqcup \mathrm{area}(x, y)
\end{aligned}$$

where the first equality is obtained using the relation $x \sqcup y = x \prec y + y \prec x$ and the chain rule $x \prec (y \sqcup z) = (x \prec y) \prec z$, and the second equality is derived again from the relation $x \prec y = 1/2(\mathrm{area}(x, y) + x \sqcup y)$. Rearranging the terms of the above equation one gets the identity of the lemma. □

4.4 Polynomials in pure areas generate the shuffle algebra

Theorem (4.4.1) is already presented in [115]. Our proof is significantly more direct and based on induction.

Definition 4.4.1 (**pure area**)**.** An element x of $\mathrm{Sh}(A)$ is a **pure area** if it is in the image under area of the magma. That is to say, there exists a tree $t \in M(A)$ so that $x = \mathsf{area}(t)$.

Definition 4.4.2. A **shuffle monomial** of shuffle-degree n is the shuffle product of n pure areas

$$A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_n. \tag{4.13}$$

The empty monomial e has shuffle-degree 0. A shuffle polynomial of shuffle-degree n is a non-degenerate linear combination of such shuffle monomials. Its shuffle-degree is the maximal shuffle-degree of the monomials in the expression.

The sequence defined in the following lemma will play a role in what follows.

Lemma 4.4.0.1. The sequence of negative rationals $\beta_k := -(k-1)/(k+1)$ with $k \geq 1$ is monotone decreasing to -1 and satisfies the following recursion

$$\beta_1 = 0, \;\; \beta_k = \frac{\beta_{k-1} - 1}{\beta_{k-1} + 3}. \tag{4.14}$$

Exploiting the identities we introduced in the previous section we give a short and direct proof of the main result in [115], namely that polynomials in pure areas generate the shuffle algebra $\mathrm{Sh}(A)$.

Theorem 4.4.1. [115, Corollary 5.6] Any element in $\mathrm{Sh}(A)$ can be written as a shuffle polynomial in pure areas $\{\mathsf{area}(t) \mid t \in M(A)\}$.

Before reproving the theorem we establish the following fundamental re-writing rule that allows one to rewrite the area of a shuffle polynomial in pure areas with a single pure area as a new shuffle polynomial in pure areas, and provides an explicit expression for the monomial of highest shuffle-degree. The proof will crucially depend on both lemmas 4.3.0.2, 4.3.0.3.

Theorem 4.4.2. For any $n \geq 1$ and any n pure areas $A_1, \ldots, A_n$ and an additional pure area A, the following relation holds

$$\mathrm{area}(A, A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_n) = \beta_n A \sqcup\!\sqcup A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_n + Q \tag{4.15}$$

where Q is a shuffle polynomial in pure areas of shuffle-degree at most n.

Proof. We prove the statement (4.15) by induction on n. If $n = 1$ then the statement is trivially true, with $\beta_1 = 0$ and $Q = \mathrm{area}(A_1, A)$.

Suppose the statement (4.15) holds for any $n < k$. Consider k pure areas $A_1, \ldots, A_k$ and an additional pure area A. We recall that the shuffle product $\sqcup\!\sqcup$ is associative and commutative

on $\mathrm{Sh}(A)$. By the shuffle-pullout identity we have

$$\begin{aligned}3\,\mathrm{area}(A, A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) &= A_1 \sqcup\!\sqcup \mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) \\ &\quad + A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k \sqcup\!\sqcup \mathrm{area}(A, A_1) \\ &\quad - A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k \sqcup\!\sqcup A \\ &\quad + \mathrm{area}(\mathrm{area}(A, A_1), A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) \\ &\quad + \mathrm{area}(\mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k), A_1).\end{aligned}$$

By induction ($n = k - 1$) we have that

$$\begin{aligned}A_1 \sqcup\!\sqcup \mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) &= A_1 \sqcup\!\sqcup (\beta_{k-1} A \sqcup\!\sqcup A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k + Q_1') \\ &= \beta_{k-1} A \sqcup\!\sqcup A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k + A_1 \sqcup\!\sqcup Q_1'\end{aligned}$$

where $A_1 \sqcup\!\sqcup Q_1'$ is a shuffle-polynomial of shuffle-degree k. By definition $\mathrm{area}(A, A_1)$ is a pure area and so

$$Q_2' := A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k \sqcup\!\sqcup \mathrm{area}(A, A_1)$$

is a shuffle monomial of shuffle-degree k. Similarly, the induction hypothesis implies that

$$Q_3' := \mathrm{area}(\mathrm{area}(A, A_1), A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k)$$

is a shuffle-polynomial of shuffle-degree k, where $\hat{Q}_3'$ is a shuffle-polynomial of shuffle-degree $k - 1$. Hence, $Q' = A_k \sqcup\!\sqcup Q_1' + Q_2' + Q_3'$ is a shuffle polynomial of shuffle-degree k and

$$\begin{aligned}3\,\mathrm{area}(A, A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) &= Q' + (\beta_{k-1} - 1) A_1 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k \sqcup\!\sqcup A \\ &\quad + \mathrm{area}(\mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k), A_1).\end{aligned} \tag{4.16}$$

It remains to consider the last term $\mathrm{area}(\mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k), A_1)$. By the Area-Jacobi identity and the anticommutativity of area we can rewrite this term as follows

$$\begin{aligned}\mathrm{area}(\mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k), A_1) &= \mathrm{area}(\mathrm{area}(A_1, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k), A) \\ &\quad - \mathrm{area}(\mathrm{area}(A_1, A), A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) \\ &\quad - A \sqcup\!\sqcup \mathrm{area}(A_1, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k) \\ &\quad + A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k \sqcup\!\sqcup \mathrm{area}(A_1, A) \\ &\quad + A_1 \sqcup\!\sqcup \mathrm{area}(A, A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k).\end{aligned}$$

Again, $\mathrm{area}(A_1, A)$ is a pure area, and by induction the term

$$Q_1'' := -\,\mathrm{area}(\mathrm{area}(A_1, A), A_2 \sqcup\!\sqcup \ldots \sqcup\!\sqcup A_k)$$

is a polynomial in pure areas of shuffle-degree at most k. The term

$$Q_2'' := A_2 \sqcup \ldots \sqcup A_k \sqcup \operatorname{area}(A_1, A)$$

is clearly a monomial in pure areas of shuffle-degree k. By induction we have that

$$P_1 := \operatorname{area}(A_1, A_2 \sqcup \ldots \sqcup A_k) = \beta_{k-1} A_1 \sqcup \ldots \sqcup A_k + P_1'$$

where P_1' is a polynomial in pure areas of shuffle-degree $k-1$. Similarly

$$P_2 := \operatorname{area}(A, A_2 \sqcup \ldots \sqcup A_k) = \beta_{k-1} A \sqcup A_2 \sqcup \ldots \sqcup A_k + P_2'$$

where P_2' is a polynomial in pure areas of shuffle-degree $k-1$. Therefore

$$Q_3'' := -A \sqcup \operatorname{area}(A_1, A_2 \sqcup \ldots \sqcup A_k) = -\beta_{k-1} A \sqcup A_1 \sqcup \ldots A_k - A \sqcup P_1'.$$

Similarly

$$Q_4'' := A_1 \sqcup \operatorname{area}(A, A_2 \sqcup \ldots \sqcup A_k) = \beta_{k-1} A \sqcup A_1 \sqcup \ldots A_k + A_1 \sqcup P_2'.$$

Combining terms we get a cancellation and degree reduction so that

$$\begin{aligned} Q_3'' + Q_4'' &= -\beta_{k-1} A_1 \sqcup \ldots A_k \sqcup A - A \sqcup P_1' + \beta_{k-1} A_1 \sqcup \ldots A_k \sqcup A + A_1 \sqcup P_2' \\ &= A_1 \sqcup P_2' - A \sqcup P_1' \end{aligned}$$

is a polynmomial in pure areas of shuffle-degree k. Setting $Q'' := Q_1'' + Q_2'' + Q_3'' + Q_4''$ (which is a polynomial in pure areas of shuffle degree k) and substituting in equation (4.16) we get

$$\begin{aligned} 3\operatorname{area}(A, A_1 \sqcup \ldots \sqcup A_k) &= (\beta_{k-1} - 1) A_1 \sqcup \ldots \sqcup A_k \sqcup A \\ &\quad + \operatorname{area}(\operatorname{area}(A_1, A_2 \sqcup \ldots \sqcup A_k), A) + Q' + Q'' \\ &= (\beta_{k-1} - 1) A_1 \sqcup \ldots \sqcup A_k \sqcup A + \operatorname{area}(P_1, A) + Q' + Q''. \end{aligned} \tag{4.17}$$

