Learning on Sequential Data with Evolution Equations

by

Maud Lemercier

Thesis

Submitted to The University of Warwick

for the degree of

Doctor of Philosophy

Department of Statistics

April 2022

Contents

List of Figures

List of Algorithms

Acknowledgments

Foremost, I am extremely thankful to my supervisor Prof. Theo Damoulas for his unwavering support, encouragement and guidance throughout my PhD. I am deeply grateful to Theo for providing exciting real-world problems which motivated and inspired my research, for always supporting me in exploring new perspectives, and for connecting me with dynamic research communities at The Alan Turing Institute.

I would like to extend my deepest gratitude to Prof. Terry Lyons, and the members of the DataSig programme who welcomed me into the team, gave me opportunities to present my work at multiple workshops and conferences, and generously provided me with computational resources when needed.

My sincere thanks to Dr. Andris Gerasimovičs, Dr. Blanka Horvath, Dr. Chong Liu, Dr. Cristopher Salvi, Dr. Edwin V. Bonilla and Dr. Thomas Cass. Thank you for lending me your expertise. I have been very fortunate to work with you and I have greatly enjoyed our collaborations.

I would also like to thank the Oxford-Warwick Statistical Programme (OxWaSP) for funding my PhD. Thanks to my OxWaSP peers for creating such a friendly working environment during the first year of the CDT.

Special thanks go to Prof. Alessandra Russo, Prof. Francesco P. Andriulli and Prof. Valérie Burdin, who supervised me during my graduate studies in France and the UK, and encouraged me to pursue a PhD.

Last but not least, I would like to thank my parents and Cris for their loving support and encouragement.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself under the supervision of Prof. Theo Damoulas. I also confirm that this thesis has not been submitted for a degree at another university. Parts of this thesis have been previously published by the author:

1. The work presented in Chapter 3 was previously published in *The 38th International Conference on Machine Learning (ICML 2021)* as **SigGPDE: Scaling Sparse Gaussian Processes on Sequential Data** by Maud Lemercier, Cristopher Salvi, Thomas Cass, Edwin V. Bonilla, Theodoros Damoulas and Terry Lyons.

2. The work presented in Chapter 4 was previously published in *The 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021)* as **Distribution Regression for Sequential Data** by Maud Lemercier, Cristopher Salvi, Theodoros Damoulas, Edwin V. Bonilla and Terry Lyons.

3. The work presented in Chapter 5 was previously published in *The 35th Conference on Neural Information Processing Systems (NeurIPS 2021)* as **Higher Order Kernel Mean Embeddings to Capture Filtrations of Stochastic Processes** by Cristopher Salvi, Maud Lemercier, Chong Liu, Blanka Hovarth, Theodoros Damoulas and Terry Lyons.

4. The work presented in Chapter 6 was previously published in *The 36th Conference on Neural Information Processing Systems (NeurIPS 2022)* as **Neural Stochastic Partial Differential Equations: Resolution-Invariant Learning of Continuous Spatiotemporal Dynamics** by Cristopher Salvi, Maud Lemercier and Andris Gerasimovičs.

Abstract

Data which have a sequential structure are ubiquitous in many scientific domains such as physical sciences or mathematical finance. This motivates an important research effort in developing statistical and machine learning models for sequential data. Recently, the signature map, rooted in the theory of controlled differential equations, has emerged as a principled and systematic way to encode sequences into finite-dimensional vector representations. The signature kernel provides an interface with kernel methods which are recognized as a powerful class of algorithms for learning on structured data. Furthermore, the signature underpins the theory of neural controlled differential equations, neural networks which can handle sequential inputs, and more specifically the case of irregularly sampled time-series.

This thesis is at the intersection of these three research areas and addresses key modelling and computational challenges for learning on sequential data. We make use of the well-established theory of reproducing kernels and the rich mathematical properties of the signature to derive an approximate inference scheme for Gaussian processes, Bayesian kernel methods, for learning with large datasets of multi-channel sequential data. Then, we construct new basis functions and kernel functions for regression problems where the inputs are sets of sequences instead of a single sequence. Finally, we use the modelling paradigm of stochastic partial differential equations to design a neural network architecture for learning functional relationships between spatio-temporal signals.

The role of differential equations of evolutionary type is central in this thesis as they are used to model the relationship between independent and dependent signals, and provide tractable algorithms for kernel methods on sequential data.

Acronyms

CDE *Controlled differential equation.*

DR *Distribution regression.*

ELBO *Evidence lower bound.*

GP *Gaussian process.*

KES *Kernel method for DR with the expected signature.*

KL *Kullback-Leibler divergence.*

KME *Kernel mean embedding.*

KRR *Kernel Ridge regression.*

ML-II *Type II maximum likelihood.*

MMD *Maximum mean discrepancy.*

NCDE *Neural controlled differential equation.*

PDE *Partial differential equation.*

RKHS *Reproducing kernel Hilbert space.*

RNN *Recurrent neural network.*

SES *Feature-based method for DR with the pathwise expected signature.*

SigGPDE *Scalable Gaussian process with signature kernel covariance function.*

SPDE *Stochastic partial differential equation.*

VI *Variational inference.*

VOSF *Variational orthogonal signature features.*

Symbols

$\mathbb{N}$	Natural numbers including zero
$\mathbb{N}_*$	Natural numbers excluding zero
$\mathcal{X}, \mathcal{Y}$	Arbitrary sets, e.g. input and output spaces
U	Compact set
V, W, H	Hilbert spaces
$\mathcal{X}(V)$	Set of continuous V-valued paths of bounded variation
$\mathscr{T}(V)$	Tensor algebra of V
$H(V)$	Feature space for embedding V-valued paths
x	Generic input variable
$\boldsymbol{x}$	Path in $\mathcal{X}(V)$
$\mathbf{x}, \mathbf{y}$	Vectors
X, Y	Random variables
$\mathcal{A}, \mathcal{B}$	Elements of a tensor algebra
$\mathcal{P}(\mathcal{X})$	Set of probability measures on $\mathcal{X}$
$\mathbb{P}, \mathbb{Q}$	Probability measures
$\mathbb{P}_X$	Law of the random variable X
$\mathscr{D}_k$	Maximum mean discrepancy associated with the kernel k
$\mu_{\mathbb{P}}$	Kernel mean embedding of the probability measure $\mathbb{P}$
μ_X	Kernel mean embedding of the law of X
$\bar{S}$	Expected signature
$\mathcal{H}$	Hilbert space of functions
$\mathcal{H}_k$	Reproducing kernel Hilbert space (RKHS) associated with k
$\mathcal{H}(V)$	RKHS for embedding V-valued paths
K	Signature kernel
K_{dr}	Distribution regression kernel
u	Solution of a differential equation

Chapter 1

Introduction

Sequential data represent a succession of events or facts, which often follows the order of time, but is not restricted to. For example, in bioinformatics, data in the form of strings of characters are commonplace, as exemplified by DNA sequences, which transcribe the order in which nucleotides are arranged along DNA strands. In Earth sciences, time-stamped data are accumulated at a high rate through the monitoring of various natural resources (e.g. soil moisture, water levels, air pollutant concentrations) with measurements conducted at regular intervals, directed by Earth observation programmes. There is an increasing need to develop statistical and machine learning methods to analyze this type of data and help individuals or organizations make scientific discoveries, predictions, and decisions. Despite the variety of phenomena represented by sequential data and the different data structures used to store and access them, they can be formally described by sequences of the form $(\mathbf{x}_k)_{k\in I}$ with $\mathbf{x}_k \in V$, where the *index set* I is a countable and totally ordered set, and the *state space* V is the set of possible events. The index set is typically a subset of natural numbers or a subset $\{t_1, t_2 \dots, t_\ell\}$ of $\mathbb{R}_+$ with $t_1 < t_2 < \dots < t_\ell$. When the state space is a vector space such as $\mathbb{R}^d$, the corresponding sequences are referred to as *multi-channel sequences* or *multivariate time-series*. In some instances, the state space may be endowed with a different mathematical structure to describe more complex types of observations such as time-varying fMRI scans (Giusti and Lee, 2021; Rieck et al., 2020) or time-series of spatial functions, a.k.a. spatio-temporal data (Chevyrev et al., 2021; Cressie and Wikle, 2015). We denote by $(X_k)_{k\in I}$ a sequential random variable, that is, a discrete-time stochastic process, and by $(\mathbf{x}_k)_{k\in I}$ an observation.

Common learning tasks where the input variable is a multi-channel sequence include: classifying sequential data (e.g. detection of viral sequences (Bzhalava et al., 2018)), predicting a scalar response from an input sequence (e.g. crop yield from series of satellite images (Tan et al., 2021)), or predicting a dependent sequence from an explanatory sequence (e.g. a patient's disease trajectory from sequential medical records (Alaa and van der Schaar, 2019)). Although the first two types of tasks have been extensively studied for multivariate inputs, the models developed in this context are not directly applicable or may be suboptimal when applied to multi-channel sequential data. The latter are (ordered) sets of points that may vary in cardinality. As a result, the same model must be able to process sequences of different lengths. Even with sequences of fixed length ℓ, the input variables may be highly correlated, as repeated observations of a system made sequentially in time, which violates the assumptions of multivariate linear models and poses colinearity problems. Moreover, sequential data are often high-dimensional, with a total number of $\ell \times d$ scalar variables. Contrary to multivariate data, in some instances, the index set may differ from one sequence to another when observations are made at different times. Crucially, the notion of a sequential variable expresses the fact that the ordering of the observations matters, a form of prior knowledge. In other words, changing the order of events in the input sequence is expected to change its effect or response. For instance, the arrangement of nucleotides in a DNA or RNA sequence determines the unique characteristics of a virus. Furthermore, the variables in a multivariate time-series exhibit correlations both along the index dimension and the state space dimension, which are often important predictive patterns. Therefore, models operating on sequential data should capture complex temporal patterns. Eventually, analyzing sequential data goes well beyond multivariate analysis. For this reason, there has been an important research effort to develop statistical and machine learning models that account for the structure of sequential input variables.

One prominent approach consists in finding a representation for the input data such that well-established models and algorithms become applicable. This representation is typically a finite-dimensional vector obtained by feature engineering. This process is often problem specific in the sense that it has to be adapted to different types of datasets, and inevitably incurs some loss of information which is difficult to quantify. Remarkably, over the past few years, the *signature method* has emerged as a principled and systematic feature extraction technique to encode sequences into finite-dimensional vector representations. Before being used as a feature map in machine learning, the signature played an important role in the theoretical analysis of *controlled differential equations* (CDEs), a class of differential equations that model the relationships between evolving systems. Perhaps the most well-known instance of this class is given by *stochastic differential equations* (SDEs) which are a prominent modelling paradigm in physics, engineering, finance and so on, as they describe evolutionary mechanisms under the influence of noise. This grounding in the theory of dynamical systems has several practical implications for machine learning applications. First, it inclines the modeller to think of sequences as functions $\boldsymbol{x} : [0,T] \to V$ of a continuous independent variable $t \in [0,T] \subset \mathbb{R}_+$ which can be thought of as representing time, but may as well be a one-dimensional spatial variable for example. Although it is not possible to measure real-world phenomena continuously, they often unravel continuously over time. This continuous-time view has several advantages compared to the discrete conception of sequential data, so we choose to stress the distinction by writing $\boldsymbol{x}(t)$ instead of $\boldsymbol{x}_t$ or $\mathbf{x}_t$. Considering sequential data as the evaluation of a function at different positions makes it possible to handle different index sets and deal with high temporal resolutions. This idea is also at the heart of *Functional Data Analysis* (FDA) (Ramsay et al., 2005). However, the signature method stands out in the context of learning on sequential data, as the focus is not on representing the time-series themselves, but on representing *functions* on time-series, with good approximation guarantees resulting from the algebraic and analytic properties of the signature. Besides, the real power of the signature arises when the time-series are multidimensional as it captures the order of events across time and channel dimensions.

By the same token, there has been a significant effort in designing *kernel functions* on sequential data, as they provide an interface with kernel methods, a well-established class of algorithms for learning on structured data. Popular algorithms include *Support Vector Machines* (SVMs), *Kernel Ridge Regression* (KRR) and *Gaussian Process Regression* (GPR), which can be transferred to virtually any input space, via a carefully designed kernel function on this space. In particular, positive definite kernels is an important class of kernel functions as they make it possible to leverage the theory of *Reproducing kernel Hilbert space*s (RKHSs) (Steinwart and Christmann, 2008). On the one hand, defining a positive definite kernel on sequences is not easier than (in some sense, it is equivalent to) constructing a feature map. On the other hand, it makes it possible to implicitly work with high-dimensional and possibly infinitely many features, since kernelized algorithms only require the ability to compute inner products in the feature space. A way to evaluate the inner product without materializing the features is called a *kernel trick*. For this reason, various kernel functions have been engineered for sequential data. They can be broadly classified into three categories. So-called *String kernels* (Leslie et al., 2003; Lodhi et al., 2002) are based on matching common substructures in a sequence. Another idea is to fit a generative probabilistic model to each input time-series, and use it to define kernel functions including *Probability Product kernels* (Jebara et al., 2004), *Autoregressive kernels* (Cuturi and Doucet, 2011), or *Fisher kernels* (Jaakkola et al., 2000). Alternatively, kernels can be contructed from similarity measures between time-series such as *Dynamic Time Warping* (DTW) (Cuturi et al., 2007). Each of these approaches has shortcomings, which are remarkably overcome by the *signature kernel* (Cass et al., 2020; Király and

Oberhauser, 2019). The latter can be applied to real-valued time-series (which is not the case of string kernels), does not rely on strong assumptions about the generating process (contrary to probability product kernels), and is positive definite (contrary to DTW-based kernels). Derived from the signature map, it naturally inherits several desirable properties which guarantee effective learning on sequential data. From a computational point of view, it makes it possible to implicitly use a high (possibly infinite) number of signature features.

With the surge of the deep learning modelling paradigm, neural network architectures have been developed to process sequential inputs, *Recurrent neural network*s (RNNs) (Rumelhart et al., 1986) being one of the most popular models. Compared to the aforementioned approaches, the features (or basis functions) need not be fixed and can be adapted to the problem at hand by optimizing a loss function through gradient-based backpropagation algorithms. RNNs repeatedly apply the same operation to each term of a sequence, making it possible to reduce the number of parameters and process sequences of different lengths. These recurrent systems can be seen as discrete-time dynamical systems, and require that the observations be regularly spaced in time. Recently, a continuous-time analogue of RNNs, termed *Neural controlled differential equation*s (NCDEs), has been introduced in Kidger et al. (2020). NCDEs are capable of processing irregularly observed time-series. As their name suggests, NCDEs are based on the formalism of controlled differential equations, and their expressive power can be explained using the properties of the signature, as may be expected.

Despite this progress, still various challenges remain when it comes to the analysis of real-world datasets of sequences. First, in order to exploit the availability of large-scale data, one needs to develop models which remain computationally tractable when the number of training instances is very large. This has motivated the development of approximate inference frameworks to mitigate the prohibitive cubic time complexity of some kernel methods. However, transferring these approaches to sequential data is not always straightforward or optimal. Second, in regression analysis, the goal is to first model the relationship between a response variable and an explanatory multi-channel sequence, and then fit the model using training examples in the form of input-output pairs. A common situation, which does not quite fit this learning framework, arises when labels are only available at the level of *sets* of sequences, rather than a single sequence. When the ensemble of sequences can be seen as a set of replicates of samples from a stochastic process, one needs to be able to represent and learn on (laws of) stochastic processes. Although fruitful directions have been proposed to address this problem for multivariate random variables, stochastic processes are not just a collection of one-dimensional marginals. In fact, compared to multivariate random variables, stochastic processes have a much richer structure that goes even beyond their laws. Third, when the set of sequences exhibits spatial dependencies, and is no longer an unordered collection, this set may be better modelled as a single observation from a spatio-temporal stochastic process, that is, a time-series of functions. Although the signature applies in theory to infinite-dimensional state spaces, this setting is challenging in practice, as the functions are observed discretely, which poses scalability and modelling issues when it comes to accounting for complex spatial dependencies. In this thesis, we propose methodologies to address these challenges using the formalisms of reproducing kernels, signatures and deep learning, and their connections with evolution equations, differential equations involving time.

1.1 Outline of the thesis

In Chapter 2 we provide some necessary background for this thesis. Each subsequent chapter can be read independently.

Chapter 3 is concerned with Bayesian kernel methods for sequential data, in which the signature kernel is used as a covariance function to define Gaussian process models. We combine two somewhat orthogonal views of the signature kernel—on the one hand its explicit feature expansion, on the other hand the kernel trick that relies on solving a partial differential equation—to develop a fast approximate inference scheme. This allows fitting Gaussian process models and performing model selection on large datasets of multivariate time-series which were previously intractable.

In Chapter 4 we construct a set of features and a kernel—respectively extending the signature and the signature kernel—that operate on probability measures corresponding to the laws of stochastic processes. In practice, this allows one to conduct *Distribution regression* (DR), a type of regression analysis where the independent variable is a set of multivariate time-series, viewed as an empirical or discrete measure, and the dependent variable is a single scalar.

In Chapter 5 we leverage the notion of conditional kernel mean embeddings from the theory of RKHSs to construct a more expressive kernel on stochastic processes, which captures information that goes beyond their law. Capitalizing on the kernel trick, we provide practical algorithms for applications ranging from DR to hypothesis testing with kernel two-sample tests.

In Chapter 6 we extend NCDEs to model the functional relationship between discretely observed spatio-temporal signals. We do so by blending an important class of models in physics, known as *Stochastic partial differential equations* (SPDEs), with the high expressivity and flexibility of deep learning.