P_1 being a polynomial in pure areas of shuffle-degree $k-1$, we have by induction that $\operatorname{area}(P_1, A)$ is a polynomial in pure areas of shuffle-degree k. Hence, by construction $Q := Q' + Q'' + \operatorname{area}(P_1, A)$ is a polynomial in pure areas of shuffle-degree k. Therefore equation (4.17) becomes

$$\begin{aligned} 3\operatorname{area}(A, A_1 \sqcup \ldots \sqcup A_k) &= (\beta_{k-1} - 1) A_1 \sqcup \ldots \sqcup A_k \sqcup A \\ &\quad - \beta_{k-1} \operatorname{area}(A, A_1 \sqcup \ldots \sqcup A_k) + Q. \end{aligned}$$

Rearranging the terms we get the following final expression

$$\operatorname{area}(A, A_1 \sqcup \ldots \sqcup A_k) = \frac{\beta_{k-1} - 1}{\beta_{k-1} + 3} A_1 \sqcup \ldots \sqcup A_k \sqcup A + \frac{1}{\beta_{k-1} + 3} Q.$$

Observing that $\beta_k = \frac{\beta_{k-1}-1}{\beta_{k-1}+3}$ and noting that $\beta_1 = 0$ the result follows from Lemma 4.4.0.1. $\square$

Proof of Theorem 4.4.1. Since linear combinations of polynomials are polynomials, it suffices to prove that words in W_A^* are polynomial in pure areas. We prove by induction that every word $w \in W_A^*$ of length $|w| = n$ can be expressed as a polynomial in pure areas of shuffle-degree n. The result is trivial for $n = 0$. Let $n \geq 1$. We assume that w is a word of length $n > 0$ and that any word of length $< n$ can be written as a polynomial in pure areas of the appropriate degree.

Since $|w| > 0$, w can be written as follows

$$w = av = a \prec_A v \tag{4.18}$$

where $v \in W_A$ is of word of length $|v| = n - 1$ and $a \in A \subset \mathrm{Sh}(A)$ is a letter. Moreover for any elements of $\mathrm{Sh}(A)$

$$a \prec_A v = \frac{1}{2}(\mathrm{area}(a, v) + a \sqcup\!\sqcup v - \langle a, e\rangle \langle v, e\rangle e) \tag{4.19}$$

$$= \frac{1}{2}(\mathrm{area}(a, v) + a \sqcup\!\sqcup v) \tag{4.20}$$

since a is a letter. The length of the word v in (4.19) is equal to $n - 1$, so by induction it can be written as a polynomial in pure areas of shuffle-degree $n - 1$. Hence, the term $a \sqcup\!\sqcup v$ is a shuffle polynomial in pure areas of shuffle-degree n. By Theorem 4.4.2 the term $\mathrm{area}(a, v)$ is also a polynomial in pure areas of shuffle-degree n, and so w a polynomial in pure areas of shuffle-degree n. This concludes the induction and the proof. □

4.5 Hall sets and the dual Poincaré-Birkhoff-Witt basis

To present our structure theorems we will need to introduce special subsets of trees in $M(A)$ with a rich combinatorial structure. These sets are classically used to construct bases for the free Lie algebra $L(A)$.

4.5.1 Hall sets

Definition 4.5.1. A subset M of the magma $M(A)$ is **closed** if for any tree $t = (t', t'') \in M$ of degree ≥ 2 then its two "branches" t', and t'' are both in M. We call such subsets of $M(A)$ **sub-magmas**.

Remark 4.5.1. A union of sub-magmas is a sub-magma. By definition a letter is a sub-magma, and by induction any tree $t = (t', t'') \in M$ defines a sub-magma

$$M(t) := \{t\} \cup M(t') \cup M(t'').$$

Definition 4.5.2. A total order $<$ on a sub-magma M is an **ancestral order** if for any tree $t = (t', t'') \in M(A)$ of degree ≥ 2 one has $t < t''$.

This definition of ancestral order makes other constructions more transparent. It is obvious by construction that ancestral orders exist on any magma and their restrictions to sub-magma are also ancestral. Any ancestral order on a sub-magma can always be extended to an ancestral order on the full magma.

Definition 4.5.3 (**Hall set**)**.** A sub-magma H of $M(A)$ is a **Hall set** if

1. $<$ is an ancestral order on H.
2. $A \subset H$.
3. For any tree $h = (h_1, h_2) \in M(A)$ of degree ≥ 2, $h \in H$ if and only if:
   (a) $h_1, h_2 \in H$ and $h_1 < h_2$
   (b) either $h_1 \in A$ or $h_2 \leq h_1''$ where $h_1 = (h_1', h_1'')$.

As pointed out in [120, Proposition 4.1] and the surrounding discussion, Hall sets exist, any ancestral order on the full magma leads in a canonical way to to a unique Hall set, and every Hall set can be obtained in this way.

Lemma 4.5.0.1. [120, proof of Proposition 4.1] If $M(A)$ is a magma with an ancestral order $<$ and H is the set defined recursively by $A \subset H$ and if $h = (h_1, h_2) \in M(A)$ is of degree ≥ 2 then $h \in H$ whenever the following two conditions are satisfied:

1. $h_1, h_2 \in H$ and $h_1 < h_2$
2. either $h_1 \in A$ or $h_2 \leq h_1''$ where $h_1 = (h_1', h_1'')$.

then H is a Hall set, and is the unique Hall set compatible with the ancestral order $<$.

Lemma 4.5.0.2. [120, corollary 4.14] The number of Hall trees of degree n and the dimension of the space of homogeneous Lie polynomials of degree n are equal to

$$\mathcal{D}_H = \frac{1}{n} \sum_{d|n} \mu(d) q^{n/d} \tag{4.21}$$

where μ is the Mobius function.

4.5.2 The dual Poincaré-Birkhoff-Witt basis for the shuffle algebra

The Jacobi identities are linear relations between degree three lie brackets arising from associativity of the underlying group operation. They make the derivation of a basis for the free Lie algebra $L(A)$ quite a deep and classic challenge.

Definition 4.5.4. The extension of the Lie bracket to the operator $[\cdot] : M(A) \to L(A)$ is defined recursively as follows

1. if $a \in A$ is a letter, then $[a] = a$,
2. otherwise, if $t = (t_1, t_2)$, then $[t] = [[t_1], [t_2]]$.

Theorem 4.5.1. [120, theorem 4.9 (i)] For any Hall set H, the collection of elements $\{[h] : h \in H\}$ form a linear basis for the free Lie algebra $L(A)$.

This basis admits a canonical extension to a basis for the tensor algebra $T(A)$.

Theorem 4.5.2 (**PBW basis**)**.** [120, Theorem 4.9] The decreasing products

$$[h_1]^{k_1} \dots [h_n]^{k_n}, \quad h_i \in H, h_1 > \dots > h_n, \quad k_i \in \mathbb{N} \tag{4.22}$$

form a basis of the tensor algebra $T(A)$. This basis is called the **Poincaré-Birkhoff-Witt (PBW) basis**.

Definition 4.5.5. A word $w \in W_A$ is called a **Hall word** if w is the image of a Hall tree $h \in H$ by the foliage map, i.e. $w = f(h)$.

Remark 4.5.2. The foliage map is injective when restricted to the Hall set H and there are efficient algorithms for recovering the Hall tree from the Hall word [120].

Lemma 4.5.2.1. [120, Corollary 4.7] Every word $w \in W_A$ can be written uniquely as a decreasing product of Hall words

$$w = f(h_1)^{k_1} \dots f(h_n)^{k_n}, \quad h_i \in H, h_1 > \dots > h_n, \quad k_i \in \mathbb{N}. \tag{4.23}$$

Definition 4.5.6 (**PBW basis of** $T(A)$)**.** If $w \in W_A$ is a word decomposed into its unique decreasing product of Hall words according to equation (4.23), then P_w is the corresponding PBW basis element as per theorem 4.5.2

$$P_w := [h_1]^{k_1} \dots [h_n]^{k_n}, \quad h_i \in H, h_1 > \dots > h_n, \quad k_i \in \mathbb{N}.$$

$\{P_w\}_{w \in W_A}$ is thus an enumeration of the PBW basis indexed by words.

Definition 4.5.7 (**dual PBW basis of** $\mathrm{Sh}(A)$)**.** The basis of $\mathrm{Sh}(A)$, dual to the PBW basis $\{P_w\}_{w \in W_A}$, is the family $\{S_w\}_{w \in W_A}$ of series such that for any word $w \in W_A$

$$w = \sum_{w \in W_A} (S_w, w) P_w. \tag{4.24}$$

The next theorem provides exact formulae for the dual PBW basis $\{S_w\}_{w \in W_A}$.

Theorem 4.5.3 (**dual PBW basis**)**.** [120, Theorem 5.3] The basis S_w has the following properties:

1. if e is the empty word then $S_e = e$,
2. if $w = f(h_1)^{k_1} \dots f(h_n)^{k_n}$ is the unique factorization of the word w in a decreasing product of Hall trees $h_1 > \dots > h_n \in H$, then
$$S_w = \frac{1}{k_1! \dots k_n!} S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \dots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n} \in \mathrm{Sh}(A), \tag{4.25}$$
3. if $h \in H$, then $f(h) = av$ for some $a \in A$ and $v \in W_A$; moreover
$$S_{f(h)} = aS_v \tag{4.26}$$

Remark 4.5.3. It follows from the first and last parts of the theorem together that for any Hall tree $h = a \in A$ is a letter then $S_{f(h)} = S_a = a \in \mathrm{Sh}(A)$. Any Hall word h can be written as av for some unique letter a. If v is not the empty word it can be rewritten as a decreasing concatenation of Hall words $v = h_1^{k_1} \dots h_n^{k_n}$. Theorem 4.5.3 then provides a recursive construction for $S_{f(h)}$:

$$S_{f(h)} = a\frac{1}{k_1! \dots k_n!} S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \dots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n} \in \mathrm{Sh}(A). \tag{4.27}$$

Theorem 4.5.3 is an important result due to Schützenberger and it is the second structure theorem mentioned in the introduction. However, in section 4.6 we provide our version of this structure theorem which consists of a more explicit recursive formula for the dual PBW basis elements $\{S_w\}_{w \in W_A}$ and identify them as Hall integrals.

4.6 Polynomials in Hall integrals uniquely generate the shuffle algebra

We fix some choice of alphabet A, some choice of ancestral order $<$ on the magma $M(A)$, and we call the associated Hall set $H \subset M(A)$.

Definition 4.6.1 (**Hall integral**)**.** An element x of $\mathrm{Sh}(A)$ is called a **Hall integral** if it is the image under the operator $\prec : M(A) \to \mathrm{Sh}(A)$ of a Hall tree. That is to say, there exists a Hall tree $h \in H \subset M(A)$ so that $x = \prec(h)$.

Definition 4.6.2. A **(shuffle) polynomial in Hall integrals** is simply a sum of shuffles of Hall integrals.

Definition 4.6.3 (**Lazard decomposition**)**.** Any Hall tree $h \in H$ can be decomposed as

$$h = (h_1 h_2^k) = (\dots((h_1, h_2), h_2), \dots h_2) \tag{4.28}$$

with $h_1 \neq h_2$, $h_1, h_2 \in H$ and where the h_2 bracketing is repeated k times [120]. We call this the Lazard decomposition of h.

Definition 4.6.4 (**Lazard depth**)**.** If $h = (h_1 h_2^k)$ is the Lazard decomposition of a Hall tree $h \in H$ then we define the **Lazard depth** α_h of h to be $1/k$.

Definition 4.6.5 (**accumulated Lazard depth**)**.** The accumulated Lazard depth of a Hall tree $h \in H$ is defined recursively: $\mathcal{A}_h = 1$ if $h \in A$, otherwise $h = (h', h'')$ and $\mathcal{A}_h = \alpha_h \mathcal{A}_{h'} \mathcal{A}_{h''}$.

The following are the main results of this section.

4.6.1 Hall integrals are the dual Poincaré-Birkhoff-Witt basis

Theorem 4.6.1. For any Hall tree $h \in H \setminus A$ one has $h = (h', h'')$ and

$$S_{f(h)} = \alpha_h \left(S_{f(h')} \prec S_{f(h'')} \right) \tag{4.29}$$

where $\alpha_h \in \mathbb{Q}$ is the Lazard depth of h.

Theorem 4.6.2. For any Hall tree $h \in H$ one has

$$S_{f(h)} = \mathcal{A}_h(\prec(h)) \tag{4.30}$$

where $\mathcal{A}_h \in \mathbb{Q}$ is the accumulated Lazard depth of h.

Theorem 4.6.3. Consider all decreasing sequences $h_i \in H$, $h_1 > \ldots > h_n$, and strictly positive integers $k_i > 0$; then the elements

$$S_w = \frac{\mathcal{A}_{h_1}^{k_1} \dots \mathcal{A}_{h_n}^{k_n}}{k_1! \dots k_n!} (\prec(h_1))^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \dots \sqcup\!\sqcup (\prec(h_n))^{\sqcup\!\sqcup k_n} \tag{4.31}$$

are the dual basis in $\mathrm{Sh}(A)$ to the PBW basis $\{P_w = [h_1]^{k_1} \dots [h_n]^{k_n}\}_{w \in W_A}$ for $T(A)$.

Every element of $\mathrm{Sh}(A)$ is uniquely expressible as a shuffle polynomial in Hall integrals.

Before proving Theorem 4.6.1 we need the following combinatorial lemma.

Lemma 4.6.3.1. [120, corollary 5.14] Let $h = (h', h'') \in H$ be a Hall tree. Now $f(h) = av$, where $a \in A$ and $v \in W_A$. Let $v = f(h_1)^{k_1} \dots f(h_n)^{k_n}$ be the unique factorization of the word v in a decreasing product of Hall trees $h_1 > \ldots > h_n \in H$. Then

$$h'' = h_n. \tag{4.32}$$

Proof of Theorem 4.6.1. Let's write $f(h) = av$, with $a \in A$ and $v \in W_A^*$. Let $v = f(h_1)^{k_1} \dots f(h_n)^{k_n}$ be the unique factorization of the word v in a decreasing product of Hall trees $h_1 > \ldots > h_n \in H$. By Lemma 4.6.3.1 $h'' = h_n$. By Theorem 4.5.3 we also know that

$$\begin{aligned}
S_{f(h)} &= aS_v && (4.33)\\
&= S_a \prec S_v && (4.34)\\
&= \frac{1}{k_1! \dots k_n!} S_a \prec (S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \ldots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n}) && (4.35)\\
&= \frac{1}{k_1! \dots k_n!} S_a \prec ((S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \ldots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n - 1}) \sqcup\!\sqcup S_{f(h_n)}) && (4.36)\\
&= \frac{1}{k_1! \dots k_n!} S_a \prec ((S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \ldots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n - 1}) \sqcup\!\sqcup S_{f(h'')}) && (4.37)\\
&= \frac{1}{k_1! \dots k_n!} (S_a \prec (S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \ldots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n - 1})) \prec S_{f(h'')}. && (4.38)
\end{aligned}$$

Equation (4.33) is a restatement of (4.26) in Theorem 4.5.3. Equation (4.34) is immediate from the definition of $\prec$. Equation (4.35) follows from (4.25) in Theorem 4.5.3. Equation (4.36) is simply the associative property of the shuffle product. Equation (4.37) follows from Lemma 4.6.3.1. Equation (4.38) follows from the chain rule (4.9). Furthermore, the recursion identity (4.27) allows one to re-interpret the inner term in equation (4.38) as $S_{f(h')}$:

$$S_{f(h')} = \frac{1}{k_1! \dots (k_n - 1)!} S_a \prec (S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \ldots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n - 1}).$$

Substituting this into equation (4.38) and recalling the definition of the Lazard depth α_h we obtain

$$\begin{aligned}
S_{f(h)} &= \frac{1}{k_n} (S_{f(h')} \prec S_{f(h'')})\\
&= \alpha_h (S_{f(h')} \prec S_{f(h'')}).
\end{aligned}$$

□

Proof of Theorem 4.6.2. We may proceed by induction. For any Hall tree $h \in A$ one has $S_{f(h)} = h \in \mathrm{Sh}(A)$, $\prec(h) = h$, and $\mathcal{A}_h = 1$ and so the theorem is true. On the other hand if $h = (h', h'')$ then, assuming the result holds for h', h'':

$$\begin{aligned}
S_{f(h)} &= \alpha_h \left(S_{f(h')} \prec S_{f(h'')} \right)\\
&= \alpha_h \left((\mathcal{A}_{h'}(\prec(h'))) \prec (\mathcal{A}_{h''}(\prec(h''))) \right)\\
&= \mathcal{A}_h(\prec(h)),
\end{aligned}$$

where we use Theorem 4.6.1 for the first step, the truth of the result for h' and h'' for the second, and the recursive definitions of $\prec(h)$ and $\mathcal{A}_h$ for the third. So the result is true for h. This completes the induction. □

Proof of Theorem 4.6.3. Recall from Schützenberger's theorem (Theorem 4.5.3 in this chapter) that any element S_w in the dual basis to the PWB basis can be expressed uniquely as a shuffle monomial in $S_{f(h)}$. More precisely, consider the unique factorization of the word w as a decreasing product of Hall words $w = f(h_1)^{k_1} \dots f(h_n)^{k_n}$ where $h_1 > \dots > h_n \in H$, then the dual basis element

$$S_w = \frac{1}{k_1! \dots k_n!} S_{f(h_1)}^{\sqcup\!\sqcup k_1} \sqcup\!\sqcup \dots \sqcup\!\sqcup S_{f(h_n)}^{\sqcup\!\sqcup k_n} \in \mathrm{Sh}(A).$$

Theorem 4.6.2 allows for $i = 1 \dots n$ the substitution of $\mathcal{A}_{h_i}(\prec(h_i))$ for $S_{f(h_i)}$ in this formulae which gives the specified expression for the dual basis element in terms of Hall integrals and completes the proof. □

4.7 Conclusion

In this chapter we provided: 1) a more direct proof of the main result in [115] stating that polynomials in pure areas generate the shuffle algebra (Theorem 4.4.1); 2) a more direct proof of the main result in [116] reported without proof also in [117, 118] stating that polynomials in Hall integrals uniquely generate the shuffle algebra (Theorem 4.5.1).