Chapter 7 concludes by summarizing the contributions of this thesis and discussing potential future avenues of research.

Chapter 2

Background

The objective of this chapter is to provide some necessary background on the mathematical tools which underpin the methodologies developed in this thesis. To this aim, we start by introducing the *signature map*, a central object in the theory of rough paths in stochastic analysis, which also provides a systematic way to extract features from data with a sequential structure that can be easily exploited by machine learning models. In particular, we will explain how the mathematical properties of the signature map yield canonical basis function expansions for approximating real-valued functions on sequential data. We then provide some background on kernel methods and the theory of RKHSs which can be transferred to sequential data leveraging the *signature kernel*, another prominent but more recent tool in the theory of rough paths. Finally, we give a brief account on NCDEs, continuous-time analogues of RNNs.

The goal of this thesis is to develop methodologies for learning tasks where the input variable has a sequential structure. To fix the notation, each input will be a sequence $(\mathbf{x}_k)_{k=1}^{\ell}$ in the form of a collection of points $\hat{\mathbf{x}}_k \in V$ with corresponding time stamps $t_k \in \mathbb{R}_+$ such that $0 = t_1 < ... < t_\ell = T$ and

$$(\mathbf{x}_k)_{k=1}^{\ell} = \left((t_1, \hat{\mathbf{x}}_1), \ldots, (t_\ell, \hat{\mathbf{x}}_\ell)\right). \tag{2.1}$$

We assume that V is a vector space, and represent the sequence as a *path*, a continuous function $\boldsymbol{x} : [0, T] \to V$ such that $\boldsymbol{x}(t_k) = \hat{\mathbf{x}}_k$. Such embeddings can be obtained by various interpolation methods (e.g. linear interpolation, cubic splines), but other transformations may be considered, as discussed in Fermanian (2019). When the data arise from the monitoring of a continuous-time process, an interpolation provides an approximation of the underlying process. While these approximations can be the object of a study, in this thesis, this step is considered as a representation of sequential data, rather than an imputation.

2.1 CDEs, signature and regression on paths

In supervised learning the goal is to find, given a dataset $\{(x_i, y_i)\}_{i=1}^n$ of input-output pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, an approximation g to the function $f : \mathcal{X} \to \mathcal{Y}$ that underlies the predictive relationship between the input (or independent variable) and output (or dependent variable), such that $g(x_*)$ can be used to predict the response for an unseen input $x_* \in \mathcal{X}$. In some instances, the input-output relationship can be approximately described by a mechanistic model, usually in the form of a system of differential equations. When we do not know the data generating process, a class of useful approximators for nonlinear real-valued functions can be expressed as a linear basis expansion

$$f_\theta(x) = \theta_0 + \sum_{k=1}^{K} \theta_k \phi_k(x), \tag{2.2}$$

that is, a linear combination of $K \in \mathbb{N}_*$ known basis functions $\phi_k : \mathcal{X} \to \mathbb{R}$ (the bias term corresponds to $\phi_0 \equiv 1$). The basis functions should be easy to evaluate and such that any continuous target function can be approximated arbitrarily well by a finite linear combination of these basis functions. This type of guarantee is known as the *universal approximation property.* More formally, we say that the basis functions $\{\phi_k \mid k \in \mathbb{N}\}$ have the universal approximation property if it is guaranteed that for any continuous $f : \mathcal{X} \to \mathbb{R}$ and any scalar $\epsilon > 0$ there exists a g of the form (2.2) such that $||f - g||_\infty < \epsilon$. The set of functions of the form (2.2), that is linear combinations of basis functions, will be denoted by $\mathcal{F} = \text{span}\{\phi_k \mid k \in \mathbb{N}\}$.

The celebrated Stone-Weierstrass theorem provides sufficient conditions for guaranteeing the universal approximation property, as stated below.

Theorem 2.1.1. *Let (U, d) be a compact metric space and $\mathcal{F} \subset \mathscr{C}(U, \mathbb{R})$ be an algebra. If $\mathcal{F}$ is such that*

1. *for all $x \in U$, there exists an $f \in \mathcal{F}$ with $f(x) \neq 0$*
2. *for all $x, y \in U$ with $x \neq y$, there exists an $f \in \mathcal{F}$ with $f(x) \neq f(y)$*

then $\mathcal{F}$ is dense in $\mathscr{C}(U, \mathbb{R})$.

Basis functions on Euclidean spaces

The monomial basis is one of the most familiar example when the input space $\mathcal{X}$ is an interval of $\mathbb{R}$, with $\phi_0 \equiv 1$ and

$$\phi_1(x) = x,\ \phi_2(x) = x^2,\ \phi_3(x) = x^3,\ \dots\ ,\ \phi_k(x) = x^k,\ \dots\ .$$

The Fourier basis is another common example when the input space is the real line and the target function is periodic (e.g. $\phi_{2k-1}(x) = \sin(kx)$, $\phi_{2k}(x) = \cos(kx)$ for 2π-periodic functions). These bases can be extended to multivariate inputs. For example, the multivariate monomial basis is given by

$$\phi_{i_1 \ldots i_k}(\mathbf{x}) = x_{i_1} \ldots x_{i_k}, \qquad \mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d, \tag{2.3}$$

where $i_1, \ldots, i_k \in \{1, \ldots, d\}$. The corresponding basis expansions read as

$$f_\theta(\mathbf{x}) = \theta_0 + \sum_{k=1}^{K} \sum_{i_1,\ldots,i_k} \theta_{i_1\ldots i_k} \phi_{i_1\ldots i_k}(\mathbf{x}),$$

where the inner sum is taken over $i_1, \ldots, i_k \in \{1, \ldots, d\}$. These basis functions have the aforementioned universal approximation property as per the Stone-Weierstrass theorem. In this section, we will introduce a set of basis functions which generalize the monomials in eq. (2.3) when the input space is a suitable space of paths. Instead of starting with the definition of these basis functions, we shall explain how they are intimately connected to differential equations, as this link will be further exploited in the subsequent chapters of this thesis.

Defining the input space

So far we have embedded sequences in the form of eq. (2.1) into continuous paths $\boldsymbol{x} : [0, T] \to V$. We start by adding more structure and regularity constraints to the paths we consider. This will allow us to give a meaning to a certain type of differential equations and define a set of basis functions for modelling real-valued functions on paths. First, we consider paths $\boldsymbol{x} : [0, T] \to V$ which evolve in a real Hilbert space V with inner product $\langle \cdot, \cdot \rangle_V$. In view of practical applications and computations, an important example is the case $V = \mathbb{R}^d$. However, the mathematical objects that we will introduce in this section can be defined when the state space V is infinite-dimensional and we will see in the next section how to handle this situation computationally. Second, we consider paths which are of bounded variation, as defined hereafter.

Definition 1 (Path of bounded variation)**.** *We say that a path $\boldsymbol{x} : [0, T] \to V$ is of bounded variation if*

$$\mathcal{V}(\boldsymbol{x}) := \sup_{\mathcal{D}} \sum_{t_i \in \mathcal{D}} \|\boldsymbol{x}(t_{i+1}) - \boldsymbol{x}(t_i)\|_V < +\infty,$$

where $\|\cdot\|_V$ denotes the norm induced by the inner product on V and the supremum is taken over

$$\mathcal{D} \in \{(t_1, \ldots, t_n) \mid n \geq 1,\ 0 = t_1 < \ldots < t_n = T\},$$

all finite partitions of $[0, T]$.

We denote a set of V-valued continuous paths of bounded variation by $\mathcal{X}(V)$

$$\mathcal{X}(V) = \{\boldsymbol{x} \in \mathscr{C}([0,T], V) \mid \mathcal{V}(\boldsymbol{x}) < +\infty\}\,. \tag{2.4}$$

We note that the piecewise linear embedding of a sequence with d channels is in $\mathcal{X}(\mathbb{R}^d)$ and $\mathcal{X}(V)$ is a Banach space with norm $\|\cdot\|$ defined for all $\boldsymbol{x} \in \mathcal{X}(V)$ by

$$\|\boldsymbol{x}\| = \mathcal{V}(\boldsymbol{x}) + \sup_{t\in[0,T]} \|\boldsymbol{x}(t)\|_V\,. \tag{2.5}$$

2.1.1 Controlled differential equations

The theory of rough paths is concerned with the study of *Controlled differential equations* (CDEs), an important class of differential equations that model the effect of an input signal $\boldsymbol{x} : [0,T] \to V$ on a response signal $\boldsymbol{y} : [0,T] \to W$. We assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are continuous and consider a continuous function $F : W \to L(V, W)$, where $L(V, W)$ denotes the set of continuous linear mappings from V to W. We say that $\boldsymbol{x}, \boldsymbol{y}$ and F solve the controlled differential equation

$$d\boldsymbol{y} = F(\boldsymbol{y})d\boldsymbol{x}, \quad \boldsymbol{y}(0) = \mathbf{a}, \tag{2.6}$$

if they satisfy the following integral equation for every $t \in [0, T]$

$$\boldsymbol{y}(t) = \mathbf{a} + \int_0^t F\left(\boldsymbol{y}(s)\right) d\boldsymbol{x}(s). \tag{2.7}$$

If $\boldsymbol{x} \in \mathcal{X}(V)$, the integral on the right hand side of eq. (2.7) can be interpreted in the Riemann-Stieltjes sense. Furthermore, when F is Lipschitz continuous, the solution is unique (Lyons et al., 2007, Thm 1.3). Such differential equations may be used to model functions on paths, since each $(\mathbf{a}, F)$ induces a mapping f between $\mathcal{X}(V)$ and W (think for example $W = \mathbb{R}^e$) given by

$$\begin{aligned} f : \mathcal{X}(V) &\to W \\ \boldsymbol{x} &\mapsto \mathbf{a} + \int_0^T F\left(\boldsymbol{y}(s)\right) d\boldsymbol{x}(s). \end{aligned}$$

Besides this map is continuous (w.r.t $\|\cdot\|$). We note that rough path theory extends these results for less regular input paths $\boldsymbol{x}$ (such as Brownian motion paths in SDEs) and refer the interested reader to Lyons (1998); Lyons et al. (2007). More precisely, by parametrising the *vector field* F one can obtain a hypothesis space of functions on $\mathcal{X}(V)$. We will discuss this approach in the last section of this chapter. For now, we show that when F is linear, a sequence of iterated integrals naturally appears. For simplicity, we assume that $V = \mathbb{R}^d$

and $W = \mathbb{R}^e$. A CDE with linear vector field reads as

$$d\boldsymbol{y} = A\boldsymbol{y}d\boldsymbol{x}, \quad \boldsymbol{y}(0) = \mathbf{a},$$

where A is a linear map from $\mathbb{R}^e$ to $\mathbb{R}^{d\times e}$. However, by interpreting A as a map which acts on $d\boldsymbol{x}(t)$ and returns a function in $\mathscr{C}(\mathbb{R}^e, \mathbb{R}^e)$, we can express the response at final time by

$$\boldsymbol{y}(T) = \Big(\mathbf{I}_e + \sum_{k=1}^{\infty} A^{\otimes k}\left(S_k(\boldsymbol{x})\right)\Big)\mathbf{a}, \tag{2.8}$$

where $S_k(\boldsymbol{x})$ denotes the tensor in $(\mathbb{R}^d)^{\otimes k}$ (tensor product of $\mathbb{R}^d$ with itself k times), defined as the following k-fold iterated integral

$$S_k(\boldsymbol{x}) = \underset{0<t_1<\ldots<t_k<T}{\int\cdots\int} d\boldsymbol{x}(t_1)\otimes\ldots\otimes d\boldsymbol{x}(t_k) \tag{2.9}$$

and $A^{\otimes k}(\mathbf{v}_1\otimes\ldots\otimes\mathbf{v}_k) = A(\mathbf{v}_1)\ldots A(\mathbf{v}_k)$ for any $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{R}^d$. The convergence of the series in eq. (2.8) is ensured by the factorial decay of the norm of $S_k(\boldsymbol{x})$ in $(\mathbb{R}^d)^{\otimes k}$ (Lyons et al., 2007, Proposition 2.2): for any $\boldsymbol{x} \in \mathcal{X}(V)$

$$\|S_k(\boldsymbol{x})\| \leq \frac{\mathcal{V}(\boldsymbol{x})^k}{k!}. \tag{2.10}$$

The expansion in eq. (2.8) tells us that given the iterated integrals $S_1(\boldsymbol{x}), S_2(\boldsymbol{x}), \ldots$ of the path $\boldsymbol{x}$, we can determine the response of any linear controlled differential equation driven by $\boldsymbol{x}$. For a path with d coordinates $\boldsymbol{x} : t \mapsto (x^1(t), \ldots, x^d(t))$, the vector $S_1(\boldsymbol{x})$ is of dimension d, and is given by

$$S_1(\boldsymbol{x}) = \int_0^T d\boldsymbol{x}(t) = \left(x^i(T) - x^i(0)\right)_{1\leq i\leq d}.$$

The i^{th} entry of this vector corresponds to the increment of the i^{th} coordinate of $\boldsymbol{x}$ between $t = 0$ and $t = T$. The tensor $S_2(\boldsymbol{x})$ can be seen as a $d \times d$ matrix

$$S_2(\boldsymbol{x}) = \underset{0<s<t<T}{\int\int} d\boldsymbol{x}(s)\otimes d\boldsymbol{x}(t) = \left(\int_0^T \left(x^i(t) - x^i(0)\right) dx^j(t)\right)_{1\leq i,j\leq d},$$

whose components also have a geometrical meaning (see the next section).

2.1.2 The signature

So far we have seen how the k-fold iterated integrals of a path $\boldsymbol{x} \in \mathcal{X}(V)$ in eq. (2.9) appear in the expansion of the solution of a linear differential equation controlled by $\boldsymbol{x}$. We will now explain the role of these iterated integrals in the construction of basis functions for approximating nonlinear real-valued functions on paths. To this aim, we introduce the *signature map*, a central object in rough path theory, which maps a path to its collection of iterated integrals. The signature maps a path into the following space of sequences

$$\mathscr{T}(V) = \left\{\mathcal{A} = (A_0, A_1, A_2, \dots) \mid \forall k \geq 0,\ A_k \in V^{\otimes k}\right\}. \tag{2.11}$$

Definition 2 (Signature)**.** *The signature is the map defined by*

$$\begin{aligned} S : \mathcal{X}(V) &\to \mathscr{T}(V) \\ \boldsymbol{x} &\mapsto (1, S_1(\boldsymbol{x}), S_2(\boldsymbol{x}), \dots) \end{aligned}$$

where the k^{th} term in the sequence is the order-k tensor in $V^{\otimes k}$,

$$S_k(\boldsymbol{x}) = \underset{0<t_1<\ldots<t_k<T}{\int \dots \int} d\boldsymbol{x}(t_1) \otimes \ldots \otimes d\boldsymbol{x}(t_k) \tag{2.12}$$

which is referred to as the k^{th} level of the signature.

The signature of $\mathbb{R}^d$-valued paths

The above definition may appear rather abstract. However, as we have already remarked, when $V = \mathbb{R}^d$, each term $S_k(\boldsymbol{x}) \in (\mathbb{R}^d)^{\otimes k}$ of the signature (each k-fold iterated integral) can be represented by its coordinates with respect to the basis $(\mathbf{e}_{i_1} \otimes \ldots \otimes \mathbf{e}_{i_k})_{(i_1,\ldots,i_k)\in\{1,\ldots,d\}^k}$ of $(\mathbb{R}^d)^{\otimes k}$, such that

$$S_k(\boldsymbol{x}) = \sum_{i_1\ldots i_k} S^{i_1\ldots i_k}(\boldsymbol{x})\mathbf{e}_{i_1} \otimes \ldots \otimes \mathbf{e}_{i_k}, \tag{2.13}$$

where $(\mathbf{e}_i)_{i\in\{1,\ldots,d\}}$ is the standard basis of $\mathbb{R}^d$, and each coordinate is given by

$$S^{i_1\ldots i_k}(\boldsymbol{x}) = \underset{0<t_1<\ldots<t_k<T}{\int \dots \int} dx^{i_1}(t_1) \ldots dx^{i_k}(t_k), \tag{2.14}$$

where we have used the fact that $\boldsymbol{x}(t) = \sum_{i=1}^d x^i(t)\mathbf{e}_i$. In other words, the k^{th} level of the signature can be seen as a $\underbrace{d \times \ldots \times d}_{k \text{ times}}$ array. See Fig. 2.1 for a geometric interpretation of the coordinates of the 2^{nd} level of the signature.

Our goal is to identify a set of basis functions to expand functions on $\mathcal{X}(\mathbb{R}^d)$. We will now explain why the real-valued functions $\boldsymbol{x} \mapsto S^{i_1\cdots i_k}(\boldsymbol{x})$ are suitable basis functions. To this aim, we shall state the core properties of the signature that allow us to verify the conditions of Thm. 2.1.1.

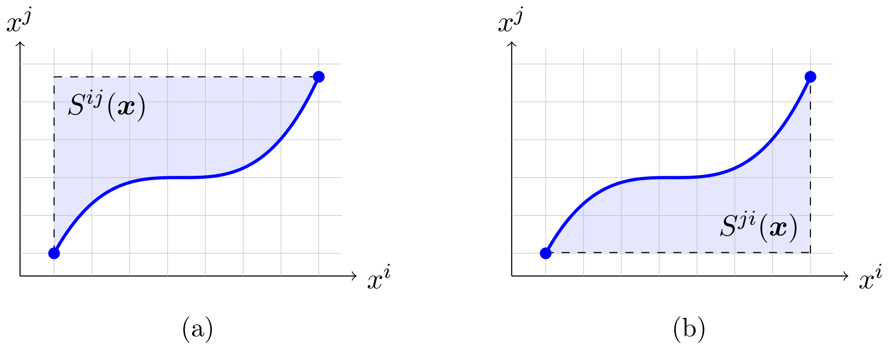


Figure 2.1: Geometric meaning of the coordinates of the 2nd level of the signature of a d-dimensional path $\boldsymbol{x}$. The coordinates $S^{ij}(\boldsymbol{x})$ and $S^{ji}(\boldsymbol{x})$ correspond to the (light blue) area enclosed between the (bold blue) curve $t \mapsto (x^i(t), x^j(t))$ and two perpendicular (dashed) lines passing through the endpoints of $\boldsymbol{x}$.