These results can be interpreted as fundamental structure theorems for streamed information splitting functions uniquely into two components: a) a first that engages with the underlying stream, systematically extracts and packages the relevant information into atomic objects (pure areas, Hall integrals) whilst removing what's irrelevant; b) a second that evaluates polynomials in these features without reference to the original stream.

Appendix A

Background on rough path theory

In this appendix we present a brief summary of rough path theory and explain all the necessary material required to understand the results presented in Section 1.4, which should be self-contained. V, W will be two Banach spaces and $L(V, W)$ will denote the set of bounded linear maps from V to W. We will denote by $I = [0, T]$ a generic compact (time) interval. Consider two continuous paths $x : I \to V$ and $y : I \to W$ and a continuous function $f : W \to L(V, W)$, and let's assume that x, y and f are regular enough for the integral $\int_0^t f(y_s)dx_s$ to make sense for any $t \in I$. The case where x and y are of bounded variation is discussed in Chapter 3.

A.1 CDEs driven by paths of finite p-variation with $p < 2$

If our goal is to solve differential equations driven by more irregular paths than ones of bounded variation we should be prepared to compensate by allowing more smoothness on the vector fields f. In particular, if p and γ are real numbers such that $1 \leq p < 2$ and $p - 1 < \gamma \leq 1$ and x has finite p-variation, and if f is Hölder continuous with exponent γ, then the CDE 3.3 admits a solution [2, Theorem 1.20]. In order to get uniqueness we require f to be more regular than Hölder continuous. In the next definition we introduce a class of functions exhibiting the appropriate level of regularity to ensure uniqueness.

Definition A.1.1. Let V, W be Banach spaces, $k \geq 0$ an integer, $\gamma \in (k, k + 1]$ and C a closed subset of V. Let $f : C \to W$ be a function. For each integer $n = 1, ..., k$, let $f^n : C \to L(V^{\otimes n}, W)$ be a function taking its values in the space of symmetric n-linear forms from V to W, where $\otimes$ denotes the tensor product of vector space (that we will define more precisely in the next section). The collection $(f_0 = f, f^1, ..., f^k)$ is an element of $Lip(\gamma, C)$ if

there exists a constant $M \geq 0$ such that for each $n = 0, ..., k$

$$\sup_{x \in C} |f^n(x)| \leq M \tag{A.1}$$

and there exists a function $R_n : V \times V \to L(V^{\otimes n}, W)$ such that for any $x, y \in C$ and any $v \in V^{\otimes n}$

$$f^n(y)(v) = \sum_{i=0}^{k-n} \frac{1}{i!} f^{n+i}(x)(v \otimes (y-x)^{\otimes i}) + R_n(x,y)(v) \tag{A.2}$$

and

$$|R_n(x,y)| \leq M|x-y|^{\gamma - n}. \tag{A.3}$$

The smallest constant M for which the inequalities above hold for all n is called the $Lip(\gamma, C)$-norm of f and is denoted by $||f||_{Lip(\gamma)}$. We will call such function f a $Lip(\gamma)$ function (or one-form).

Remark A.1.1. The best way to understand of a $Lip(\gamma)$ function is to think about it as a function that "locally looks like a polynomial function". Let us explain the connection to paths. Let $k \geq 0$ be an integer. Let $P : V \to \mathbb{R}$ be a polynomial of degree k. Let $x : [0, T] \to V$ be a continuous path of bounded variation. If $P^1 : V \to L(V, \mathbb{R})$ denotes the derivative of P then by Taylor's theorem we have that for any $0 \leq s \leq t \leq T$

$$P(x_t) = P(x_s) + \int_s^t P^1(x_u) dx_u. \tag{A.4}$$

Substituting the expression of P into P^1 in the last expression we get

$$P(x_t) = P(x_s) + P^1(x_s) \int_{s<u_1<t} dx_{u_1} + \iint_{s<u_1<u_2<t} P^2(x_{u_1}) dx_{u_1} \otimes dx_{u_2}, \tag{A.5}$$

where $P^2 : V \to L(V \otimes V, \mathbb{R})$ is the second derivative of P, and $L(V \otimes V, \mathbb{R})$ is the space of bilinear forms on V. Iterating the substitutions, the procedure stops at level $k+1$ yielding the following expression

$$\begin{aligned} P(x_t) = P(x_s) + P^1(x_s) \int_{s<u_1<t} dx_{u_1} + P^2(x_s) \iint_{s<u_1<u_2<t} dx_{u_1} \otimes dx_{u_2} \\ + ... + P^k(x_s) \int_{s<u_1<...<u_k<t} ... \int dx_{u_1} \otimes ... \otimes dx_{u_k}, \end{aligned} \tag{A.6}$$

where for each $n \in \{0, ..., k\}$, $P^n : V \to L(V^{\otimes n}, \mathbb{R})$ denotes the n^{th} derivative of P which takes its values in the space of symmetric n-linear forms on V. The symmetric part of the tensor $\int_{0<u_1<...<u_k<t} ... \int dx_{u_1} \otimes ... \otimes dx_{u_k}$ is equal to $\frac{1}{k!}(x_t - x_s)^{\otimes k}$. Hence

$$P(x_t) = P(x_s) + \sum_{n=1}^{k} P^n(x_s) \frac{(x_t - x_s)^{\otimes k}}{k!}. \tag{A.7}$$

The general concept of $Lip(\gamma)$ function where $\gamma \in (k, k+1]$ for some $k \geq 0$ mimics closely the behaviour defined by Equation (A.7) up to an error which is Hölder continuous of exponent $\gamma - k$. Indeed, using the notation from Definition A.1.1, if $x : [0,T] \to C$ takes its values in a closed subset C of V, for any $0 \leq s \leq t \leq T$, any $n = 0, ..., k$ and any $v \in V^{\otimes n}$ we have the following relation

$$f^n(x_t)(v) = \sum_{i=0}^{k-n} f^{n+i}(x_s)\left(v \otimes \int_{0<u_1<...<u_i<t} ... \int dx_{u_1} \otimes ... \otimes dx_{u_i}\right) + R_n(x_s, x_t)(v), \qquad (A.8)$$

which is analogous to Equation (A.6). This is the general form of a $Lip(\gamma)$ function that we will consider from now on.

If p and γ are such that $1 \leq p < 2$ and $p < \gamma$, $x : [0,T] \to V$ is a continuous path of finite p-variation and f be a $Lip(\gamma)$ function, then for any $a \in W$, the CDE (3.3) admits a unique solution. Furthermore, if $y = I_f(x, a)$ denotes the unique solution, then the function I_f, that maps V-valued paths of finite p-variation to W-valued paths of finite p-variation, is continuous in the $||\cdot||_p$ norm [2, Theorem 1.28].

We are now able to give meaning to CDEs where the control is not necessarily of bounded variation. However, if the driving path x has finite 2-variation but infinite p-variation for every $p < 2$, we are still not able to give a meaning to the CDE (3.3). This is quite a hard restriction given that Brownian paths have finite p-variation only for $p > 2$.

To be able to push this barrier even further we first need to introduce some algebraic spaces and construct one of the central objects in rough path theory, the signature of a path. We do so very briefly in the next section and then explain in Section 3.4.1 the connection with (linear) CDEs.

A.2 Multiplicative functionals and the Extension Theorem

In this section we push the limit on the regularity of the driver even further and explain how to solve CDEs driven by rough paths, as presented in the seminal paper [112]. This will demand to define what a (geometric) rough path is and to present a powerful theory of integration of one-forms along rough paths to what will follow the Universal Limit Theorem for CDEs driven by rough paths (Theorem A.4.1). In what follows, Δ_T will denote the simplex

$$\Delta_T = \{(s,t) \in [0,T]^2 : 0 \leq s \leq t \leq T\}. \qquad (A.9)$$

Definition A.2.1. Let $N \geq 1$ be an integer and let $X : \Delta_T \to T^N(V)$ be a continuous map

$$X_{s,t} = (X^0_{s,t}, X^1_{s,t}, ..., X^N_{s,t}) \in \mathbb{R} \oplus V \oplus V^{\otimes 2} \oplus ... \oplus V^{\otimes N}. \tag{A.10}$$

X is said to be a multiplicative functional if $X^0_{s,t} = 1$ for all $(s,t) \in \Delta_T$ and

$$X_{s,u} \otimes X_{u,t} = X_{s,t}, \quad \forall s,u,t \in [0,T],\ s \leq u \leq t. \tag{A.11}$$

The Chen's identity (A.11) imposed on a multiplicative functional is a purely algebraic one. We are now going to describe an analytic condition for a multiplicative functional through the notion of p-variation.

Definition A.2.2. A control is a continuous non-negative function $\omega : \Delta_T \to [0, +\infty)$ which is super-additive in the sense that

$$w(s,t) + w(t,u) \leq w(s,u), \quad \forall s \leq t \leq u \in I \tag{A.12}$$

and for which $w(t,t) = 0$ for all $t \in I$.

Definition A.2.3. Let $p \geq 1$ be a real number and $N \geq 1$ be an integer. Let $\omega : [0,T] \to [0, +\infty)$ be a control and $X : \Delta_T \to T^N(V)$ be a multiplicative functional. We say that X has finite p-variation on Δ_T controlled by ω if

$$||X^i_{s,t}||_{V^{\otimes i}} \leq \frac{\omega(s,t)^{i/p}}{\beta(i/p)!}, \quad \forall (s,t) \in \Delta_T,\ \forall i = 1, ..., N \tag{A.13}$$

where $(i/p)! = \Gamma(i/p)$ and where β is a real constant that depends only on p and that we will make explicit later in the chapter. We say that X has finite p-variation if there exists a control ω such that this condition is satisfied.

Remark A.2.1. The p-variation $w(s,t) = ||x||^p_{p,[s,t]}$ function defines a control.

Definition A.2.4. Let $X, Y : \Delta_T \to T^N(V)$ be two multiplicative functionals. The p-variation metric is defined as follows

$$d_p(X,Y) = \sup_{1 \leq i \leq N} \sup_{\mathcal{D}} \left(\sum_{t_k \in \mathcal{D}} ||X^i_{t_k,t_{k+1}} - Y^i_{t_k,t_{k+1}}||^{p/i} \right)^{1/p} \tag{A.14}$$

where $||\cdot||$ denotes any norm on the corresponding level $V^{\otimes i}$.

The next theorem is one of the fundamental results in rough path theory. It states that every multiplicative functional of degree N and of finite p-variation can be extended in a unique way to a multiplicative functional of arbitrarily high degree provided N is greater than the integer part of p, denoted by $\lfloor p \rfloor$. Furthermore the extension map is continuous in the p-variation metric.

Theorem A.2.1. [2, Extension Theorems 3.7 & 3.10] Let $p \geq 1$ be a real number and $N \geq 1$ an integer. Let $X : \Delta_T \to T^N(V)$ be a multiplicative functional of finite p-variation controlled by a control ω. Assume that $N \geq \lfloor p \rfloor$. Then, for any $m \geq \lfloor p \rfloor + 1$, there exist a unique continuous function $X^m : \Delta_T \to V^{\otimes m}$ such that

$$(s,t) \mapsto X_{s,t} = (1, X^1_{s,t}, ..., X^{\lfloor p \rfloor}_{s,t}, ..., X^m_{s,t}, ...) \in T((V)) \tag{A.15}$$

is a multiplicative functional of finite p-variation controlled by w according to the inequality (A.13) with

$$\beta = p^2 \left(1 + \sum_{r=3}^{\infty} \left(\frac{2}{r-2} \right)^{\frac{\lfloor p \rfloor + 1}{l}} \right). \tag{A.16}$$

Moreover, the extension map is continuous in the p-variation metric in the sense that if for some $\epsilon > 0$ we have

$$||X^i_{s,t} - Y^i_{s,t}|| \leq \epsilon \frac{\omega(s,t)^{i/p}}{\beta(i/p)!}, \quad i = 1, ..., N, \ (s,t) \in \Delta_T \tag{A.17}$$

then, provided

$$\beta \geq 2p^2 \left(1 + \sum_{r=3}^{\infty} \left(\frac{2}{r-2} \right)^{\frac{\lfloor p \rfloor + 1}{l}} \right) \tag{A.18}$$

the bound in Equation (A.17) holds for all $i \geq 1$.

Definition A.2.5. Let V be a Banach space and $p \geq 1$ be a real number. A p-rough path in V is a multiplicative functional of degree $\lfloor p \rfloor$ with finite p-variation. We denote the space of p-rough paths over V by $\Omega_p(V)$.

Hence, a p-rough path is a continuous map from $\Delta_T \to T^{\lfloor p \rfloor}(V)$ that satisfies an algebraic condition (Chen's identity) and an analytic condition (finite p-variation). As a consequence of the Extension Theorem (A.2.1), we get that every p-rough path in $\Omega_p(V)$ has a full signature, i.e. it can be extended to a multiplicative functional of arbitrary high degree with finite p-variation in V.

Remark A.2.2. The signature of a paths of bounded variation, truncated at any level, is a multiplicative functional, so in particular it satisfies Chen's identity. This allows for fast signature computations for time series data available in several software packages [121, 109, 38].

Recall that a path of finite p-variation has finite q-variation for any $q \geq p$. In particular, a path of bounded variation is a p-rough path for any $p \geq 1$. We now dispose of all the necessary ingredients to define what a geometric p-rough path is.

Definition A.2.6. A geometric p-rough path is a p-rough path that can be expressed as the limit of a sequence of 1-rough paths in the p-variation metric. We denote by $G\Omega_p(V)$ the space of geometric p-rough paths in V.

Hence, $G\Omega_p(V)$ is the closure in $(\Omega_p(V), d_p)$ of the space $\Omega_1(V)$ of continuous paths of bounded variation. Next we apply the ideas developed so far in order to define a powerful theory of integration for solving differential equations driven by (geometric) rough paths.

A.3 Integration of a one-form along a rough path

In order to define what we mean by a solution of a differential equation driven by a rough path, we need a theory of integration for rough paths. The correct notion of integral turns out to be the integral of a one-form along a rough path. We start by defining the concept of almost p-rough path.

Definition A.3.1. Let $p \geq 1$ be a real number. Let $\omega : \Delta_I \to [0, +\infty)$ be a control. A function $X : \Delta_I \to T^{\lfloor p \rfloor}(V)$ is called an almost p-rough path if

1. it has finite p-variation controlled by w, i.e.

$$\left\| X^i_{s,t} \right\| \leq \frac{\omega(s,t)^{i/p}}{\beta(i/p)!}, \quad \forall (s,t) \in \Delta_t, \ \forall i = 0, ..., \lfloor p \rfloor; \tag{A.19}$$

2. it is an almost multiplicative functional, i.e. there exists $\theta > 1$ such that

$$||(X_{s,u} \otimes X_{u,t})^i - X^i_{s,t}|| \leq \omega(s,t)^\theta, \quad \forall (s,t) \in \Delta_I, \forall i = 0, ..., \lfloor p. \rfloor \tag{A.20}$$

To be specific we say that X is a θ-almost p-rough path controlled by ω.

For any almost p-rough path there exists a unique p-rough path that it is close to it in some specific sense, as stated in the next theorem.

Theorem A.3.1. [2, theorem 4.3 & 4.4] Consider two scalars $p \geq 1$ and $\theta \geq 1$, a control $\omega : \Delta_T \to [0, +\infty)$. Let $X : \Delta_T \to T^{\lfloor p \rfloor}(V)$ be a θ-almost p-rough path controlled by ω. Then there exists a unique p-rough path $\widetilde{X} : \Delta_T \to T^{\leq \lfloor p \rfloor}(V)$ such that

$$\sup_{\substack{0 \leq s < t \leq T \\ i = 0, ..., \lfloor p \rfloor}} \frac{||\widetilde{X}^i_{s,t} - X^i_{s,t}||}{\omega(s,t)^\theta} < +\infty. \tag{A.21}$$

Furthermore, the map that to an almost p-rough path associates a rough path is continuous in p-variation.