Co-domain structure

The co-domain $\mathscr{T}(V)$ of the signature map (see eq. (2.11)) is endowed with two internal operations: an addition and a product. For any two elements $\mathcal{A} = (A_0, A_1, \ldots)$ and $\mathcal{B} = (B_0, B_1, \ldots)$ of $\mathscr{T}(V)$ and any scalar $\lambda \in \mathbb{R}$

$$\lambda\mathcal{A} + \mathcal{B} = (\lambda A_0 + B_0,\, \lambda A_1 + B_1, \ldots) \tag{2.15}$$

$$\mathcal{A} \otimes \mathcal{B} = (C_0, C_1, \ldots) \quad \text{with} \quad C_n = \sum_{k=0}^{n} A_k \otimes B_{n-k}. \tag{2.16}$$

The space $\mathscr{T}(V)$ endowed with these operations is a (non-commutative) algebra $(\mathscr{T}(V), \cdot, +, \otimes)$, and $\mathbf{1} = (1, 0, 0, \ldots)$ is an identity element with respect to $\otimes$.

Invariances

Models that capture the invariances of a problem often perform better than those that ignore them (Goodfellow et al., 2009; van der Wilk et al., 2018). Remarkably, the signature is invariant to two important types of transformations.

First, the signature is invariant to time-reparametrizations. More precisely, let I and J be two closed intervals of $\mathbb{R}_+$ and $\tau : J \to I$ be a strictly increasing function. Consider the paths $\boldsymbol{x} : I \to V$ and $\tilde{\boldsymbol{x}} : J \to V$ that are related by $\tilde{\boldsymbol{x}} : t \mapsto \boldsymbol{x}(\tau(t))$. Then, a change of variables in eq. (2.12) yields $S(\tilde{\boldsymbol{x}}) = S(\boldsymbol{x})$. This means that the signature is insensitive to the speed at which the path unravels through time. Many real-world problems exhibit this parametrization invariance. For example, in character recognition tasks, the speed at which a character is drawn is irrelevant.

Second, the signature is invariant to translations. In other words, the signature does not retain information about the initial point of a path. More formally, let $\mathbf{a} \in V$, and consider the path $\tilde{\boldsymbol{x}} : t \mapsto \boldsymbol{x}(t) + \mathbf{a}$. Then $S(\tilde{\boldsymbol{x}}) = S(\boldsymbol{x})$.

Injectivity of the signature

In light of the above, two paths $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ in $\mathcal{X}(V)$ can have the same signature; the signature is not injective. A characterization of paths which have the same signature as $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^d)$ (up to translations) is given in Hambly and Lyons (2010). To ensure the injectivity of the signature (that to a signature corresponds a unique path), it suffices to consider all paths in $\mathcal{X}(V)$ which start at the same point (say $\mathbf{0} \in V$) and add a monotonically increasing coordinate (for example time). For this reason, we introduce the following set of paths

$$\widetilde{\mathcal{X}}(\mathbb{R} \oplus V) = \left\{ \tilde{\boldsymbol{x}} : t \mapsto (t, \boldsymbol{x}(t)) \;\middle|\; \boldsymbol{x} \in \mathcal{X}(V),\ \boldsymbol{x}(0) = \mathbf{0} \right\}.$$

When V is $\mathbb{R}^{d-1}$ we may simply write

$$\widetilde{\mathcal{X}}(\mathbb{R}^d) = \left\{ \tilde{\boldsymbol{x}} : t \mapsto (t, x^1(t), \dots, x^{d-1}(t)) \;\middle|\; \boldsymbol{x} \in \mathcal{X}(\mathbb{R}^{d-1}),\ \boldsymbol{x}(0) = \mathbf{0} \right\}.$$

The injectivity of the signature restricted to $\widetilde{\mathcal{X}}(\mathbb{R} \oplus V)$ ensures point 2. in the Stone-Weierstrass theorem (Thm. 2.1.1).

Linear functionals on the signature

In order to use a Stone-Weierstrass argument (Thm. 2.1.1) and establish the universal approximation property of our set of basis functions, we need to verify that the span of the latter forms an algebra $\mathcal{F}$. In particular the product of any two elements of $\mathcal{F}$ must be an element of $\mathcal{F}$. In the sequel, we will define $\mathcal{F}$ and state the theorems that establish that $\mathcal{F}$ is closed under multiplication.

Definition 3 (Coordinate iterated integrals)**.** *Let V be a Hilbert space of dimension $d \in \mathbb{N}_*$ and $(\mathbf{e}_i)_{i \in \{1,\dots,d\}}$ be a basis of V. Let $(e^{i_1 \dots i_k})_{i_1,\dots,i_k \in \{1,\dots,d\}^k}$ be the dual basis of $(V^{\otimes k})'$. The coordinate iterated integrals are the maps*

$$\begin{aligned} S^{i_1 \dots i_k} : \mathcal{X}(V) &\to \mathbb{R} \\ \boldsymbol{x} &\mapsto e^{i_1 \dots i_k}(S(\boldsymbol{x})) \end{aligned}$$

where we use the natural extension $e^{i_1 \dots i_k}(\mathcal{A}) = e^{i_1 \dots i_k}(A_k)$, for $\mathcal{A} \in \mathscr{T}(V)$.

Before stating an identity which ensures that the span of coordinate iterated integrals is closed under multiplication, we define the shuffle product of two multi-indices.

Definition 4 (Shuffle product)**.** *Given two multi-indices $(i_1, \dots, i_m)$ and $(j_1, \dots, j_n)$, consider the multi-index $(r_1, \dots, r_{m+n}) = (i_1, \dots, i_m, j_1, \dots, j_n)$. The shuffle product of $(i_1, \dots, i_m)$ and $(j_1, \dots, j_n)$ is the set defined by*

$$(i_1, \dots, i_m) \sqcup\!\sqcup (j_1, \dots, j_n) = \{ r_{\sigma(1)}, \dots, r_{\sigma(m+n)} \mid \sigma \in \textit{Shuffles}(m, n) \},$$

where Shuffles(m,n) denotes the collection of all permutations of $\{1,\ldots,m+n\}$ which satisfy $\sigma^{-1}(1) < \ldots < \sigma^{-1}(m)$ and $\sigma^{-1}(m+1) < \ldots < \sigma^{-1}(m+n)$.

Theorem 2.1.2 (Shuffle identity)**.** *For any two multi-indices $(i_1,\ldots,i_m)$ and $(j_1,\ldots,j_n)$ the product of the corresponding iterated integrals can be written as a sum of higher-order iterated integrals,*

$$S^{i_1\ldots i_m}(\boldsymbol{x})S^{j_1\ldots j_n}(\boldsymbol{x}) = \sum_{k_1\ldots k_{m+n}} S^{k_1\ldots k_{m+n}}(\boldsymbol{x}) \tag{2.17}$$

where the sum is taken over all multi-indices $(k_1,\ldots,k_{m+n})$ of length $m+n$ such that $(k_1,\ldots,k_{m+n}) \in (i_1,\ldots,i_m) \sqcup\!\sqcup (j_1,\ldots,j_n)$ as defined in Def. 4.

Therefore, any polynomial expression in the terms of the signature can be rewritten as a linear expression in higher-order terms of the signature. This result is extended to the signature of paths valued in a possibly infinite-dimensional space in Cass et al. (2016, Corollary 3.9).

Theorem 2.1.3 (Close under multiplication)**.** *Let $\mathcal{L}_m \in (V^{\otimes m})'$ and $\mathcal{L}_n \in (V^{\otimes n})'$. Then there exists $\mathcal{L}_{m+n} \in (V^{\otimes(m+n)})'$ such that for all $\boldsymbol{x} \in \mathcal{X}(V)$*

$$\mathcal{L}_m(S(\boldsymbol{x}))\mathcal{L}_n(S(\boldsymbol{x})) = \mathcal{L}_{m+n}(S(\boldsymbol{x})),$$

where we have used the natural extension $\mathcal{L}_m(\mathcal{A}) = \mathcal{L}_m(A_m)$, for $\mathcal{A} \in \mathscr{T}(V)$.

2.1.3 Signature-based basis functions

The properties that we have stated make it possible to apply Thm. 2.1.1 and obtain the following density result for the algebra of functions

$$\mathcal{F} = \operatorname{span}\left\{\boldsymbol{x} \mapsto \mathcal{L}_k \circ S(\boldsymbol{x}) \;\middle|\; \mathcal{L}_k \in (V^{\otimes k})',\ k \in \mathbb{N}\right\}.$$

Theorem 2.1.4. *Let $U \subset \widetilde{\mathcal{X}}(\mathbb{R} \oplus V)$ be a compact set of paths and consider a continuous function $f : U \to \mathbb{R}$. Then for any $\epsilon > 0$ there exists an element $g \in \mathcal{F}$, that is a function of the form*

$$g = \sum_{k=0}^{K} \mathcal{L}_k \circ S,$$

such that $\|f-g\|_\infty < \epsilon$. In other words, $\mathcal{F}$ is dense in $\mathscr{C}(U,\mathbb{R})$.

If $V = \mathbb{R}^d$, we can write more explicitly the space of approximators as

$$\mathcal{F} = \operatorname{span}\left\{\boldsymbol{x} \mapsto e^{i_1\ldots i_k} \circ S(\boldsymbol{x}) \mid i_1,\ldots,i_k \in \{1,\ldots,d\},\ k \in \mathbb{N}\right\}.$$

According to Thm. 2.1.4, there exists a truncation level $K \geq 0$ and scalar coefficients $\theta_0,\ \theta_{i_1 \dots i_k} \in \mathbb{R}$ such that for any path $\boldsymbol{x} \in U$, the following holds

$$\left| f(\boldsymbol{x}) - \theta_0 - \sum_{k=1}^{K} \sum_{i_1 \dots i_k} \theta_{i_1 \dots i_k} S^{i_1 \dots i_k}(\boldsymbol{x}) \right| < \epsilon.$$

In light of this result, to learn a regression function from input-output pairs $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n) \in \widetilde{\mathcal{X}}(\mathbb{R}^d) \times \mathbb{R}$, we start by replacing the inputs $\boldsymbol{x}_i$ with the new variables

$$\Phi_i := \operatorname{vec}\left((1, S_1(\boldsymbol{x}_i), \dots, S_K(\boldsymbol{x}_i)) \right),$$

to obtain the mapped dataset $(\Phi_1, y_1), \dots, (\Phi_n, y_n) \in \mathbb{R}^e \times \mathbb{R}$ with dimension $e = (d^{K+1} - 1)/(d-1)$ where $d > 1$.

Remark. *In practice, the inputs are more realistically time-series. We have already discussed the transformation of time-series $(\mathbf{x}_k)_{k=1}^{\ell}$ of the form of eq.* (2.1) *into a path $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^{d-1})$. In order to map time-series into $\widetilde{\mathcal{X}}(\mathbb{R}^d)$ it suffices to add the point $(t_0, \mathbf{0})$ such that $(\mathbf{x}_k)_{k=1}^{\ell} \mapsto ((t_0, \mathbf{0}), (t_1, \hat{\mathbf{x}}_1), \dots, (t_\ell, \hat{\mathbf{x}}_\ell))$ and consider the path $\boldsymbol{x}(t_k) = (t_k, \hat{\mathbf{x}}_k)$ for all $k \in \{0, \dots, \ell\}$.*

From there, it remains to compute the Φ_i. Another algebraic property of the signature map plays an important role in addressing this practical point: Chen's theorem allows one to compute the signature of piecewise linear paths. Before stating this result, we define the concatenation operation between paths.

Definition 5 (Concatenation)**.** *Let $\boldsymbol{x} : [0, s] \to V$ and $\boldsymbol{y} : [s, t] \to V$ be two continuous paths. Their concatenation is the path $\boldsymbol{x} \star \boldsymbol{y} : [0, t] \to V$ defined by*

$$(\boldsymbol{x} \star \boldsymbol{y})(u) = \begin{cases} \boldsymbol{x}(u) & \text{if } u \in [0, s] \\ \boldsymbol{x}(s) + \boldsymbol{y}(u) - \boldsymbol{y}(s) & \text{if } u \in [s, t]. \end{cases}$$

Chen's theorem (Chen, 1958) states that the signature of the concatenation of two paths is given by the tensor product (see eq. (2.16)) of their signatures.

Theorem 2.1.5. *Let $\boldsymbol{x} : [0, s] \to V$ and $\boldsymbol{y} : [s, t] \to V$ be two continuous paths of bounded variation. Then $S(\boldsymbol{x} \star \boldsymbol{y}) = S(\boldsymbol{x}) \otimes S(\boldsymbol{y})$.*

The practical importance of this theorem lies in the fact that a piecewise linear path with ℓ knots is the concatenation of $\ell - 1$ linear paths, and that for a linear path $\boldsymbol{x} : t \mapsto \mathbf{a} + t\mathbf{v}$, where $\mathbf{a}, \mathbf{v} \in V$, it can be shown that the levels of the signature have the following simple expression

$$S_k(\boldsymbol{x}) = \frac{\mathbf{v}^{\otimes k}}{k!}, \quad \forall k \geq 0.$$

Therefore, one can compute the signature of a sequence of length ℓ, viewed as a piecewise linear path, by iteratively applying the product $\otimes$ to the signature. This procedure is outlined in Alg. 1 where we use the notation $\exp_K(\cdot)$ for

$$\exp_K(\mathbf{v}) = \left(1, \mathbf{v}, \frac{1}{2}\mathbf{v}^{\otimes 2}, \dots, \frac{1}{K!}\mathbf{v}^{\otimes K}\right),$$

and $\otimes_K$ defined for any $\mathcal{A} = (A_0, A_1, \dots, A_K)$ and $\mathcal{B} = (B_0, B_1, \dots, B_K)$ by

$$\mathcal{A} \otimes_K \mathcal{B} = (C_0, C_1, \dots, C_K) \quad \text{with} \quad C_n = \sum_{k=0}^{n} A_k \otimes B_{n-k},$$

for all $0 \leq n \leq K$.

Algorithm 1 Sig $\mathcal{O}(\ell d^K)$

```
1: Input:  A stream $(\mathbf{x}_k)_{k=1}^{\ell}$ with $\mathbf{x}_k \in \mathbb{R}^d$ and truncation level $K \in \mathbb{N}_*$
2: Initialize $S \leftarrow \mathbf{1}$
3: for each $k \in \{1, 2, \dots, \ell - 1\}$ do
4:     $S \leftarrow S \otimes_K \exp_K(\mathbf{x}_{k+1} - \mathbf{x}_k)$
5: end for
6: $\Phi \leftarrow \text{vec}(S)$
7: Output: The truncated signature $\Phi$ at level $K$ of $(\mathbf{x}_k)_{k=1}^{\ell}$
```

The vector $\Phi = \text{vec}\left((1, S_1(\boldsymbol{x}), \dots, S_K(\boldsymbol{x}))\right)$ has $1+d+d^2+\ldots+d^K$ components. The number of features depends only on the dimension of the state space and not the length of the sequence. As we explained, the first step to fit a linear model on n signatures is to compute the latter. The time complexity of this preprocessing step is $\mathcal{O}(n\ell d^K)$ and the memory complexity $\mathcal{O}(nd^K)$. These complexities can be prohibitive for large d. However, this computationally expensive explicit vectorization of the data may be circumvented if the subsequent learning algorithm only requires the evaluation of inner products and if the latter can be computed efficiently. This is the key idea of kernel methods. For this reason, the signature kernel, introduced for the first time in Király and Oberhauser (2019), is a more recent but not less powerful tool for machine learning on sequential data as we shall see next.

2.2 PDEs, signature kernel, and kernel methods

In this section, we will start by defining kernel functions and subsequently introduce the signature kernel. Finally, we will see that a kernel function induces a particular Hilbert space of functions, and how the kernel trick allows one to develop efficient algorithms to solve optimization problems and conduct computations in these spaces of functions. Due to the flexibility of kernel methods, the ability to define a kernel on a space of paths allows us to seamlessly deploy these algorithms on sequential data.

We denote by $\mathcal{X}$ a general nonempty set and by $\mathcal{X}(V)$ the set of V-valued continuous paths of bounded variation. In the previous section, the letter y was used to denote a response variable. Here we often need to consider a pair of elements in the input space $\mathcal{X}$ which we will denote by (x, y).

Definition 6 (Positive definite kernel)**.** *A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel if for every $n \geq 1$ and $x_1, \ldots, x_n \in \mathcal{X}$ the $n \times n$ matrix $(k(x_i, x_j))_{1 \leq i,j \leq n}$ is positive semidefinite.*

We can prove that a given function k is a positive definite kernel using the following lemma.

Lemma 2.2.1. *$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if and only if there exists a Hilbert space H and a map $\Phi : \mathcal{X} \to H$ such that*

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle_H, \ \forall x, y \in \mathcal{X}.$$

We call Φ a feature map and H a feature space.