Remark A.3.1. It is important to note that, if X and Y are two p-rough paths with $p \geq 2$, and f is a map, even the smoothest one, it is not possible to give a sensible meaning to something like $\int f(Y)dX$. In effect, as shown by the simple example $\int_s^t Y_u dX_u = \iint\limits_{s<u_1<u_2<t} dY_{u_1} dX_{u_2}$, the definition of such an object would necessarily involve some cross-iterated integrals of X and Y, which are not available directly in the data of X and Y. In other words, two rough paths $X \in \Omega_p(V)$ and $Y \in \Omega_p(W)$ do not determine the joint path (X, Y) as a rough path in $\Omega_p(V \oplus W)$. However, if one knows the joint path $Z = (X, Y)$ as a rough path, then the integral $\int f(Y)dX$ can be defined as a special case of $\int \alpha(Z)dZ$, where α is a one-form. This is the object we are going to construct in the sequel.

We will define the integral of a one-form along a rough path as the unique p-rough path associated to, in the sense of Theorem A.3.1, a specific almost p-rough path that we now construct.

Let $\gamma > p \geq 1$ be scalars and let $Z : \Delta_T \to T^{\lfloor p \rfloor}(V)$ be a p-rough path controlled by some control ω. Let $\alpha : V \to L(V, W)$ be a $Lip(\gamma - 1)$ function (Definition A.1.1) that we will call a one-form from now on. Hence, we are given $\alpha^0 = \alpha$ and at least $\lfloor p \rfloor - 1$ auxiliary functions $\alpha^1, ..., \alpha^{\lfloor p \rfloor - 1}$ such that for any $k = 1, ..., \lfloor p \rfloor - 1$

$$\alpha^k : V \to L(V^{\otimes k}, L(V, W)), \qquad k = 1...\lfloor p \rfloor - 1 \tag{A.22}$$

satisfying the Taylor-like expansion: $\forall x, y \in V$

$$\alpha(y) = \alpha(x) + \sum_{k=1}^{\lfloor p \rfloor - 1} \alpha^k(x) \frac{(y-x)^{\otimes k}}{k!} + R_0(x, y) \tag{A.23}$$

with $||R_0(x, y)|| \leq ||\alpha||_{Lip} ||x - y||^{\gamma - 1}$, where $||\alpha||_{Lip}$ is the Lipschitz constant of α. We are going to look for an approximation of $\int_s^t \alpha(Z_u)dZ_u$ in the form of a Taylor expansion. For any $(s, u) \in \Delta_T$ we can write

$$\alpha(Z_u) = \sum_{k=0}^{\lfloor p \rfloor - 1} \alpha^k(Z_s) Z_{s,u}^k + R_0(Z_s, Z_u). \tag{A.24}$$

Now, by definition of the iterated integrals of Z, the following relation holds for any $k \geq 0$

$$\int_s^t Z_{s,u}^k dZ_u = Z_{s,t}^{k+1}, \tag{A.25}$$

where with the multiplication $Z_{s,u}^k dZ_u$ we mean $Z_{s,u}^k \otimes dZ_u$. Combining Equation (A.24) and Equation (A.25) we obtain

$$\int_s^t \alpha(Z_u)dZ_u = \sum_{k=0}^{\lfloor p \rfloor - 1} \alpha^k(Z_s) Z_{s,t}^{k+1} + \int_s^t R_0(Z_s, Z_u)dZ_u. \tag{A.26}$$

Let us now define the following W-valued path

$$Y^1_{s,t} := \sum_{k=0}^{\lfloor p \rfloor - 1} \alpha^k(Z_s) Z^{k+1}_{s,t} \tag{A.27}$$

and let us compute the higher order iterated integrals of Y in such a way that it becomes an almost rough path. For any $n \geq 2$ it can be shown after some tedious calculations that

$$Y^n_{s,t} = \sum_{k_1,\dots,k_n=1}^{\lfloor p \rfloor} \alpha^{k_1-1}(Z_s)...\alpha^{k_n-1}(Z_s) \int_{s<u_1<\dots<u_n<t} \dots \int dZ^{k_1}_{s,u_1} \otimes ... \otimes dZ^{k_n}_{s,u_n}. \tag{A.28}$$

It turns out that the Y^n we have just defined is an almost rough path, as stated in the next theorem. We will use this as our approximation for $\int \alpha(Z) dZ$.

Theorem A.3.2. [32, Theorem 4.6] Let $Z : \Delta_T \to T^{\lfloor p \rfloor}(V)$ be a geometric p-rough path and $\alpha : V \to L(V, W)$ be a $Lip(\gamma - 1)$ one-form for some $\gamma > p$. Then $Y : \Delta_T \to T^{\lfloor p \rfloor}(W)$ defined for all $(s,t) \in \Delta_T$ and any $n \geq 1$ by Equation (A.28) is a $\frac{\gamma}{p}$-almost p-rough path.

We can now define the integral $\int \alpha(Z) dZ$ of the one-form α along the rough path Z as the unique rough path associated to the almost rough path Y according to Theorem A.3.1.

Definition A.3.2. Let Z, α and Y be as in Theorem A.3.2. The unique p-rough path $\mathcal{I} : \Delta_T \to T^{\lfloor p \rfloor}(W)$ associated to Y by Theorem A.3.1 is called the integral of the one-form α along the rough path Z and is denoted by $\mathcal{I}_{s,t} := \int_s^t \alpha(Z_u) dZ_u$.

Remark A.3.2. As stated in [2, Theorem 4.12], the map $Z \mapsto \int \alpha(Z) dZ$ is continuous in p-variation from $G\Omega_p(V)$ to $G\Omega_p(W)$.

Definition A.3.3. Let $A \in L(V, W)$ be a continuous linear map between Banach spaces and $X : \Delta_T \to T^{\lfloor p \rfloor}(V)$ be a geometric p-rough path. The image $A(X)$ of the rough path X by the linear map A is a rough path defined in the following way. A induces a linear map from $V^{\otimes k}$ to $W^{\otimes k}$ which sends $x_1 \otimes ... \otimes x_k$ to $Ax_1 \otimes ... \otimes Ax_k$. By linearity of the direct sum defining $T^{\lfloor p \rfloor}(V)$ and $T^{\lfloor p \rfloor}(W)$, A can be further extended to a linear map between these two spaces, denoted by $\mathcal{T}(A) \in L\left(T^{\lfloor p \rfloor}(V), T^{\lfloor p \rfloor}(W)\right)$. Then, $A(X)$ is defined as the following rough path

$$A(X)_{s,t} = \mathcal{T}(A)(X_{s,t}), \quad \forall (s,t) \in \Delta_T. \tag{A.29}$$

In the next section we will finally describe how to solve CDEs driven by rough paths.

A.4 Non-linear CDEs driven by rough paths

In this section we explain how to solve CDEs driven by rough paths. As in the classical case of CDEs driven by paths of bounded variation, the existence and uniqueness of the solution depend on assumptions about the smoothness of the vector field.

Let V, W be two Banach spaces and $\gamma > p \geq 1$ be two real numbers. Let $f : W \to L(V, W)$ be a $Lip(\gamma - 1)$ one-form. Consider a geometric p-rough path $X \in G\Omega_p(V)$ and a point $a \in W$. It is easy to see that when X has finite 1-variation, the equation

$$dY_t = f(Y_t)dX_t, \quad Y_0 = a \tag{A.30}$$

is equivalent, up to translation of the initial condition $a \in W$, to the following system

$$dY_t = f_a(Y_t)dX_t, \quad Y_0 = 0 \tag{A.31}$$

$$dX_t = dX_t \tag{A.32}$$

where $f_a(w) = f(w + a)$ for all $w \in W$. Consider now the one-form $h : V \oplus W \to L(V \oplus W, V \oplus W)$ defined as follows

$$h(v, w) = \begin{pmatrix} I_V & 0 \\ f_a(w) & 0 \end{pmatrix} \tag{A.33}$$

where $I_V : V \to V$ is the identity map on V. Then, when $X \in \Omega_1(V)$ is of bounded variation, solving (A.33) is equivalent to finding $Z \in \Omega_1(V \oplus W)$ such that

$$dZ_t = h(Z_t)dZ_t, \quad Z_0 = 0, \ \pi_V(Z) = X \tag{A.34}$$

where π_V is the canonical projection of $V \oplus W$ onto V. What the next definition states is that this reformulation makes sense even when X is a rough path.

Definition A.4.1. Consider the notation introduced above. We call $Z \in G\Omega_p(V \oplus W)$ a solution of the differential equation (A.30) if the following two conditions hold:

$$Z_{s,t} = \int_s^t h(Z_u)dZ_u \tag{A.35}$$

$$\pi_V(Z) = X \tag{A.36}$$

where $\pi_V(Z)$ is the image of the rough path Z by the linear map π_V in the sense of Definition A.3.3.

In order to find a solution to the CDE (A.30) we are going to, once again, use Picard iteration. Define $Z(0) = (X, \mathbf{0}) \in G\Omega_p(V \oplus W)$, where $\mathbf{0}$ denotes the 0-rough path $\mathbf{0}_{s,t} = (1, 0, 0, ...)$. Then, for any $n \geq 0$ we define the sequence of rough paths

$$Z(n)_{s,t} = \int_s^t h(Z(n))dZ(n). \tag{A.37}$$

Let us now denote $Y(n) = \pi_W(Z(n))$ so that $Z(n) = (X, Y(n))$. One of the main results in the seminal paper [112] is the following Universal Limit Theorem (ULT). Note that for the solution to be unique we require one extra degree of smoothness on f.

Theorem A.4.1 (Universal Limit Theorem (ULT)). [2, Theorem 5.3] Let $p \geq 1$ and $\gamma > p$ be real numbers. Let $f : W \to L(V, W)$ be a $Lip(\gamma)$ function. For all $X \in G\Omega_p(V)$ and all $a \in W$ the equation

$$dY_t = f(Y_t)dX_t, \qquad Y_0 = a \tag{A.38}$$

admits a unique solution $Z = (X, Y) \in G\Omega_p(V \oplus W)$ in the sense of Definition A.4.1. Furthermore, the rough path Y is the limit of of the sequence of rough paths $Y(n)$ defined above, and the mapping $I_f : G\Omega_p(V) \times W \to G\Omega_p(W)$ which sends (X, a) to Y is continuous in p-variation.

Bibliography

[1] Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 2019.

[2] Terry Lyons, Michael Caruana, and Thierry Lévy. Differential equations driven by rough paths. *Ecole d'été de Probabilités de Saint-Flour XXXIV*, pages 1–93, 2004.

[3] Adeline Fermanian. Embedding and learning with signatures. *arXiv preprint arXiv:1911.13211*, 2019.

[4] Terry Lyons. Rough paths, signatures and the modelling of functions on streams. *International Congress of Mathematicians, Seoul*, 2014.

[5] Imanol Perez Arribas, Guy M Goodwin, John R Geddes, Terry Lyons, and Kate EA Saunders. A signature-based machine learning model for distinguishing bipolar disorder and borderline personality disorder. *Translational psychiatry*, 8(1):1–7, 2018.

[6] James H Morrill, Andrey Kormilitzin, Alejo J Nevado-Holgado, Sumanth Swaminathan, Samuel D Howison, and Terry J Lyons. Utilization of the signature method to identify the early onset of sepsis from multivariate physiological time series in critical care monitoring. *Critical Care Medicine*, 48(10):e976–e981, 2020.

[7] Patrick Kidger, Patric Bonnier, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[8] Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, and Lianwen Jin. Developing the path signature methodology and its application to landmark-based human action recognition. *arXiv preprint arXiv:1707.03993*, 2017.

[9] PJ Moore, TJ Lyons, J Gallacher, and Alzheimer's Disease Neuroimaging Initiative. Using path signatures to predict a diagnosis of alzheimer's disease. *PloS one*, 14(9):e0222212, 2019.

[10] Thomas Cochrane, Peter Foster, Varun Chhabra, Maud Lemercier, Cristopher Salvi, and Terry Lyons. Sk-tree: a systematic malware detection algorithm on streaming trees via the signature kernel. *arXiv preprint arXiv:2102.07904*, 2021.

[11] Terry Lyons, Sina Nejad, and Imanol Perez Arribas. Numerical method for model-free pricing of exotic derivatives in discrete time using rough path signatures. *Applied Mathematical Finance*, 26(6):583–597, 2019.

[12] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[13] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[14] Marco Cuturi. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 929–936, 2011.

[15] Edouard Goursat. *A Course in Mathematical Analysis: pt. 2. Differential equations.[c1917*, volume 2. Dover Publications, 1916.

[16] Csaba Toth and Harald Oberhauser. Bayesian learning from sequential data using gaussian processes with signature covariances. In *Proceedings of the international conference on machine learning (ICML)*, 2020.

[17] Ilya Chevyrev and Harald Oberhauser. Signature moments to characterize laws of stochastic processes. *arXiv preprint arXiv:1810.10971*, 2018.

[18] Khalil Chouk and Massimiliano Gubinelli. Rough sheets. *arXiv preprint arXiv:1406.7748*, 2014.

[19] Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *arXiv preprint arXiv:1601.08169*, 2016.

[20] Milton Lees. The goursat problem. *Journal of the Society for Industrial and Applied Mathematics*, 8(3):518–530, 1960.

[21] J. T. Day. A runge-kutta method for the numerical solution of the goursat problem in hyperbolic partial differential equations. *The Computer Journal*, 9:81–83, 1966.

[22] A. M. Wazwaz. On the numerical solution for the goursat problem. *Applied Mathematics and Computation*, 59:89–95, 1993.

[23] Thomas Cass, Terry Lyons, Cristopher Salvi, and Weixin Yang. Computing the full signature kernel as the solution of a goursat problem. *arXiv preprint arXiv:2006.14794*, 2020.

[24] Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019.

[25] Patrick Kidger, James Foster, Xuechen Li, Harald Oberhauser, and Terry Lyons. Neural sdes as infinite-dimensional gans. *arXiv preprint arXiv:2102.03657*, 2021.

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

[27] Christa Cuchiero, Wahid Khosrawi, and Josef Teichmann. A generative adversarial network approach to calibration of local stochastic volatility models. *Risks*, 8(4):101, 2020.

[28] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative Finance*, 18(6):933–949, 2018.

[29] Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.

[30] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[31] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[32] Terry Lyons and Nicolas Victoir. An extension theorem to rough paths. In *Annales de l'IHP Analyse non linéaire*, volume 24, pages 835–847, 2007.

[33] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660, 2017.

[34] Francesco Cosentino, Harald Oberhauser, and Alessandro Abate. A randomized algorithm to reduce the support of discrete measures. *arXiv preprint arXiv:2006.01757*, 2020.

[35] Nicholas I Sapankevych and Ravi Sankar. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, 4(2):24–38, 2009.

[36] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[37] C. Toth C. Salvi. personal communication.

[38] Jeremy Reizenstein and Benjamin Graham. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv preprint arXiv:1802.08252*, 2018.

[39] Vladimir Vapnik. The support vector method of function estimation. In *Nonlinear Modeling*, pages 55–85. Springer, 1998.

[40] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[41] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, 2001.

[42] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & operations research*, 32(10):2513–2522, 2005.

[43] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[44] Yutian Chen, Max Welling, and Alex Smola. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 109–116, 2010.

[45] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[46] Francis R Bach, Simon Lacoste-Julien, and Guillaume Obozinski. On the equivalence between herding and conditional gradient algorithms. In *ICML*, 2012.

[47] Christian Litterer, Terry Lyons, et al. High order recombination and an application to cubature on wiener space. *The Annals of Applied Probability*, 22(4):1301–1327, 2012.

[48] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Found. Trends Mach. Learn.*, 2012.

[49] Mark Schmidt, Nicolas L Roux, and Francis R Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in neural information processing systems*, pages 1458–1466, 2011.

[50] Linda E Reichl. A modern course in statistical physics, 1999.

[51] Anastasia Papavasiliou, Christophe Ladroue, et al. Parameter estimation for rough differential equations. *The Annals of Statistics*, 39(4):2047–2073, 2011.

[52] Snehal S Dahikar and Sandeep V Rode. Agricultural crop yield prediction using artificial neural network approach. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, 2(1):683–686, 2014.

[53] Jiaxuan You, Xiaocheng Li, Melvin Low, David Lobell, and Stefano Ermon. Deep gaussian process for crop yield prediction based on remote sensing data. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[54] Ilya Chevyrev, Terry Lyons, et al. Characteristic functions of measures on geometric rough paths. *The Annals of Probability*, 44(6):4049–4082, 2016.

[55] Terry Lyons, Hao Ni, et al. Expected signature of brownian motion up to the first exit time from a bounded domain. *The Annals of Probability*, 43(5):2729–2762, 2015.

[56] Hao Ni. *The expected signature of a stochastic process.* PhD thesis, Oxford University, UK, 2012.

[57] Thomas Fawcett. *Problems in stochastic analysis: Connections between rough paths and non-commutative harmonic analysis.* PhD thesis, University of Oxford, 2002.

[58] Terry J Lyons, Michael Caruana, and Thierry Lévy. *Differential equations driven by rough paths.* Springer, 2007.

[59] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[60] Andreas Christmann and Ingo Steinwart. Universal kernels on non-standard input spaces. In *Advances in neural information processing systems*, pages 406–414, 2010.