In light of this result, it is natural to consider the signature as a feature map and define a kernel on paths by the inner product of signatures, provided it has a meaning. So far we have seen that the signature maps a path $\boldsymbol{x} \in \mathcal{X}(V)$ into the algebra $(\mathscr{T}(V), \cdot, +, \otimes)$. We will now see that the range of the signature is also contained in a Hilbert space. To this aim, consider the subset $H(V)$ of $\mathscr{T}(V)$ defined by

$$H(V) = \Big\{ \mathcal{A} = (A_0, A_1, \ldots) \in \mathscr{T}(V) \ \Big| \ \sum_{k=0}^{\infty} ||A_k||^2_{V^{\otimes k}} < +\infty \Big\}. \tag{2.18}$$

The factorial decay eq. (2.10) ensures that if $\boldsymbol{x} \in \mathcal{X}(V)$, then $S(\boldsymbol{x}) \in H(V)$ as

$$\sum_{k=0}^{\infty} ||S_k(\boldsymbol{x})||^2_{V^{\otimes k}} \leq \sum_{k=0}^{\infty} \frac{\mathcal{V}(\boldsymbol{x})^{2k}}{(k!)^2} < +\infty.$$

As shown in (Cass et al., 2020) the space $H(V)$ is a Hilbert space equipped with the inner product

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{k=0}^{\infty} \langle A_k, B_k \rangle_{V^{\otimes k}}.$$

We can now instantiate (in the above lemma) the feature map Φ with the signature map S and the feature space H with $(H(V), \langle \cdot, \cdot \rangle)$.

2.2.1 The signature kernel

We are now in a position to define the signature kernel.

Definition 7 (Signature kernel)**.** *The signature kernel is the map defined by*

$$\begin{aligned} K : \mathcal{X}(V) \times \mathcal{X}(V) &\to \mathbb{R} \\ (\boldsymbol{x}, \boldsymbol{y}) &\mapsto \langle S(\boldsymbol{x}), S(\boldsymbol{y}) \rangle . \end{aligned}$$

As we have already noted, the ability to evaluate a kernel function without materializing the underlying features (which are not unique) is called a *kernel trick* and is key for developing efficient algorithms. A first kernel trick, based on the Horner's scheme, was provided in Király and Oberhauser (2019), making it possible to compute the inner product of truncated signatures for any truncation level, and to consider sequences valued in an infinite-dimensional state space. Subsequently, it has been shown in Cass et al. (2020) that the signature kernel can be approximated by solving a *Partial differential equation* (PDE).

Although we refer the reader to Cass et al. (2020) for the original proof of this result, we find it relevant to state the following theorem which establishes the fact that the signature solves a controlled differential equation, and which is the starting point of the proof.

Theorem 2.2.1. *Let $\boldsymbol{x} \in \mathcal{X}(V)$ and $\mathcal{A} = (A_0, A_1, ...) \in H(V)$. Consider the vector field $F : H(V) \to L(V, H(V))$ defined as follows*

$$F(\mathcal{A})(\mathbf{v}) = (0, A_0 \otimes \mathbf{v}, A_1 \otimes \mathbf{v}, \ldots) := \mathcal{A} \otimes \mathbf{v}.$$

Then, the unique solution to the following controlled differential equation,

$$d\boldsymbol{y} = F(\boldsymbol{y}) d\boldsymbol{x}, \quad \boldsymbol{y}(0) = \mathbf{1} = (1, 0, 0, \ldots),$$

is the path $\boldsymbol{y} : t \mapsto S(\boldsymbol{x}|_{[0,t]})$, where $\boldsymbol{x}|_{[0,t]}$ denotes the restriction of the path $\boldsymbol{x}$ to the interval $[0, t]$. In integral form we have

$$S(\boldsymbol{x}|_{[0,t]}) = \mathbf{1} + \int_0^t S(\boldsymbol{x}|_{[0,s]}) \otimes d\boldsymbol{x}(s). \tag{2.19}$$

We will refer to the map $S_{\text{path}} : t \mapsto S(\boldsymbol{x}|_{[0,t]})$ as the *pathwise signature*. Similarly, consider the function of two time variables,

$$\begin{aligned} u : [0, T] \times [0, T] &\to \mathbb{R} \\ (s, t) &\mapsto K(\boldsymbol{x}|_{[0,s]}, \boldsymbol{y}|_{[0,t]}). \end{aligned} \tag{2.20}$$

As shown in Cass et al. (2020), leveraging eq. (2.19) and the algebraic properties of the signature, one can formally write the integral equation

$$\big\langle S(\boldsymbol{x}|_{[0,s]}), S(\boldsymbol{y}|_{[0,t]})\big\rangle = 1 + \int_0^s \int_0^t \langle S(\boldsymbol{x}|_{[0,s']}), S(\boldsymbol{y}|_{[0,t']})\rangle \langle d\boldsymbol{x}(s'), d\boldsymbol{y}(t')\rangle.$$

Using eq. (2.20), we can replace the inner products of signatures to obtain

$$u(s,t) = 1 + \int_0^s \int_0^t u(s',t')\langle d\boldsymbol{x}(s'), d\boldsymbol{y}(t')\rangle.$$

If we further assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are continuously differentiable, that is $\boldsymbol{x}, \boldsymbol{y} \in \mathscr{C}^1([0,T], V)$, this equation can be simplified and the fundamental theorem of calculus applies, yielding the following result.

Theorem 2.2.2. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathscr{C}^1([0,T], V)$ and $u : (s,t) \mapsto K(\boldsymbol{x}|_{[0,s]}, \boldsymbol{y}|_{[0,t]})$. The two parameter function u solves the partial differential equation*

$$\frac{\partial^2 u}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t)\rangle\, u, \qquad u(0,\cdot) = 1 \text{ and } u(\cdot, 0) = 1, \tag{2.21}$$

and where $\dot{\boldsymbol{z}}(s) = \frac{d\boldsymbol{z}(r)}{dr}\big|_{r=s}$.

Evaluating the signature kernel

In light of this result, the signature kernel $K(\cdot,\cdot)$ can be evaluated at two input paths $\boldsymbol{x}, \boldsymbol{y}$, by solving the PDE in eq. (2.21) and evaluating the solution at $(s,t) = (T,T)$. During the development of this thesis, we contributed to the construction and implementation of a finite difference scheme for approximating the signature kernel in the sigkernel Python package which is publicly available at `https://github.com/crispitagorico/sigkernel`. The principle of the finite difference method is to first discretize the domain $[0,T] \times [0,T]$ into a two-dimensional grid $\mathcal{D} = \{(s_i, t_i)\}$, and then approximate the value of the solution u at the points in $\mathcal{D}$ using an update rule. More precisely, on each cell $\mathcal{R}_{i,j} = \{(s,t) \in [0,T] \times [0,T] \mid s_i \le s \le s_{i+1},\ t_j \le t \le t_{j+1}\}$ the inner product in eq. (2.21) is approximated by

$$\langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t)\rangle \approx \left\langle \frac{\boldsymbol{x}(s_{i+1}) - \boldsymbol{x}(s_i)}{s_{i+1} - s_i}, \frac{\boldsymbol{y}(t_{j+1}) - \boldsymbol{y}(t_j)}{t_{j+1} - t_j} \right\rangle := \Delta_{\boldsymbol{xy}}(i,j), \tag{2.22}$$

for all $(s,t) \in \mathcal{R}_{i,j}$. Then, the value of solution at the upper-right corner of $\mathcal{R}_{i,j}$ is approximated using the solution at the other three corners of $\mathcal{R}_{i,j}$

$$u(s_{i+1}, t_{j+1}) \approx f_{\text{upd}}\left(u(s_i, t_{j+1}), u(s_{i+1}, t_j), u(s_i, t_j), \Delta_{\boldsymbol{xy}}(i,j)\right),$$

where f_{upd} denotes an update rule. We refer the reader to Cass et al. (2020) for specific instantiations of f_{upd}.

In Alg. 2 we outline the resulting numerical scheme for evaluating the signature kernel at two piecewise linear paths $\boldsymbol{x}$ and $\boldsymbol{y}$ resulting for example from a piecewise linear embedding of two time-series. We note that this numerical scheme provides an approximation to $K(\boldsymbol{x}, \boldsymbol{y})$ with error estimates given in Cass et al. (2020). Although there is a natural grid $\mathcal{D}$ associated with piecewise linear paths, such that $\langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle$ is constant on each cell $\mathcal{R}_{i,j}$, in Alg. 2 we use the fact that this grid may be refined (for example using a dyadic refinement) in order to increase the accuracy of the solver. Another situation in which kernels need to be approximated can be found in the context of *latent force models* (LFMs) where *Gaussian process*s (GPs) are combined with differential equations (Alvarez et al., 2009).

Algorithm 2 PDESolve $\mathcal{O}(2^{2\kappa} d \ell_{\mathbf{x}} \ell_{\mathbf{y}})$

```
1: Input:  Two streams $(\mathbf{x}_k)_{k=1}^{\ell_\mathbf{x}}$, $(\mathbf{y}_k)_{k=1}^{\ell_\mathbf{y}}$ of dimension $d$ and level $\kappa$ of the
   dyadic refinement (step size $= 2^{-\kappa}$)
2: Create array $U$ to store the solution of the PDE
3: Initialize $U[i,:] \leftarrow 1$ for $i \in \{1, 2, \dots, 2^\kappa * (\ell_\mathbf{x} - 1) + 1\}$
4: Initialize $U[:,j] \leftarrow 1$ for $j \in \{1, 2, \dots, 2^\kappa * (\ell_\mathbf{y} - 1) + 1\}$
5: for each $i \in \{1, 2, \dots, 2^\kappa * (\ell_\mathbf{x} - 1)\}$ do
6:   for each $j \in \{1, 2, \dots, 2^\kappa * (\ell_\mathbf{y} - 1)\}$ do
7:     $\Delta_\mathbf{x} = (\mathbf{x}_{\lceil i/(2^\kappa) \rceil + 1} - \mathbf{x}_{\lceil i/(2^\kappa) \rceil})/2^\kappa$
8:     $\Delta_\mathbf{y} = (\mathbf{y}_{\lceil j/(2^\kappa) \rceil + 1} - \mathbf{y}_{\lceil j/(2^\kappa) \rceil})/2^\kappa$
9:     $\Delta_{\mathbf{xy}} = \langle \Delta_\mathbf{x}, \Delta_\mathbf{y} \rangle$
10:    // Update the value of the solution using the solution at nearby points
11:    $U[i+1, j+1] = U[i, j+1] + U[i+1, j] + (\Delta_{\mathbf{xy}} - 1.) * U[i, j]$
12:  end for
13: end for
14: Output: The solution of the PDE at the final times $U[-1, -1]$
```

Sequentializing static kernels

Until now we have assumed that the state space of the path had a Hilbert space structure. Besides, we could only compute the signature of paths valued in a finite-dimensional space. Working with the signature kernel offers more flexibility. Indeed, if we have a map φ mapping a general non-empty set $\mathcal{Z}$ onto a Hilbert space V, we can define a new kernel K_φ on $\mathcal{X}(\mathcal{Z})$ as follows (as per Steinwart and Christmann (2008, Lemma 4.3))

$$K_\varphi(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x}_\varphi, \boldsymbol{y}_\varphi),$$

where $\boldsymbol{x}_\varphi : t \mapsto \varphi(\boldsymbol{x}(t))$. In particular, if k is a kernel function on $\mathcal{Z}$ and $\varphi(\boldsymbol{x}(t)) = k(\cdot, \boldsymbol{x}(t))$, provided φ is differentiable, we obtain the PDE

$$\frac{\partial^2 u}{\partial s \partial t} = \frac{\partial^2 k(\boldsymbol{x}(s), \boldsymbol{y}(t))}{\partial s \partial t} u,$$

where we can use a finite difference approximation as in eq. (2.22), that is

$$\begin{aligned} \frac{\partial^2 k(\boldsymbol{x}(s), \boldsymbol{y}(t))}{\partial s \partial t} \approx \frac{1}{\delta^2} \big(& k(\boldsymbol{x}(s+\delta), \boldsymbol{y}(t+\delta)) - k(\boldsymbol{x}(s), \boldsymbol{y}(t+\delta)) \\ & - k(\boldsymbol{x}(s+\delta), \boldsymbol{y}(t)) + k(\boldsymbol{x}(s), \boldsymbol{y}(t)) \big). \end{aligned}$$

One may still use the numerical scheme in Alg. 2 changing the lines $7:9$ to compute the above approximation with kernel k instead of the Euclidean inner product in eq. (2.22). The consequence is that if one has a kernel function on $\mathcal{Z}$, the signature kernel provides a way to sequentialize it for learning on time-evolving structured objects $\boldsymbol{x} : [0, T] \to \mathcal{Z}$, as explained in Király and Oberhauser (2019).

2.2.2 Kernel methods

So far we have defined positive definite kernels, and seen the example of the signature kernel which realizes the inner product of signature features. Although a kernel does not uniquely define a feature map and a feature space, we will see that there is a canonical feature space associated to a kernel, called a *Reproducing kernel Hilbert space* (RKHS). Within this RKHS framework, we will describe learning techniques which only require the ability to define a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is a non-empty set, making it possible to work on different input spaces $\mathcal{X}$ in a unified manner. The ability to evaluate the kernel via a kernel trick is key to obtain efficient algorithms. In everything that follows, one can in principle instantiate the input space with $\mathcal{X}(V)$ and the kernel with the signature kernel.

To this aim we introduce the notion of *reproducing kernels* which is ultimately linked to the notion of positive definite kernels that we have been using so far.

Definition 8 (Reproducing kernel)**.** *Let $\mathcal{H}$ be Hilbert space of functions from $\mathcal{X}$ to $\mathbb{R}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a reproducing kernel of $\mathcal{H}$ if*

1. *$k(\cdot, x) \in \mathcal{H}$ for all $x \in \mathcal{X}$,*
2. *$f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$ and all $x \in \mathcal{X}$ (reproducing property).*

In particular, points 1. and 2. imply that $\langle k(\cdot, y), k(\cdot, x)\rangle_{\mathcal{H}} = k(x, y)$, when k is a reproducing kernel. Therefore, using lemma 2.2.1, k is a positive definite kernel with feature map

$$\Phi : \mathcal{X} \to \mathcal{H},\ x \mapsto k(\cdot, x).$$

A Hilbert space of functions which possesses a reproducing kernel is called a *Reproducing kernel Hilbert space* (RKHS). The following theorem (proved in Steinwart and Christmann (2008, Thm. 4.21)) gives a construction for the unique RKHS associated to a kernel.

Theorem 2.2.3. *Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel with feature space H and feature map $\Phi : \mathcal{X} \to H$. Then the RKHS for which k is a reproducing kernel is unique, and is given by*

$$\mathcal{H}_k = \left\{ f : \mathcal{X} \to \mathbb{R} \mid \exists \boldsymbol{w} \in H \text{ with } f(\cdot) = \langle \boldsymbol{w}, \Phi(\cdot)\rangle_H \right\},$$

equipped with the norm

$$\|f\|_{\mathcal{H}_k} = \inf \left\{ \|\boldsymbol{w}\|_H \mid \boldsymbol{w} \in H \text{ and } f(\cdot) = \langle \boldsymbol{w}, \Phi(\cdot)\rangle_H \right\}. \tag{2.23}$$

An RKHS framework for nonparameteric regression

A representer theorem reduces infinite-dimensional optimization problems to tractable finite-dimensional ones. In the context of RKHS, the following representer theorem is due to Schölkopf et al. (2001).

Theorem 2.2.4. *Any solution to the optimization problem*

$$\operatorname*{arg\,min}_{f \in \mathcal{H}_k} c\left((x_1, y_1, f(x_1)), \dots, (x_n, y_n, f(x_n))\right) + \Omega\left(\|f\|_{\mathcal{H}_k}\right), \tag{2.24}$$

where $c : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{+\infty\}$ is a cost function, and Ω is a strictly increasing real-valued function on $[0, +\infty[$, admits a representation of the form

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i).$$

Regularization is commonly used to overcome the problem of learning in a high or infinite-dimensional space. *Kernel Ridge regression* (KRR) is obtained by regularizing the mean squared error (MSE) by the squared RKHS norm

$$\operatorname*{arg\,min}_{f \in \mathcal{H}_k} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \|f\|^2_{\mathcal{H}_k}. \tag{2.25}$$

According to the representer theorem, the solution of the KRR problem can be expressed as a finite linear combination of the representers $k(\cdot, x_i)$ of the data points x_i. Furthermore, the coefficients $\boldsymbol{\alpha} = (\alpha_i)_{i=1}^n$ are given by

$$\boldsymbol{\alpha} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{y}, \qquad [\mathbf{K}]_{i,j} = k(x_i, x_j), \tag{2.26}$$

where $\mathbf{y} = (y_i)_{i=1}^n$ and $1 \leq i, j \leq n$.

Remark. *Suppose $\mathcal{H}_k$ is finite-dimensional such that $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ where $\Phi(x_i) \in \mathbb{R}^d$ is a feature vector. Then the optimization problem in eq.* (2.25) *is equivalent to*

$$\arg\min_{\mathbf{w}} \sum_{i=1}^n (y_i - \mathbf{w}'\Phi(x_i))^2 + \lambda \|\mathbf{w}\|^2,$$

with solution given by

$$\mathbf{w} = (\boldsymbol{\Phi}'\boldsymbol{\Phi} + \lambda \mathbf{I}_d)^{-1}\boldsymbol{\Phi}'\mathbf{y}, \qquad [\boldsymbol{\Phi}]_{i,:} = \Phi(x_i).$$

The prediction function is $f^*(x) = \mathbf{w}'\Phi(x)$. Computing the (primal) variables $\mathbf{w}$ takes $\mathcal{O}(d^3)$, whilst computing the (dual) variables $\boldsymbol{\alpha}$ takes $\mathcal{O}(n^3)$.

Support vector machines (SVM), another popular kernel method, can also be written in the form of eq. (2.24) so that the representer theorem applies.

Whilst the regularization prevents overfitting, one should also avoid underfitting. Universal kernels ensure that the associated RKHS is large enough to provide good approximations of the target function.

Definition 9 (Universal kernel)**.** *Let U be a compact metric space. A continuous kernel $k : U \times U \to \mathbb{R}$ is called universal if the RKHS $\mathcal{H}_k$ is dense in $\mathscr{C}(U, \mathbb{R})$, i.e., for any function $f \in \mathscr{C}(U, \mathbb{R})$ and any $\epsilon > 0$ there exists $g \in \mathcal{H}_k$ such that*

$$\|f - g\|_\infty \leq \epsilon.$$

The signature kernel K is universal. Indeed, since the signature has the universal approximation property on U as defined in Thm. 2.1.4, for any function $f \in \mathscr{C}(U, \mathbb{R})$ and any $\epsilon > 0$ there exists an $\mathcal{A} \in H(\mathbb{R} \oplus V)$ such that $\|f - \langle \mathcal{A}, S(\cdot) \rangle\|_\infty \leq \epsilon$ with $\langle \mathcal{A}, S(\cdot) \rangle \in \mathcal{H}_K$.

Gaussian process models

Gaussian process models provide another widely used approach for learning functions based on positive definite kernels. These are Bayesian nonparametric models in which a prior is placed over an unknown function of interest, by means of Gaussian processes, an important class of stochastic processes.

Definition 10 (real-valued Gaussian process)**.** *A Gaussian process (GP) f with index set $\mathcal{X}$ is a collection of random variables $\{f_x\}_{x\in\mathcal{X}}$ such that for any $n \in \mathbb{N}$ and any set of points $x_1, \dots, x_n \in \mathcal{X}$, $(f_{x_1}, \dots, f_{x_n})$ is multivariate Gaussian.*

A GP is fully determined by the mean function $m : \mathcal{X} \to \mathbb{R}$ and the covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which are defined for all $x, x' \in \mathcal{X}$ by,

$$m(x) = \mathbb{E}[\, f_x \,] , \qquad k(x, x') = \mathrm{Cov}(f_x, f_{x'}).$$

We write $f \sim \mathcal{GP}(m, k)$ and have the following finite-dimensional distributions

$$(f_{x_1}, \dots, f_{x_n}) \sim \mathcal{N}(\mathbf{m}, \mathbf{K}),$$

with mean vector $[\mathbf{m}]_i = m(x_i)$ and covariance matrix $[\mathbf{K}]_{i,j} = k(x_i, x_j)$ for $1 \leq i, j \leq n$. A mean function $m : \mathcal{X} \to \mathbb{R}$ and a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ define a GP prior $f \sim \mathcal{GP}(m, k)$ over an unknown function $f : \mathcal{X} \to \mathbb{R}$. In this context, it is common to write $f(x)$ instead of f_x.

Gaussian process regression Now we briefly expose how a GP model can be defined to reason probabilistically about a regression function given a dataset $\{(x_i, y_i)\}_{i=1}^n$ of input-output pairs $(x_i, y_i) \in \mathcal{X} \times \mathbb{R}$. For ease of exposition we consider a zero-mean prior, that is $m \equiv 0$. A GP model is completed by specifying a conditional likelihood

$$f \sim \mathcal{GP}(0, k), \qquad p(y_i \,|\, f(x_i)) = \mathcal{N}(y; f(x_i), \sigma^2).$$

In Bayesian statistics, inference about unknown quantities consists in calculating the posterior. Here, the combination of the GP prior and the data leads to closed-form posterior distributions. Let $\mathbf{y} = (y_1, \dots, y_n)'$ and $\mathbf{f}_* = (f(x_1^*), \dots, f(x_p^*))'$ where $x_1^*, \dots, x_p^* \in \mathcal{X}$. Considering the joint distribution $(\mathbf{y}, \mathbf{f}_*)$ and using the Gaussian conditioning properties, one straightforwardly obtains the posterior

$$\mathbf{f}_* \,|\, \mathbf{y} \sim \mathcal{N}(\mathbf{m}_*, \mathbf{K}_*),$$

with mean vector and covariance matrix defined component-wise for $1 \leq i, j \leq p$

$$\begin{aligned} [\mathbf{m}_*]_i &= \mathbf{k}(x_i^*)'(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{y} \\ [\mathbf{K}_*]_{i,j} &= k(x_i^*, x_j^*) - \mathbf{k}(x_i^*)'(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1} \mathbf{k}(x_j^*), \end{aligned}$$

where $\mathbf{k}(x^*) \in \mathbb{R}^n$ with $[\mathbf{k}(x^*)]_i = k(x_i, x^*)$. Therefore, the posterior is a GP with mean function m_* and covariance function k_* defined for any $x, x' \in \mathcal{X}$ by

$$\begin{aligned} m_*(x) &= \mathbf{k}(x)'(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1}\mathbf{y} \\ k_*(x, x') &= k(x, x') - \mathbf{k}(x)'(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1}\mathbf{k}(x'). \end{aligned}$$

One can note that the kernel ridge regressor in eq. (2.26) is identical to the posterior mean of the Gaussian process. However, the GP approach provides uncertainty estimates through $k_*(x, x)$. Besides, the Bayesian inference principles provide ways to adjust the various hyperparameters of the model including the parameters of the kernel function and the variance σ^2. We denote these parameters (previously omitted from the notations) collectively by $\boldsymbol{\theta}$. It can be easily shown that the log-probability $\log p(\mathbf{y}; \boldsymbol{\theta})$, called the log-marginal likelihood, is given by

$$\log p(\mathbf{y}; \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}'(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1}\mathbf{y} - \frac{1}{2}\log\left|\mathbf{K} + \sigma^2 \mathbf{I}_n\right| - \frac{n}{2}\log 2\pi.$$

It is standard practice to maximize the log-marginal likelihood w.r.t. the hyperparameters, which is known as *Type II maximum likelihood* (ML-II). In addition, there exist libraries dedicated for GP models using gradient-based optimizers from TensorFlow or PyTorch (e.g. GPflow (De G. Matthews et al., 2017), GPyTorch (Gardner et al., 2018)) that leverage automatic differentiation. However, optimizing the log-marginal likelihood is computationally expensive, as one needs to repeatedly compute the inverse of the $n \times n$ matrix $\mathbf{K} + \sigma^2 \mathbf{I}_n$ which has a time complexity $\mathcal{O}(n^3)$. Once this inverse has been computed, the posterior mean and covariance can be evaluated in $\mathcal{O}(pn^2)$. The cubic complexity becomes prohibitive for large training datasets, which motivates approximate methods, that we will describe next.

Non-Gaussian likelihood A GP model can also be specified to learn a classifier. The conditional likelihood can be defined through a *link function* $\sigma : \mathbb{R} \to [0, 1]$ such as a probit or logit link function

$$f \sim \mathcal{GP}(0, k), \qquad p(y_i = 1 \,|\, f(x_i)) = \sigma(f(x_i)).$$

However, in this case, $(\mathbf{y}, \mathbf{f}_*)$ is no longer jointly Gaussian and the conditional distribution $\mathbf{f}_* \,|\, \mathbf{y}$ is no longer available in closed form. For this reason, different approximation techniques have been proposed (Rasmussen and Williams, 2006, Chap. 3). In the rest of this part we will give a brief account on so-called sparse variational GPs, which address this problem as well as the aforementioned large-scale regression challenge.

Variational inference Sparse variational GPs are based on the principle of *Variational inference* (VI) which turns the problem of approximating a posterior distribution $p(\boldsymbol{\alpha} \,|\, \mathbf{y})$ into an optimization problem (Blei et al., 2017). The idea behind VI is to first posit a family of densities $\mathcal{Q}$ and then to find a member $q^*(\boldsymbol{\alpha}) \in \mathcal{Q}$ that minimizes the *Kullback-Leibler divergence* (KL) to the target density $p(\boldsymbol{\alpha} \,|\, \mathbf{y})$

$$q^*(\boldsymbol{\alpha}) = \operatorname*{arg\,min}_{q(\boldsymbol{\alpha}) \in \mathcal{Q}} \mathrm{KL}\left[q(\boldsymbol{\alpha}) \,\|\, p(\boldsymbol{\alpha} \,|\, \mathbf{y})\right]. \tag{2.27}$$

Although the KL can not be computed, the optimization problem in eq. (2.27) is equivalent to the following one

$$q^*(\boldsymbol{\alpha}) = \operatorname*{arg\,max}_{q(\boldsymbol{\alpha}) \in \mathcal{Q}} \left\{\mathbb{E}_{q(\boldsymbol{\alpha})}[\log p(\mathbf{y} \,|\, \boldsymbol{\alpha})] - \mathrm{KL}\left[q(\boldsymbol{\alpha}) \,\|\, p(\boldsymbol{\alpha})\right]\right\}. \tag{2.28}$$

This new objective is termed the *Evidence lower bound* (ELBO) and is a lower bound on the marginal log likelihood $p(\mathbf{y})$

$$\log p(\mathbf{y}) \geq \mathbb{E}_{q(\boldsymbol{\alpha})}[\log p(\mathbf{y} \,|\, \boldsymbol{\alpha})] - \mathrm{KL}\left[q(\boldsymbol{\alpha}) \,\|\, p(\boldsymbol{\alpha})\right].$$

Key to the success of VI, is to choose the family of approximate posteriors $\mathcal{Q}$ to be simple enough to efficiently solve the optimization problem in eq. (2.28), but flexible enough to yield a good approximation to $p(\boldsymbol{\alpha} \,|\, \mathbf{y})$.

Variational inference for GPs In the context of GPs, the goal is to approximate $f \,|\, \mathbf{y}$. In sparse variational GPs a family of approximate posterior GPs is constructed as follows. Let $\mathbf{z} = \{z_1, \dots, z_M\}$ be a collection of M points in $\mathcal{X}$—the index set of the GP—and denote by $\mathbf{u} = (f(z_1), \dots, f(z_M))'$, the finite-dimensional projection of the GP on these points. Under the GP prior we have that $p(\mathbf{u}, \mathbf{f}) = p(\mathbf{f} \,|\, \mathbf{u})p(\mathbf{u})$ for any $\mathbf{f} = (f(x_1), \dots, f(x_n))'$

$$\begin{aligned} p(\mathbf{u}) &= \mathcal{N}(\mathbf{0}_M, \mathbf{K}_{\mathbf{zz}}) \\ p(\mathbf{f} \,|\, \mathbf{u}) &= \mathcal{N}\left(\mathbf{K}_{\mathbf{zx}}' \mathbf{K}_{\mathbf{zz}}^{-1} \mathbf{u}, \mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{zx}}' \mathbf{K}_{\mathbf{zz}}^{-1} \mathbf{K}_{\mathbf{zx}}\right), \end{aligned}$$

where $\mathbf{K}_{\mathbf{zz}} \in \mathbb{R}^{M \times M}$ and $\mathbf{K}_{\mathbf{zx}} \in \mathbb{R}^{M \times n}$ denote the covariance matrices

$$[\mathbf{K}_{\mathbf{zz}}]_{m,m'} = \mathrm{Cov}(f(z_m), f(z_{m'})) \qquad [\mathbf{K}_{\mathbf{zx}}]_{m,i} = \mathrm{Cov}(f(z_m), f(x_i)),$$

The variational family is then defined as a family of GPs with finite-dimensional marginal densities of the form $q(\mathbf{u}, \mathbf{f}) = q(\mathbf{f}|\mathbf{u})q(\mathbf{u})$ with

$$\begin{aligned} q(\mathbf{u}) &= \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma}) \\ q(\mathbf{f} \,|\, \mathbf{u}) &= \mathcal{N}\left(\mathbf{K}_{\mathbf{zx}}' \mathbf{K}_{\mathbf{zz}}^{-1} \mathbf{u}, \mathbf{K}_{\mathbf{xx}} - \mathbf{K}_{\mathbf{zx}}' \mathbf{K}_{\mathbf{zz}}^{-1} \mathbf{K}_{\mathbf{zx}}\right). \end{aligned}$$

The family of approximate posterior GPs is parametrized by a mean vector $\mathbf{m}$ and a covariance matrix $\mathbf{\Sigma}$. Marginalizing $q(\mathbf{f}\,|\,\mathbf{u})q(\mathbf{u})$ over $\mathbf{u}$, we can show that the mean function μ and covariance function ν are defined for all $x, x' \in \mathcal{X}$ by

$$\begin{aligned} \mu(x) &= \mathbf{k}_{\mathbf{z}}(x)'\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m} \\ \nu(x,x') &= k(x,x') - \mathbf{k}_{\mathbf{z}}(x)'\mathbf{K}_{\mathbf{zz}}^{-1}(\mathbf{K}_{\mathbf{zz}} - \mathbf{\Sigma})\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{k}_{\mathbf{z}}(x'), \end{aligned}$$

where $\mathbf{k}_{\mathbf{z}}(x) \in \mathbb{R}^M$ with $[\mathbf{k}_{\mathbf{z}}(x)]_m = k(z_m, x)$. After choosing the family of approximate posteriors, following the VI framework, one minimizes a KL objective which is equivalent to maximizing an ELBO. In the context of GPs defining the KL objective is technically more involved, due to the fact that inference is conducted on a stochastic process f and not on a finite-dimensional random variable $\boldsymbol{\alpha}$. However, all objectives considered in the literature (Matthews, 2017; Titsias, 2009; Wild and Wynne, 2021) yield the following lower bound on the log-marginal likelihood (ELBO)

$$\log p(\mathbf{y};\boldsymbol{\theta}) \geq \mathbb{E}_{q(\mathbf{u})q(\mathbf{f}\,|\,\mathbf{u})}[\log p(\mathbf{y}\,|\,\mathbf{f})] - \mathrm{KL}\left[q(\mathbf{u}) \,\|\, p(\mathbf{u})\right]. \tag{2.29}$$

By the definition of the *Kullback-Leibler divergence*,

$$\mathrm{KL}\left[q(\mathbf{u}) \,\|\, p(\mathbf{u})\right] = \mathbb{E}_{q(\mathbf{u})}\left[\log \frac{q(\mathbf{u})}{p(\mathbf{u})}\right] := \mathrm{KL},$$

which is available in closed form due to choosing $q(\mathbf{u})$ multivariate Gaussian

$$\mathrm{KL} = \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{\Sigma}\right) + \mathbf{m}'\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{m} - M + \log\left(\frac{\det \mathbf{K}_{\mathbf{zz}}}{\det \mathbf{\Sigma}}\right)\right), \tag{2.30}$$

and can be computed in $\mathcal{O}(M^3)$. Concerning the first term in the ELBO

$$\mathbb{E}_{q(\mathbf{u})q(\mathbf{f}\,|\,\mathbf{u})}[\log p(\mathbf{y}\,|\,\mathbf{f})] = \sum_{i=1}^{n} \mathbb{E}_{q(f(x_i))}[\log p(y_i\,|\,f(x_i))], \tag{2.31}$$

the expectations can be estimated with a Gauss-Hermite quadrature (in the case where $p(y_i\,|\,f(x_i))$ is non-Gaussian) once the moments of $q(\mathbf{f})$ have been computed. The overall computational complexity is $\mathcal{O}(M^3 + nM^2)$. Note that this variational inference framework also provides the ability to carry out ML-II to select the hyperparameters of the model through maximization of the ELBO. Here, these hyperparameters include the kernel parameters, but also the variational parameters $\mathbf{m}$ and $\mathbf{\Sigma}$, as well as the inducing points z_m.

Remark. *We have not made any assumptions on the index set $\mathcal{X}$ of the GP. Therefore, in principle, the framework of sparse variational GPs applies to spaces of paths. However, choosing (or optimizing) the inducing variables z_m in $\mathcal{X}(V)$ is not trivial. Sparse variational GPs for GPs indexed on $\mathcal{X}(V)$ have* *been developed in Toth and Oberhauser (2020).*

As justified in Matthews (2017), the inducing variables which have been defined as $\mathbf{u} = (f(z_1), \dots, f(z_M))'$ may be replaced by another collection of M real-valued random variables provided they are deterministic conditioned on the Gaussian process f. The ELBO can still be expressed as eq. (2.29), however we may choose different notations for the covariance matrices (which are no longer the classical kernel matrices) as follows:

$$\begin{aligned}
[\mathbf{K_{zz}}]_{m,m'} &= \mathrm{Cov}(f(z_m), f(z_{m'})) &&\to & [\mathbf{C_{uu}}]_{m,m'} &= \mathrm{Cov}(u_m, u_{m'}) \\
[\mathbf{K_{zx}}]_{m,i} &= \mathrm{Cov}(f(z_m), f(x_i)) &&\to & [\mathbf{C_{uf}}]_{m,i} &= \mathrm{Cov}(u_m, f(x_i)) \\
[\mathbf{k_z}(x)]_{m} &= \mathrm{Cov}(f(z_m), f(x)) &&\to & [\mathbf{c_u}(x)]_{m} &= \mathrm{Cov}(u_m, f(x)).
\end{aligned}$$

Representing probability measures into a RKHS

An important type of non-vectorial inputs, that can be handled with kernel methods, are probability measures. Under suitable assumptions on the kernel function k, probability distributions can be embedded into $\mathcal{H}_k$ by their *Kernel mean embedding* (KME) defined hereafter.