[61] Charles Walkden. Ergodic theory. *Lecture Notes University of Manchester*, 2014.

[62] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[63] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[65] Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

[66] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pages 7576–7586, 2018.

[67] Oliver Hamelijnck, Theodoros Damoulas, Kangrui Wang, and Mark Girolami. Multi-resolution multi-task gaussian processes. In *Advances in Neural Information Processing Systems*, pages 14025–14035, 2019.

[68] Ho Chung Law, Dino Sejdinovic, Ewan Cameron, Tim Lucas, Seth Flaxman, Katherine Battle, and Kenji Fukumizu. Variational learning on aggregate outputs with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 6081–6091, 2018.

[69] David R Musicant, Janara M Christensen, and Jamie F Olson. Supervised learning by training on aggregate outputs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 252–261. IEEE, 2007.

[70] Kiri L Wagstaff, Terran Lane, and Alex Roper. Multiple-instance regression with structured data. In *2008 IEEE International Conference on Data Mining Workshops*, pages 291–300. IEEE, 2008.

[71] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[72] Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A Osborne. On the limitations of representing functions on sets. In *International Conference on Machine Learning*, pages 6487–6494. PMLR, 2019.

[73] Zoltán Szabó, Bharath K Sriperumbudur, Barnabás Póczos, and Arthur Gretton. Learning theory for distribution regression. *The Journal of Machine Learning Research*, 17(1):5272–5311, 2016.

[74] Krikamol Muandet, Kenji Fukumizu, Francesco Dinuzzo, and Bernhard Schölkopf. Learning from distributions via support measure machines. In *Advances in neural information processing systems*, pages 10–18, 2012.

[75] Seth R Flaxman. *Machine learning in space and time*. PhD thesis, Ph. D. thesis, Carnegie Mellon University, 2015.

[76] Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes, and Tomoko Matsui. A kernel for time series based on global alignments. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 2, pages II–413. IEEE, 2007.

[77] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International Conference on Machine Learning*, pages 894–903. PMLR, 2017.

[78] Clement John Adkins and Clement John Adkins. *Equilibrium thermodynamics*. Cambridge University Press, 1983.

[79] Jeffrey Chang. Simulating an ideal gas to verify statistical mechanics, 2015. `http://stanford.edu/~jeffjar/files/simulating-ideal-gas.pdf`.

[80] Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. *arXiv preprint arXiv:1603.03788*, 2016.

[81] Imanol Perez Arribas, Cristopher Salvi, and Lukasz Szpruch. Sig-sdes model for quantitative finance. In *ACM International Conference on AI in Finance*, 2020.

[82] Laurent Decreusefond et al. Stochastic analysis of the fractional brownian motion. *Potential analysis*, 10(2):177–214, 1999.

[83] M. Rodell, P. R. Houser, U. Jambor, J. Gottschalck, K. Mitchell, C.-J. Meng, K. Arsenault, B. Cosgrove, J. Radakovich, M. Bosilovich, J. K. Entin, J. P. Walker, D. Lohmann, and D. Toll. The global land data assimilation system. *Bulletin of the American Meteorological Society*, 85(3):381–394, 2004.

[84] Alfredo Huete, Kamel Didan, Tomoaki Miura, E Patricia Rodriguez, Xiang Gao, and Laerte G Ferreira. Overview of the radiometric and biophysical performance of the modis vegetation indices. *Remote sensing of environment*, 83(1-2):195–213, 2002.

[85] Laurence Hubert-Moy, Jeanne Thibault, Elodie Fabre, Clémence Rozo, Damien Arvor, Thomas Corpetti, and Sébastien Rapinel. Time-series spectral dataset for croplands in france (2006–2017). *Data in brief*, 27:104810, 2019.

[86] Md Rejaur Rahman, AHMH Islam, and Md Ataur Rahman. Ndvi derived sugarcane area identification and crop condition assessment. *Plan Plus*, 1(2):1–12, 2004.

[87] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[88] Alexander D. Kent. Comprehensive, Multi-Source Cyber-Security Events. Los Alamos National Laboratory, 2015.

[89] Melissa J. M. Turcotte, Alexander D. Kent, and Curtis Hash. *Unified Host and Network Data Set*, chapter 1, pages 1–22. World Scientific, Nov 2018.

[90] DARPA. Operationally Transparent Cyber data release, 2020.

[91] Aaron Walker and Shamik Sengupta. Malware family fingerprinting through behavioral analysis. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–5. IEEE, 2020.

[92] Mark Whitehouse, Marina Evangelou, and Niall M Adams. Activity-based temporal anomaly detection in enterprise-cyber security. In *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, pages 248–250. IEEE, 2016.

[93] Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pages 1–8, 2018.

[94] Elizabeth Riddle-Workman, Marina Evangelou, and Niall M Adams. Adaptive anomaly detection on network data streams. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 19–24. IEEE, 2018.

[95] Maksim E Eren, Juston S Moore, and Boian S Alexandro. Multi-dimensional anomalous entity detection via poisson tensor factorization. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6. IEEE, 2020.

[96] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. *arXiv preprint arXiv:1710.00811*, 2017.

[97] Francesco Sanna Passino and Nicholas A Heard. Classification of periodic arrivals in event time data for filtering computer network traffic. *Statistics and Computing*, 30(5):1241–1254, 2020.

[98] Ronald Holt, Spencer Aubrey, Auston DeVille, William Haight, Todd Gary, and Qingguo Wang. Deep autoencoder neural networks for detecting lateral movement in computer networks. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pages 277–283. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2019.

[99] Cynthia Wagner, Gerard Wagener, Radu State, and Thomas Engel. Malware analysis with graph kernels and support vector machines. In *2009 4th International Conference on Malicious and Unwanted Software (Malware)*, pages 63–68. IEEE, 2009.

[100] Krijn Wijands. Detecting malware using process tree and process activity data. Master's thesis, Technische Universiteit Delft, 2015.

[101] Robert Luh and Sebastian Schrittwieser. Advanced threat intelligence: detection and classification of anomalous behaviour in system processes. *Elektrotechnik & Informationstechnik*, 137:38–44, 2020.

[102] MITRE. CAR data model.

[103] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

[104] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

[105] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*, 2020.

[106] James Morrill, Cristopher Salvi, Patrick Kidger, James Foster, and Terry Lyons. Neural rough differential equations for long time series. *arXiv preprint arXiv:2009.08295*, 2020.

[107] Christophe Reutenauer. Free Lie algebras. In *Handbook of algebra*, volume 3, pages 887–903. Elsevier, 2003.

[108] Youness Boutaib, Lajos Gergely Gyurkó, Terry Lyons, and Danyu Yang. Dimension-free Euler estimates of rough differential equations. *Revue Roumaine de Mathmatiques Pures et Appliques*, 59, 2014.

[109] Patrick Kidger and Terry Lyons. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. *arXiv:2001.00706*, 2020.

[110] Ricky T. Q. Chen. `torchdiffeq`, 2018. `https://github.com/rtqichen/torchdiffeq`.

[111] Terry Lyons. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537*, 2014.

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

[113] Ben Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, pages 109–167, 2010.

[114] Horatio Boedihardjo, Xi Geng, Terry Lyons, and Danyu Yang. The signature of a rough path: uniqueness. *Advances in Mathematics*, 293:720–737, 2016.

[115] Joscha Diehl, Terry Lyons, Rosa Preiß, and Jeremy Reizenstein. Areas of areas generate the shuffle algebra. *arXiv preprint arXiv:2002.02338*, 2020.

[116] Hector Sussmann. A product expansion for the Chen series. In Christopher I Byrnes and Anders Lindquist, editors, *Theory and applications of nonlinear control systems*, pages 323–335. North-Holland, 1986.

[117] Matthias Kawski. Chronological algebras: combinatorics and control. *Geometric control theory (Russian)(Moscow, 1998), ser. Itogi Nauki Tekh. Ser. Sovrem. Mat. Prilozh. Temat. Obz. Moscow: Vseross. Inst. Nauchn. i Tekhn. Inform.(VINITI)*, 64:144–178, 1999.

[118] Eric Gehrig and Matthias Kawski. A Hopf-algebraic formula for compositions of noncommuting flows. In *2008 47th IEEE Conference on Decision and Control*, pages 1569–1574. IEEE, 2008.

[119] Nicolas Bourbaki. *Lie groups and Lie algebras: chapters 2-3*. Springer Science & Business Media, 2008.

[120] C Reutenauer. *Free Lie Algebras*. London Mathematical Society Monographs. Oxford Science Publications, The Clarendon Press, Oxford University Press, 1993.

[121] Terry Lyons et al. Coropa computational rough paths (software library). 2010.