Definition 11 (Kernel mean embedding (KME))**.** *Given $\mathbb{P}$ a Borel probability measure on $\mathcal{X}$ and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a kernel such that $\mathbb{E}_{X\sim\mathbb{P}}[k(X,X)] < +\infty$, then we can define the Bochner integral*

$$\mu_{\mathbb{P}} = \int_{\mathcal{X}} k(\cdot, x) d\mathbb{P}(x).$$

We call $\mu_{\mathbb{P}} \in \mathcal{H}_k$ the kernel mean embedding (KME) of $\mathbb{P}$.

Analogously to the reproducing property $\mathbb{E}_{X\sim\mathbb{P}}[f(X)] = \langle f, \mu_{\mathbb{P}} \rangle_{\mathcal{H}_k}$. The combination of the KME and the kernel trick allows one to evaluate an integral probability metric known as the *Maximum mean discrepancy* (MMD), defined as follows,

$$\mathscr{D}_k(\mathbb{P}, \mathbb{Q}) = \sup_{\|f\|_{\mathcal{H}_k} \leq 1} \left(\int_{\mathcal{X}} f(x) d\mathbb{P}(x) - \int_{\mathcal{X}} f(x) d\mathbb{Q}(x) \right).$$

Lemma 2.2.2. *If $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfies $\mathbb{E}_{X\sim\mathbb{P}}[k(X,X)] < +\infty$, then*

$$\mathscr{D}_k(\mathbb{P}, \mathbb{Q}) = \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}_k}.$$

Unlike other integral probability metrics, the MMD provides the flexibility of measuring the distance between probability measures defined on structured domains $\mathcal{X}$, provided that one has a suitable kernel function on $\mathcal{X} \times \mathcal{X}$. Using the above lemma the MMD can be written in terms of the kernel function

$$(\mathscr{D}_k(\mathbb{P},\mathbb{Q}))^2 = \mathbb{E}_{X,X'}\big[k(X,X')\big] - 2\mathbb{E}_{X,Y}\big[k(X,Y)\big] + \mathbb{E}_{Y,Y'}\big[k(Y,Y')\big]\,,$$

where $X, X' \overset{\text{i.i.d}}{\sim} \mathbb{P}$ and $Y, Y' \overset{\text{i.i.d}}{\sim} \mathbb{Q}$. Given two samples $X_1, \dots, X_m \overset{\text{i.i.d}}{\sim} \mathbb{P}$ and $Y_1, \dots, Y_n \overset{\text{i.i.d}}{\sim} \mathbb{Q}$, an unbiased estimator is given by

$$\begin{aligned}\left(\hat{\mathscr{D}}_k(\mathbb{P},\mathbb{Q})\right)^2 =& \frac{1}{m(m-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{m} k(X_i,X_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(X_i,Y_j)\\ &+ \frac{1}{n(n-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{n} k(Y_i,Y_j),\end{aligned}$$

which is fully expressed in terms of evaluations of kernel. Other estimators are discussed in Gretton et al. (2012a).

In general the MMD is not a metric. However, it is a metric when the kernel k is *characteristic*, i.e. is a kernel for which the map $\mathbb{P} \mapsto \mu_{\mathbb{P}}$ is injective. Gretton et al. (2012a) establishes a condition on the RKHS $\mathcal{H}_k$, such that the kernel is characteristic and the MMD is a metric.

Theorem 2.2.5. *If the kernel k is universal on a compact metric space $\mathcal{X}$, then $\mathscr{D}_k(\mathbb{P},\mathbb{Q}) = 0$ if and only if $\mathbb{P} = \mathbb{Q}$.*

This result allows to use the MMD as a test statistic for statistical hypothesis tests to determine if two samples are drawn from different distributions.

2.3 Neural controlled differential equations

In Sec. 2.1 we explained how the signature map provides canonical basis function expansions for real-valued functions on paths $f : \mathcal{X}(\mathbb{R}^d) \to \mathbb{R}$. We have seen that the signature features naturally appear when studying the solutions of linear CDEs. In Sec. 2.2 we stated the fact that the signature itself solves a particular CDE, a result that is exploited to derive a kernel trick that allows for efficient kernel methods on sequential data. In this section, we will give a brief account on NCDEs, a class of neural networks introduced in Kidger et al. (2020), which also leverages CDEs for approximating functions $f : \mathcal{X}(\mathbb{R}^d) \to \mathbb{R}^e$, yet in a different way.

Recall from Sec. 2.1 that the solution map of a CDE of the form of eq. (2.6)

$$d\boldsymbol{y} = F(\boldsymbol{y})d\boldsymbol{x}, \quad \boldsymbol{y}(0) \in \mathbb{R}^e,$$

with Lipschitz continuous vector field $F : \mathbb{R}^e \to \mathbb{R}^{e\times d}$, gives rise to a continuous function $f : \mathcal{X}(\mathbb{R}^d) \to \mathbb{R}^e$ such that for all $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^d)$

$$f(\boldsymbol{x}) = \boldsymbol{y}(T) = \boldsymbol{y}(0) + \int_0^T F(\boldsymbol{y}(t))d\boldsymbol{x}(t), \tag{2.32}$$

where the integral on the right hand side is interpreted in the Riemann-Stieltjes sense. The key idea of NCDEs is to parametrize the vector field by a neural network $F_\theta : \mathbb{R}^e \to \mathbb{R}^{e\times d}$ and learn the parameters θ from data by optimizing a given loss function.

2.3.1 The NCDE model

More precisely, the NCDE model uses eq. (2.32) to model the evolution of a latent (or hidden) state $\boldsymbol{y}(t) \in \mathbb{R}^h$. It also adds architectural features to model the initial latent state $\boldsymbol{y}(0)$ and read information from the hidden state at final time $\boldsymbol{y}(T)$. With this, the NCDE model reads as

$$\boldsymbol{y}(0) = \ell_\theta(\boldsymbol{x}(0)), \quad \boldsymbol{y}(T) = \boldsymbol{y}(0) + \int_0^T F_\theta(\boldsymbol{y}(t))d\boldsymbol{x}(t), \quad f(\boldsymbol{x}) = \pi_\theta(\boldsymbol{y}(T)),$$

where $\ell_\theta : \mathbb{R}^d \to \mathbb{R}^h$ and $\pi_\theta : \mathbb{R}^h \to \mathbb{R}^e$ are feedforward neural networks. The output is then fed to a loss function (mean squared, cross entropy, etc.) and trained via stochastic gradient descent.

In the original version of NCDEs (Kidger et al., 2020), $\boldsymbol{x} : [0,T] \to \mathbb{R}^d$ is assumed differentiable and obtained via natural cubic splines interpolation of a time-series; therefore the term "$d\boldsymbol{x}(t)$" can be interpreted as "$\dot{\boldsymbol{x}}(t)dt$" and the integral can be evaluated numerically via a call to an ODE solver of choice (e.g. Euler, Runge-Kutta, adaptive schemes). For this reason, the implementation of NCDEs relies on the software developed in the context of *neural ordinary differential equations* (NODE) models (Chen et al., 2018).

The aforementioned NODE is a deep learning architecture which popularized the field of *neural differential equations*, hybrid approaches combining neural networks and differential equations. NODE exploited the fact that certain models may be interpreted as approximations to differential equations. In particular, an L-layers *residual network* (ResNet) consists in applying the following sequence of transformations

$$\mathbf{y}_k = \mathbf{y}_{k-1} + F_\theta(\mathbf{y}_{k-1}), \tag{2.33}$$

to a hidden state $\mathbf{y}_k \in \mathbb{R}^h$, for $k = 1, \dots, L$. These iterative updates can be seen as a forward Euler method (with a fixed step size $\Delta t = 1$) for solving the following ODE

$$\dot{\boldsymbol{y}}(t) = F_\theta(\boldsymbol{y}(t)),$$

on a time interval $[0, T]$, where the latent state $\boldsymbol{y}(t) \in \mathbb{R}^h$. As a result, the output of the neural network $\boldsymbol{y}(T)$ can be computed by calling an ODE solver. Although this observation was made earlier, Chen et al. (2018) turned it into a practical algorithm. This fostered the development of ODE solvers benefiting from the automatic differentiation capabilities of deep learning frameworks (e.g. PyTorch). Indeed, the solution $\boldsymbol{y}(T)$ is fed into a loss function, whose optimization requires backpropagating through the operations of the ODE solver. Furthermore, Chen et al. (2018) developed a backpropagation scheme that does not require storing a potentially high number of intermediate $\boldsymbol{y}(t_k)$.

When the data is sequential, recurrent neural networks (RNNs) are a popular modelling choice. RNNs use a dynamical system driven by an arbitrary length (discrete-time) signal $(\mathbf{x}_1, \dots, \mathbf{x}_\ell)$, in other words a sequence

$$\mathbf{y}_k = F_\theta(\mathbf{y}_{k-1}, \mathbf{x}_k), \tag{2.34}$$

to model the values of hidden units $\mathbf{y}_k$. The fact that the same transition function (or update rule) F_θ is used at every time step allows processing sequences of different lengths. The residual version of eq. (2.34) is given by

$$\mathbf{y}_k = \mathbf{y}_{k+1} + F_\theta(\mathbf{y}_{k-1}, \mathbf{x}_k).$$

As observed in Kidger et al. (2020), such RNNs can be regarded as a discrete approximation to some differential equation where the time-inhomogeneity depends on an underlying continuous signal $\boldsymbol{x}$

$$\dot{\boldsymbol{y}}(t) = F_\theta(\boldsymbol{y}(t), \boldsymbol{x}(t)),$$

and NCDEs can be seen as the continuous-time analogue of RNNs (Kidger et al., 2020, Sec. 3.3).

2.3.2 Extensions to less regular controls

Depending on the regularity assumptions about the control $\boldsymbol{x}(t) = \int_0^t \boldsymbol{\xi}(s)ds$, the integral in eq. (2.32) can be interpreted as a Riemann–Stieltjes, a stochastic, or even a rough integral.

Neural stochastic differential equations (Neural SDEs) are Neural CDEs where the control $\boldsymbol{\xi}$ is a sample path from a $\mathbb{R}^d$-dimensional Brownian motion. Besides, Neural SDEs can be used as generative models (trained either as VAEs or GANs) for time-series (Kidger et al., 2021).

Neural rough differential equations (Neural RDEs) (Morrill et al., 2021) generalize Neural CDEs in that they allow us to relax the regularity assumption on $\boldsymbol{x}$ and consider a larger class of controls. In practice, Neural RDEs are particularly well suited for long time-series. This is due to the fact that model can be evaluated via a numerical scheme from stochastic analysis (called the *log-ODE method*) over intervals much larger than what would be expected given the sampling rate or length of the time-series.

Remark. *Despite offering many advantages for modelling temporal dynamics, Neural CDEs are not designed to process signals varying both in space and in time such as physical fields arising from PDE-dynamics.*

Chapter 3

Scalable Gaussian Processes on Sequential Data

The objective of this thesis is to use tools from the theory of rough paths and reproducing kernels to tackle challenging learning problems on sequential data, such as multivariate time-series. A fundamental learning challenge that has been highlighted in Chapter 2 is making predictions and quantifying their uncertainty when the input data is sequential. *Gaussian process* (GP) models offer a flexible approach to achieve this goal. However, their main limitation is their poor scalability with the number of training examples. In this chapter, we propose to address this issue and develop a scalable sparse variational inference framework for GP models where the GP prior is indexed on sequential data. This framework also makes it possible to perform approximate inference for classication problems where the posterior is not available in closed form.

First, we construct inducing variables underpinning the sparse approximation so that the evaluation of the resulting ELBO does not require any matrix inversion. Second, we show that the gradients of the signature kernel are solutions of a hyperbolic PDE. This theoretical insight allows us to build an efficient backpropagation algorithm to optimize the ELBO. Finally, we empirically demonstrate the significant computational gains compared to existing methods, while achieving state-of-the-art performance for classification tasks on large datasets of up to 1 million multivariate time-series.

3.1 Introduction

Gaussian process (GP) models provide a sound mathematical framework for supervised learning that allows the incorporation of prior assumptions and provides uncertainty estimates when modelling unknown functions (Rasmussen and Williams, 2006). This is usually achieved by specifying a GP prior over functions with a suitable covariance function (or kernel) along with a conditional likelihood. With this, the problem boils down to that of estimating the posterior over the function (values) given the observed data.

However, this posterior distribution is often analytically intractable and, even when the conditional likelihood is Gaussian, GP models scale poorly on the number of observations N, with naïve approaches having a time complexity $\mathcal{O}(N^3)$. From a wide range of approximate techniques to scale inference in GP models to large datasets, "sparse" methods based on VI have emerged as one of the dominant approaches (Titsias, 2009). As explained in more details in Sec. 2.2.2, they consist in defining a family of approximate posteriors through M *inducing variables*, and selecting the distribution in this family that minimizes the KL divergence between the approximation and the true posterior. This is achieved by maximizing the so-called *Evidence lower bound* (ELBO). When the likelihood factorizes over datapoints, training can be done in minibatches of size $\tilde{N}$ resulting in a per iteration computational cost $\mathcal{O}(\tilde{N}M^2 + M^3)$, where the $\mathcal{O}(M^3)$ cost is due to the inversion of the covariance matrix of the M inducing variables. This yields significant computational savings when $M \ll N$.

In the seminal work of Titsias (2009) the inducing variables correspond to evaluations of the GP at M pseudo input locations, which typically results in a dense covariance matrix to invert. Subsequently, other ways of constructing inducing variables have been introduced in order to mitigate the $\mathcal{O}(M^3)$ cost (Burt et al., 2020b; Hensman et al., 2017). The core idea consists in defining (almost) independent inducing variables, such that their covariance matrix is (almost) diagonal. These inducing variables correspond to projections of the GP on basis functions, such that the covariance matrix is a Gramian matrix with respect to some inner-product. Orthogonal basis functions yield diagonal Gramian matrices, hence these methods are often referred to as *variational orthogonal features* (VOFs). However existing VOF methods are limited to stationary kernels on $\mathcal{X} \subset \mathbb{R}^d$ with $d \in \mathbb{N}_*$.

In this chapter, we propose to generalize the VOF paradigm to the case where the input space $\mathcal{X}$ is a set of *sequences* of vectors in $\mathbb{R}^d$. One may be tempted to naively concatenate each vector in a sequence of length ℓ to form a high-dimensional vector in $\mathbb{R}^{\ell d}$. However, in this case, existing VOF methods cannot be directly applied because they are limited to low dimensional vectors, with $d \leq 8$ (Dutordoir et al., 2020). Thus, one needs kernel functions specifically designed for sequential data. The *signature kernel* (Cass et al., 2020) is a natural choice that has recently emerged as a leading machine learning tool for learning on sequential data. In particular, Toth and Oberhauser (2020) have proposed GPSig, a GP inference framework, using an approximation of this covariance function (Király and Oberhauser, 2019), which achieves state-of-the-art performance on time-series classification tasks. Nevertheless, as in standard sparse variational approaches to GPs, the inducing inputs they chose (so called *inducing tensors*) are additional variational parameters to optimize, and the resulting covariance matrix is dense.

Here we develop SigGPDE, a new scalable sparse variational inference framework for GP models on sequential data. After a brief recap on the general principles of variational inference (Sec. 3.2) we identify a set of VOFs *naturally* associated with the signature kernel. These inducing variables do not depend on any variational parameter, as they are defined as projections of GP-samples onto an orthogonal basis for the RKHS associated to the signature kernel (Sec. 3.3). As a result, unlike the methods developed in Toth and Oberhauser (2020), in SigGPDE the optimization of the ELBO *does not require any matrix inversion.* Subsequently, we show that the gradients of the signature kernel are solutions of a *hyperbolic Partial differential equation* (PDE). This theoretical insight allows us to build an efficient backpropagation algorithm to optimize the ELBO (Sec. 3.4). Our experimental evaluation shows that SigGPDE is considerably faster than GPSig, whilst retaining similar predictive performances on datasets of up to 1 million multivariate time-series (Sec. 3.6).

3.2 Background

We begin with a general summary of variational inference for GPs. More details are provided in Sec. 2.2.2. In this section, it is assumed that the input space $\mathcal{X}$ is a subset of $\mathbb{R}^d$. Standard models with zero-mean GP priors and i.i.d. conditional likelihoods can be written as follows

$$f \sim \mathcal{GP}(0, k), \quad p(\mathbf{y} \,|\, \mathbf{f}) = \prod_{i=1}^{N} p(y_i \,|\, f(x_i)),$$

where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance function, $\mathbf{y} = (y_i)_{i=1}^N$ and $\mathbf{f} = (f(x_i))_{i=1}^N$. The general setting for sparse GPs consists in specifying a collection of M scalar variables as well as a joint distribution with variational parameters $\mathbf{m}$ (mean vector) and $\mathbf{\Sigma}$ (covariance matrix)

$$\mathbf{u} = (u_m)_{m=1}^M, \quad q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{\Sigma}).$$

These variables induce a family of approximate posteriors that are GPs with finite-dimensional marginal densities of the form $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \,|\, \mathbf{u})q(\mathbf{u})$. Considering any inputs $x, x' \in \mathcal{X}$, the mean function $\mu(\cdot)$ and covariance function $\nu(\cdot, \cdot)$ of these GPs are given by

$$\begin{aligned} \mu(x) &= \mathbf{c}_{\mathbf{u}}(x)'\mathbf{C}_{\mathbf{uu}}^{-1}\mathbf{m} \\ \nu(x, x') &= k(x, x') - \mathbf{c}_{\mathbf{u}}(x)'\mathbf{C}_{\mathbf{uu}}^{-1}(\mathbf{C}_{\mathbf{uu}} - \mathbf{\Sigma})\mathbf{C}_{\mathbf{uu}}^{-1}\mathbf{c}_{\mathbf{u}}(x'), \end{aligned}$$

where the vector $\mathbf{c_u}(x) \in \mathbb{R}^M$ and the matrix $\mathbf{C_{uu}} \in \mathbb{R}^{M\times M}$ are defined by,

$$[\mathbf{c_u}(x)]_m = \mathrm{Cov}\left(u_m, f(x)\right), \qquad [\mathbf{C_{uu}}]_{m,m'} = \mathrm{Cov}\left(u_m, u_{m'}\right)$$

for $1 \leq m, m' \leq M$. Provided that the inducing variables $\mathbf{u}$ are deterministic conditioned on f, one has the following lower bound (ELBO) on the marginal log-likelihood (Matthews, 2017)

$$\log p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}\,|\,\mathbf{f})] - \mathrm{KL}\left[q(\mathbf{u})\,\|\,p(\mathbf{u})\right], \tag{3.1}$$

where $p(\mathbf{u}) = \mathcal{N}(\mathbf{0}_M, \mathbf{C_{uu}})$. Maximizing the right-hand side of eq. (3.1) is equivalent to minimizing the KL divergence between the approximate GP and the true posterior (Matthews, 2017).

The original variational inference framework described in Titsias (2009) consists in setting $u_m = f(z_m)$ where $z_m \in \mathcal{X}$ is a pseudo-input, living in the same space as x, that can be fixed or optimized. The cost per iteration to optimize the ELBO is $\mathcal{O}(\tilde{N}M^2 + M^3)$, where $\tilde{N}$ is the minibatch size and M^3 is the cost of computing the inverse of $\mathbf{C_{uu}}$ via a Cholesky decomposition.

Recently, a considerable effort has been devoted to the construction of inducing variables $\mathbf{u}$ which yield a structured covariance matrix $\mathbf{C_{uu}}$ whose inversion has a reduced computational complexity (Hensman et al., 2017). This line of work is often referred to as *inter-domain* sparse GPs, owing to the fact that the pseudo inputs are not constrained to live in $\mathcal{X}$ as before. In particular, Burt et al. (2020b); Dutordoir et al. (2020) have shown that provided one can find an orthogonal basis of functions for the RKHS associated with the kernel $k(\cdot,\cdot)$, it is possible to define the inducing variables as projections of the GP samples onto this basis. This construction yields a diagonal matrix $\mathbf{C_{uu}}$.

3.3 Variational inference with orthogonal signature features

Here, we present our first contribution, namely the use of *orthogonal signature features* as inducing variables for GPs on sequential data. We begin with a summary of the theoretical background needed to define GPs endowed with the signature kernel. In this section, the input space is no longer a subspace of $\mathbb{R}^d$ but will be defined as a space of *paths* hereafter.

3.3.1 The signature

Consider a time-series $(\mathbf{x}_k)_{k=1}^{\ell}$ as a collection of points $\hat{\mathbf{x}}_k \in \mathbb{R}^{d-1}$ with corresponding time-stamps $t_k \in \mathbb{R}_+$ such that

$$(\mathbf{x}_k)_{k=1}^{\ell} = ((t_1, \hat{\mathbf{x}}_1), \dots, (t_\ell, \hat{\mathbf{x}}_\ell)),$$

with $0 = t_1 < \dots < t_\ell = T$. Let $\boldsymbol{x} : [0,T] \to \mathbb{R}^d$ be the piecewise linear interpolation of the data such that $\boldsymbol{x}(t_k) = (t_k, \hat{\mathbf{x}}_k)$. We denote by $\mathcal{X}(\mathbb{R}^d)$ the set of all continuous piecewise linear paths defined over the time interval $[0,T]$ and with values on $\mathbb{R}^d$. For any path $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^d)$ and any $i \in \{1,\dots,d\}$, we will denote its i^{th} *channel* by x^i so that at any time $t \in [0,T]$

$$\boldsymbol{x} : t \mapsto (x^1(t), \dots, x^d(t)),$$

as depicted in Fig. 3.1a with $d = 3$. The *signature* $S : \mathcal{X}(\mathbb{R}^d) \to H(\mathbb{R}^d)$ is a *feature map* defined for any path $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^d)$ as the following infinite collection of statistics

$$S(\boldsymbol{x}) = \Big(1, \big\{S^i(\boldsymbol{x})\big\}_{i=1}^{d}, \big\{S^{ij}(\boldsymbol{x})\big\}_{i,j=1}^{d}, \dots\Big),$$

where each term is a scalar equal to the coordinate iterated integral

$$S^{i_1 \dots i_k}(\boldsymbol{x}) = \underset{0<t_1<\dots<t_k<T}{\int \dots \int} dx^{i_1}(t_1) \dots dx^{i_k}(t_k). \tag{3.2}$$

Using the convention $(\mathbb{R}^d)^{\otimes 0} = \mathbb{R}$, the *feature space* $H(\mathbb{R}^d)$ associated to the signature is the Hilbert space

$$H(\mathbb{R}^d) = \left\{ \mathcal{A} \in \bigoplus_{k=0}^{\infty} (\mathbb{R}^d)^{\otimes k} \;\Big|\; \sum_{k=0}^{\infty} \|A_k\|^2_{(\mathbb{R}^d)^{\otimes k}} < +\infty \right\},$$

with inner-product $\langle \cdot, \cdot \rangle$ defined for any two $\mathcal{A}, \mathcal{B} \in H(\mathbb{R}^d)$ by

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{k=0}^{\infty} \langle A_k, B_k \rangle_{(\mathbb{R}^d)^{\otimes k}}.$$

Interpretability of the signature features An important aspect of sequential data is that the order of the observations is not commutative, in the sense that reordering the elements of a sequence can completely change its meaning. By definition the terms in the signature capture this fact. In effect, the coordinate iterated integral in eq. (3.2) is defined as an integral over the simplex $0 < t_1 < \dots < t_k < T$ which explicitly encodes the ordering of events happening across different channels $x^{i_1}, \dots, x^{i_k}$. This provides the signature features with a natural interpretability as highlighted several times in prior work (Arribas et al., 2018; Moore et al., 2019).

3.3.2 The signature kernel

The *signature kernel* $K : \mathcal{X}(\mathbb{R}^d) \times \mathcal{X}(\mathbb{R}^d) \to \mathbb{R}$ is a reproducing kernel which is defined for any pair of paths $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}(\mathbb{R}^d)$ as the following inner product

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle S(\boldsymbol{x}), S(\boldsymbol{y}) \rangle \tag{3.3}$$

in $H(\mathbb{R}^d)$. From the structure of $H(\mathbb{R}^d)$ and the properties of the signature, the signature kernel can be decomposed according to the expansion

$$K(\boldsymbol{x}, \boldsymbol{y}) = 1 + \sum_{n=1}^{\infty} \sum_{i_1 \ldots i_n} S^{i_1 \ldots i_n}(\boldsymbol{x}) S^{i_1 \ldots i_n}(\boldsymbol{y}), \tag{3.4}$$

where the inner summation is over the set $\{(i_1, \ldots, i_n) \mid i_1, \ldots, i_n \in \{1, \ldots, d\}\}$. In their recent article, Cass et al. (2020) provide a *kernel trick* for the signature kernel by proving the relation

$$K(\boldsymbol{x}, \boldsymbol{y}) = u(T, T),$$

where the function of two independent time variables $u : [0, T] \times [0, T] \to \mathbb{R}$ is the solution of the following hyperbolic PDE

$$\frac{\partial^2 u}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle \, u, \tag{3.5}$$

with boundary conditions $u(0, \cdot) = 1$ and $u(\cdot, 0) = 1$. This kernel trick is explained with simple arguments in the proof of Cass et al. (2020, Thm. 2.5). The sketch of the proof goes as follows: one first shows that the inner product in eq. (3.3) satisfies a double integral equation which comes from the fact that the signature itself solves an integral equation. Then, one uses the fundamental theorem of calculus to differentiate with respect to the two time variables to obtain the PDE. Next, we propose a simple parameterization of this kernel.

3.3.3 Parametrization of the signature kernel

In many real-world problems the input path $\boldsymbol{x}$ contains a large number of different channels, only some of which are relevant. Akin to an *automatic relevance determination* (ARD) parameterization, for any channel $i \in \{1, \ldots, d\}$ and time index $t \in [0, T]$, one can rescale each channel $x^i(t)$ by a scalar

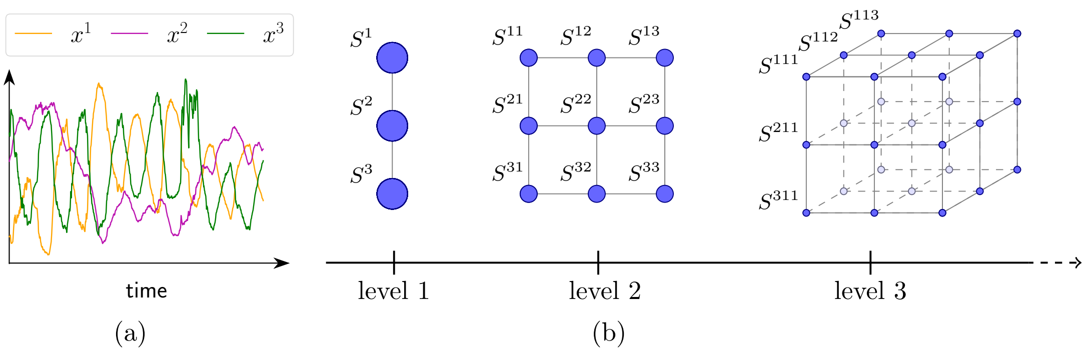


Figure 3.1: Illustration of the first terms of the signature $S(\boldsymbol{x})$ for a 3-dimensional path $\boldsymbol{x} : t \mapsto (x^1(t), x^2(t), x^3(t))$. Each blue circle corresponds to a signature feature $S^{i_1 \dots i_n}(\boldsymbol{x})$. The size of the circle reflects the feature importance according to the property $|S^{i_1 \dots i_n}(\boldsymbol{x})| = \mathcal{O}(1/n!)$. The first feature which is always equal to 1 is omitted in this schematic.

hyperparameter θ_i yielding the rescaled path

$$\boldsymbol{x}_\theta : t \mapsto (\theta_1 x^1(t), \dots, \theta_d x^d(t)). \tag{3.6}$$

The signature kernel of eq. (3.4) can be reparametrized as

$$K_\theta(\boldsymbol{x}, \boldsymbol{y}) = 1 + \sum_{n=1}^{\infty} \sum_{i_1 \dots i_n} S^{i_1 \dots i_n}(\boldsymbol{x}_\theta) S^{i_1 \dots i_n}(\boldsymbol{y}_\theta).$$

From eq. (3.2) it is straightforward to see that this parameterization corresponds to a particular rescaling of the signature features as per the following relation,

$$S^{i_1 \dots i_n}(\boldsymbol{x}_\theta) = \theta_{i_1} \dots \theta_{i_n} S^{i_1 \dots i_n}(\boldsymbol{x}) \tag{3.7}$$

for any $i_1, \dots, i_n \in \{1, \dots, d\}$. This result will turn out to be useful later on. In the sequel we will drop the subscript θ to unclutter the notation.

3.3.4 Variational orthogonal signature features

We can now build on the results from the previous sections to define the orthogonal signature features underlying our sparse variational inference framework for GPs on sequential data.

By Thm. 2.2.3 in Sec. 2.2.2, the RKHS $\mathcal{H}_K$ associated with the signature kernel can be defined as

$$\mathcal{H}_K = \left\{ \phi : \mathcal{X}(\mathbb{R}^d) \to \mathbb{R} \mid \exists \mathcal{A} \in H(\mathbb{R}^d) \text{ with } \phi(\boldsymbol{x}) = \langle \mathcal{A}, S(\boldsymbol{x}) \rangle \right\}.$$

Besides, for any $\phi \in \mathcal{H}_K$, there exists a unique element $\mathcal{A} \in H(\mathbb{R}^d)$ such that

$\phi(\boldsymbol{x}) = \langle \mathcal{A}, S(\boldsymbol{x})\rangle$ for all $\boldsymbol{x} \in \mathcal{X}$. This follows from Diehl and Reizenstein (2019, Lemma 3.4) and Xu and Zhang (2007, Lemma 5). Let $\phi_{\mathcal{A}}, \phi_{\mathcal{B}} \in \mathcal{H}_K$ such that

$$\begin{aligned} \phi_{\mathcal{A}} &: \boldsymbol{x} \mapsto \langle \mathcal{A}, S(\boldsymbol{x})\rangle \\ \phi_{\mathcal{B}} &: \boldsymbol{x} \mapsto \langle \mathcal{B}, S(\boldsymbol{x})\rangle. \end{aligned}$$

Their inner product in the RKHS $\mathcal{H}_K$ is defined by

$$\langle \phi_{\mathcal{A}}, \phi_{\mathcal{B}}\rangle_{\mathcal{H}_K} = \langle \mathcal{A}, \mathcal{B}\rangle.$$

The key to our setup is that the set of signature features (coordinate iterated integrals) forms an orthonormal basis for $\mathcal{H}_K$. Indeed, we have that

$$S^{i_1 \dots i_n} : \boldsymbol{x} \mapsto \langle \mathcal{E}_{i_1 \dots i_n}, S(\boldsymbol{x})\rangle$$

where $\mathcal{E}_{i_1 \dots i_n}$ denotes the element of $H(\mathbb{R}^d)$ where all the terms are zero except the n^{th} term which is $\mathbf{e}_{i_1} \otimes \dots \otimes \mathbf{e}_{i_n}$. Therefore, the inner product of any two signature features satisfies,

$$\left\langle S^{i_1 \dots i_n}, S^{j_1 \dots j_m}\right\rangle_{\mathcal{H}_K} = \begin{cases} 1, & \text{if } n = m, \text{ and } i_k = j_k, \ \forall k \in \{1, \dots, n\} \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

An important property of this orthonormal basis is that its elements are naturally ordered. This ordering is due to the property that for any path $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^d)$ the terms of the signature decay factorially Lyons et al. (2007),

$$\left|S^{i_1 \dots i_n}(\boldsymbol{x})\right| = \mathcal{O}\left(\frac{1}{n!}\right) \tag{3.9}$$

as illustrated in Fig. 3.1. Hence to index the signature orthogonal features $S^1, S^2, \dots, S^m, \dots$ we order first by increasing the level n, and then by sorting the multi-indices $(i_1, \dots, i_n)$ within a level. From eqs. (3.8) and (3.9) we define our inducing variables as orthogonal projections[1] of the GP onto the first M elements of the orthonormal signature basis

$$u_m = \langle f, S^m\rangle_{\mathcal{H}_K}, \quad 1 \le m \le M. \tag{3.10}$$

With this choice of inducing variables, we can easily deduce the following covariances (Hensman et al., 2017)

$$\mathbb{E}[u_m f(\boldsymbol{x})] = S^m(\boldsymbol{x}) \quad \text{and} \quad \mathbb{E}[u_m u_{m'}] = \delta_{m,m'}, \tag{3.11}$$

[1]Although f does not belong to $\mathcal{H}_K$ with probability 1 (Kanagawa et al., 2018), such projections are well defined because the span of the signature features is dense in the space of continuous functions on $\mathcal{X}(\mathbb{R}^d)$ and f is continuous (Toth and Oberhauser, 2020, Thm. 1.).

which implies that the covariance matrix $\mathbf{C_{uu}}$ is the identity. For any path $\boldsymbol{x} \in \mathcal{X}(\mathbb{R}^d)$ we use the convenient vector notation

$$\mathrm{sig}(\boldsymbol{x}) := \big(S^1(\boldsymbol{x}), \ldots, S^M(\boldsymbol{x})\big)' \in \mathbb{R}^M, \tag{3.12}$$

to obtain the approximate posterior $\mathcal{GP}(\mu, \nu)$ with mean and covariance functions defined by the following equations for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}(\mathbb{R}^d)$

$$\begin{aligned} \mu(\boldsymbol{x}) &= \mathrm{sig}(\boldsymbol{x})'\mathbf{m} \\ \nu(\boldsymbol{x}, \boldsymbol{x}') &= K(\boldsymbol{x}, \boldsymbol{x}') - \mathrm{sig}(\boldsymbol{x})'(\mathbf{I}_M - \boldsymbol{\Sigma})\mathrm{sig}(\boldsymbol{x}'). \end{aligned} \tag{3.13}$$

We note that the signature can be easily computed on real time-series using existing Python libraries Lyons (2010); Reizenstein and Graham (2018). The same holds for the signature kernel via the python package available at `https://github.com/crispitagorico/sigkernel`. During this thesis we contributed to the development of this library, notably to enable fast forward and backward computations on GPU, as we shall explain next.

3.4 Reverse-mode automatic differentiation for the signature kernel

In order to optimize the ELBO with respect to the parameters θ, in principle one needs to take derivatives of the signature features in eq. (3.12) with respect to their input path. Here, the scaling property of the signature—as seen in eq. (3.7)—becomes relevant, as it provides an easier way to obtain the gradients with respect to θ. In addition, the signatures can be precomputed on the training set. However, one also needs to differentiate the signature kernel in eq. (3.13) with respect to each of its input paths. Recall that the signature kernel solves the PDE in eq. (3.5) and is evaluated using appropriate PDE numerical solvers. Therefore, differentiation could be carried out by leveraging the automatic differentiation tools of modern deep learning libraries (Tensorflow, PyTorch etc.). However, backpropagating through the operations of the PDE solver can be inefficient.

Here we show that the gradients of the signature kernel can be efficiently computed *without backpropagating through the operations of the PDE solver* as they are the solutions of a second PDE analogous to eq. (3.5). The ability not to rely on automatic differentiation allows for an efficient fitting of SigGPDE both in terms of time complexity and memory cost.

Algorithm 3 BackwardPDESolve via the PDE in eq. (3.19)

```
1: Input: Path $\boldsymbol{x}$, localised impulses $\Gamma = \{\boldsymbol{\gamma}_{k,i}\}_{k=1,i=1}^{k=\ell,i=d}$.
2: // Boundary conditions for the augmented state
3:     $u(0,\cdot) = 1, \quad u(\cdot,0) = 1$
4:     $[u_\Gamma(0,\cdot)]_{k,i} = 0, \quad [u_\Gamma(\cdot,0)]_{k,i} = 0$
5:     $u_{\text{aug}}(0,\cdot) = [u(0,\cdot), u_\Gamma(0,\cdot)], \quad u_{\text{aug}}(\cdot,0) = [u(\cdot,0), u_\Gamma(\cdot,0)]$
6: // Dynamics for the augmented state
7:     def aug_dynamics($[u(s,t), u_\Gamma(s,t)], s, t$):
8:         $\Delta(s,t)_{k,i} = \langle \dot{\boldsymbol{\gamma}}_{k,i}(s), \dot{\boldsymbol{x}}(t)\rangle$
9:         return $[\langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{x}}(t)\rangle u(s,t), \ \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{x}}(t)\rangle u_\Gamma(s,t) + \Delta(s,t)u(s,t)]$
10:    $[u(T,T), u_\Gamma(T,T)] = \mathsf{PDESolve}(u_{\text{aug}}(0,\cdot), u_{\text{aug}}(\cdot,0), \text{aug\_dynamics}, T, T)$
11: Output: $2 \cdot u_\Gamma(T,T)$                    // Gradients of the kernel
```

3.4.1 Differentiating along the direction of a path

Consider a time-series $(\mathbf{x}_k)_{k=1}^{\ell}$ as defined in Sec. 3.3.1. Each vector $\mathbf{x}_k$ in the sequence can be written with respect to the canonical basis $(\mathbf{e}_i)_{i=1}^{d}$ of $\mathbb{R}^d$ as

$$\mathbf{x}_k = \sum_{i=1}^{d} \mathrm{x}_k^i \mathbf{e}_i.$$

Let $\boldsymbol{x} : [0,T] \to \mathbb{R}^d$ be the piecewise linear interpolation of the data such that $\boldsymbol{x}(t_k) = \mathbf{x}_k$. Similarly, for a second time-series $(\mathbf{y}_k)_{k=1}^{\ell'}$ and resulting piecewise linear interpolation $\boldsymbol{y}$. Recall the definition of signature kernel as

$$K_\theta(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x}_\theta, \boldsymbol{y}_\theta),$$

where $\boldsymbol{x}_\theta$ and $\boldsymbol{y}_\theta$ are the rescaled paths of eq. (3.6). By the chain rule, one has

$$\frac{\partial K_\theta}{\partial \theta} = \frac{\partial K}{\partial \boldsymbol{x}_\theta}\frac{\partial \boldsymbol{x}_\theta}{\partial \theta} + \frac{\partial K}{\partial \boldsymbol{y}_\theta}\frac{\partial \boldsymbol{y}_\theta}{\partial \theta}. \tag{3.14}$$

Hence, to formulate a backpropagation algorithm in a rigorous way, we need to give meaning to the following gradients

$$\left\{\frac{\partial}{\partial \mathrm{x}_k^i} K(\boldsymbol{x}, \boldsymbol{y})\right\}_{k,i=1}^{\ell,d}. \tag{3.15}$$

The technical difficulty here consists in reconciling the continuous nature of the input path $\boldsymbol{x}$ and the discrete nature of the locations x_k^i, where one wants to compute the gradients, which are given by the knots of the time-series $(\mathbf{x}_k)_{k=1}^{\ell}$.

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 eq. (3.15). These definitions will be followed by the main result of this section, namely that the directional derivative of K solves another PDE similar to eq. (3.5) for the signature kernel, for which we derive an explicit solution via the *technique of variation of parameters* (Thm. 3.4.1).

Definition 12. *For any $k = 1, \dots, \ell$ and any $i = 1, \dots, d$ define the localised impulse $\boldsymbol{\gamma}_{k,i} : [0, T] \to \mathbb{R}^d$ as the solution of the following ordinary differential equation (ODE)*

$$\dot{\boldsymbol{\gamma}}_{k,i}(t) = \frac{1}{\ell}\mathbb{1}_{[(k-1)/\ell, k/\ell)]}(t)\mathbf{e}_i, \quad \boldsymbol{\gamma}_{k,i}(0) = \mathbf{0}.$$

Definition 13. *For any path $\boldsymbol{\gamma} \in \mathcal{X}(\mathbb{R}^d)$ the directional derivative of the signature kernel K along $\boldsymbol{\gamma}$ is defined as*

$$K_{\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y}) := \frac{\partial}{\partial \epsilon} K\Big(\boldsymbol{x} + \epsilon \boldsymbol{\gamma}, \boldsymbol{y}\Big)\Big|_{\epsilon=0}.$$

Each gradient of the signature kernel at the knot x_k^i reported in eq. (3.15) can be identified with the directional derivative of the signature kernel along the localized impulse $\boldsymbol{\gamma}_{k,i}$ of Def. 12

$$\frac{\partial}{\partial \mathrm{x}_k^i} K(\boldsymbol{x}, \boldsymbol{y}) := K_{\boldsymbol{\gamma}_{k,i}}(\boldsymbol{x}, \boldsymbol{y}).$$

3.4.2 A PDE for the gradients of the signature kernel

Algorithm 4 BackwardPDESolve via Thm. 3.4.1

```
1: **Input:** Path $\boldsymbol{x}$, localised impulses $\Gamma = \{\boldsymbol{\gamma}_{k,i}\}_{k=1,i=1}^{k=\ell,i=d}$
2: // Boundary conditions for the augmented state
3:     $u(0,\cdot) = 1, \ \ u(\cdot,0) = 1, \ \ \tilde{u}(0,\cdot) = 1, \ \ \tilde{u}(\cdot,0) = 1$
4:     $u_{\mathrm{aug}}(0,\cdot) = [u(0,\cdot), \tilde{u}(0,\cdot)], \ \ u_{\mathrm{aug}}(\cdot,0) = [u(\cdot,0), \tilde{u}(\cdot,0)]$
5: // Dynamics for the augmented state
6:     def aug_dynamics($[u(s,t), \tilde{u}(s,t)], s, t$):
7:         return $[\langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{x}}(t)\rangle u(s,t), \ \langle \dot{\boldsymbol{x}}(T-s), \dot{\boldsymbol{x}}(T-t)\rangle \tilde{u}(s,t)]$
8: // Keep the solutions at each $(s,t)$
9:     $[u, \tilde{u}] = \mathsf{PDESolve}(u_{\mathrm{aug}}(0,\cdot), u_{\mathrm{aug}}(\cdot,0), \mathrm{aug\_dynamics}, T, T)$
10:    $\Delta(s,t)_{k,i} = \langle \dot{\boldsymbol{\gamma}}_{k,i}(s), \dot{\boldsymbol{x}}(t)\rangle$
11:    $\tilde{u}_{\mathrm{rev}}(s,t) \leftarrow \tilde{u}(T-s, T-t)$
12:    $u_\Gamma = \mathrm{integrate}(u \cdot \tilde{u}_{\mathrm{rev}} \cdot \Delta)$        // Simple final TensorFlow operations
13: **Output:** $2 \cdot u_\Gamma$                                   // Gradients of the kernel
```

Recall that the signature kernel solves the following PDE

$$\frac{\partial^2 u}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle \, u. \tag{3.16}$$

Integrating both sides with respect to s and t one obtains

$$u(s,t) = 1 + \int_{p=0}^{s} \int_{q=0}^{t} u(p,q) \, \langle \dot{\boldsymbol{x}}(p), \dot{\boldsymbol{y}}(q) \rangle \, dp dq. \tag{3.17}$$

Let us denote by $u_{\boldsymbol{\gamma}} : [0,T] \times [0,T] \to \mathbb{R}$ the directional derivative $K_{\boldsymbol{\gamma}}$ evaluated at the restricted paths $\boldsymbol{x}|_{[0,s]}$ and $\boldsymbol{y}|_{[0,t]}$

$$u_{\boldsymbol{\gamma}}(s,t) := K_{\boldsymbol{\gamma}} \left(\boldsymbol{x}|_{[0,s]}, \boldsymbol{y}|_{[0,t]} \right). \tag{3.18}$$

The combination of eqs. (3.17) and (3.18) yields the relation

$$\begin{aligned} u_{\boldsymbol{\gamma}}(s,t) &= \frac{\partial}{\partial \epsilon} K \Big((\boldsymbol{x} + \epsilon \boldsymbol{\gamma})|_{[0,s]}, \boldsymbol{y}|_{[0,t]} \Big) \Big|_{\epsilon=0} \\ &= \frac{\partial}{\partial \epsilon} \left(\int_0^s \int_0^t u(p,q) \langle \dot{\boldsymbol{x}}(p) + \epsilon \dot{\boldsymbol{\gamma}}(p), \dot{\boldsymbol{y}}(q) \rangle dp dq \right)_{\epsilon=0} \\ &= \int_0^s \int_0^t \left(u_{\boldsymbol{\gamma}}(p,q) \, \langle \dot{\boldsymbol{x}}(p), \dot{\boldsymbol{y}}(q) \rangle + u(p,q) \, \langle \dot{\boldsymbol{\gamma}}(q), \dot{\boldsymbol{y}}(q) \rangle \right) dp dq. \end{aligned}$$

Hence, differentiating the last equation, first with respect to t and then s, we find that the directional derivative $K_{\boldsymbol{\gamma}}$ of the signature kernel along the path $\boldsymbol{\gamma}$ solves the following PDE

$$\frac{\partial^2 u_{\boldsymbol{\gamma}}}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle \, u_{\boldsymbol{\gamma}} + \langle \dot{\boldsymbol{\gamma}}(s), \dot{\boldsymbol{y}}(t) \rangle \, u, \tag{3.19}$$

with boundary conditions $u_{\boldsymbol{\gamma}}(0, \cdot) = 0$ and $u_{\boldsymbol{\gamma}}(\cdot, 0) = 0$. As a result, the gradients in eq. (3.15) of the signature kernel with respect to each of its input paths can be computed with a single call to a PDE solver, which concatenates the original state and the partial derivatives in eq. (3.19). Each partial derivative follows the dynamics of eq. (3.19) where one replaces the direction $\boldsymbol{\gamma}$ by the relevant localized impulse $\boldsymbol{\gamma}_{k,i}$, $\boldsymbol{\tau}_{k,i}$ for $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. We outline the resulting procedure in Alg. 3, where the partial derivatives are aggregated into $u_\Gamma(s,t)$. Note that to optimize the ELBO we only need to differentiate $K(\boldsymbol{x}, \boldsymbol{x})$, which is the case presented in the algorithm. The generalization to the case $K(\boldsymbol{x}, \boldsymbol{y})$ is straightforward using the chain rule in eq. (3.14).

3.4.3 An explicit solution by variation of parameters

From this second PDE in eq. (3.19) we derive the following theorem (proved in Appendix A.1), which allows us to compute the directional derivative $K_{\boldsymbol{\gamma}}$ of the signature kernel directly from its evaluations at $\boldsymbol{x}, \boldsymbol{y}$ and at $\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}}$, where $\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}}$

are respectively the paths $\boldsymbol{x}, \boldsymbol{y}$ reversed in time (that is, $\overleftarrow{\boldsymbol{x}}(t) = \boldsymbol{x}(T-t)$).

Theorem 3.4.1. *For any $\boldsymbol{\gamma} \in \mathcal{X}(\mathbb{R}^d)$ the directional derivative $K_{\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y})$ of the signature kernel along the path $\boldsymbol{\gamma}$ satisfies the following relation*

$$K_{\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y}) = \int_0^T \int_0^T u(s,t)\widetilde{u}(T-s, T-t)\langle \dot{\boldsymbol{\gamma}}(s), \dot{\boldsymbol{y}}(t)\rangle ds dt,$$

where $\widetilde{u}(s,t) = K\left(\overleftarrow{\boldsymbol{x}}|_{[0,s]}, \overleftarrow{\boldsymbol{y}}|_{[0,t]}\right)$ and where $\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}}$ are respectively the paths $\boldsymbol{x}, \boldsymbol{y}$ reversed in time.

The full backpropagation procedure is described in Alg. 4.

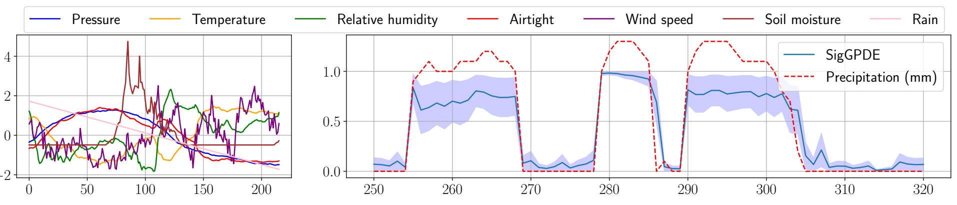
3.5 Related work

In this section, we expand on the material presented in Sec. 3.2, focusing on the most recent approaches to scalable GPs on $\mathbb{R}^d$ with VOFs and on sparse GPs for sequential data.

Variational Fourier Features In Hensman et al. (2017) the inducing variables are defined for scalar inputs ($\mathcal{X} = \mathbb{R}$) as projections of the GP-sample onto the truncated Fourier basis. This type of inducing variables is constructed for GPs with Matérn-type kernels. Although the resulting covariance matrix of the inducing variables is not diagonal, it can be decomposed into the sum of a diagonal matrix and rank one matrices. As a result, it can be inverted using the *Woodbury identity*, which makes it possible to scale GP inference on $\mathbb{R}$. The generalization to GPs on $\mathbb{R}^d$ is done by taking the outer product of the Fourier basis on $\mathbb{R}$.

Eigenfunction inducing features Closest to our work are the eigenfunction inducing features developed by Burt et al. (2020a), where the inducing variables are also defined as projections of the GP-sample onto an orthogonal basis of functions for the RKHS associated with the GP kernel. This is based on a *Mercer's expansion* of the kernel. From there, one identifies these orthogonal basis functions by solving an eigendecomposition problem. For example, Dutordoir et al. (2020) map the input data to the hypersphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$ and then show that *spherical harmonics* form an orthogonal basis for the RKHSs associated with *zonal kernels* defined on $\mathbb{S}^{d-1}$.

GPs with signature covariances Toth and Oberhauser (2020) propose a different sparse GP inference framework for sequential data with signature covariances (GPSig). In this work the inducing variables are either taken to be *inducing sequences* (IS) in the original input space (GPSig-IS) of sequences



(a) Input time series (b) Output predictions

Figure 3.2: Weather forecast dataset. (a) One (standard scaled) multivariate time-series $(\mathbf{x}_k)_{k=1}^{\ell}$ in input to the GP model. (b) Posterior mean of the SigGPDE GP when evaluated at multiple input time-series like $(\mathbf{x}_k)_{k=1}^{\ell}$ on the test set. The actual precipitation amount is given for reference.

or *inducing tensors* (IT) in the corresponding feature space (GPSig-IT). The chosen covariance function is an approximation of the signature kernel based on truncating the signature to a finite level n. For GPSig-IT, this truncation makes the feature space finite-dimensional and allows one to optimize the inducing tensors defined over such truncated space. Unlike our method, the inducing tensors are additional variational parameters to optimize. The covariance matrix $\mathbf{C}_{\mathbf{uu}}$ is dense and its inversion incurs an additional $\mathcal{O}(M^3)$ cost. In Table 3.1 we compare the computational complexities of GPSig-IT, GPSig-IS and SigGPDE. A similar table for the memory complexity can be found in Appendix A.3.

Table 3.1: Comparison of time complexities. M is the number of inducing variables, $\tilde{N}$ the batch size, d the number of channels in the time-series, ℓ the length of the sequences, n the truncation level (for GPSig-IT and GPSig-IS) and $\tilde{\ell}$ the length of the inducing sequences. The last line of the table corresponds to linear algebra operations including matrix multiplication and matrix inversion.

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$\mathbf{C}_{\mathbf{uu}}$	$\mathcal{O}(1)$	$\mathcal{O}(n^2M^2d)$	$\mathcal{O}((n+d)M^2\tilde{\ell}^2)$
$\mathbf{C}_{\mathbf{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell d)$	$\mathcal{O}((n+d)\tilde{N}M\tilde{\ell}\ell)$
diag($\mathbf{C}_{\mathbf{ff}}$)	$\mathcal{O}(d\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$
Lin. Alg.	$\mathcal{O}(\tilde{N}M^2)$	$\mathcal{O}(\tilde{N}M^2+M^3)$	$\mathcal{O}(\tilde{N}M^2+M^3)$

3.6 Experiments

In this section, we benchmark SigGPDE against GPSig-IT and GPSig-IS from Toth and Oberhauser (2020) on various multivariate time-series classification tasks. For GPSig-IS, we use inducing sequences of length $\tilde{\ell}=5$ as recommended in Toth and Oberhauser (2020). We highlight how SigGPDE performs competitively in terms of accuracy and uncertainty quantification but with a significant speed-up in the fitting compared to the other baselines.

We use a mixture of UEA & UCR time-series datasets and real-world data for the final example.

We measure the classification accuracy on the test set, assess the uncertainty quantification with mean negative log-predictive probabilities (NLPP) and report the runtime per iteration. For each dataset, all models are trained 3 times using a random training-validation split. The validation split is used to monitor the NLPP when optimizing the hyperparameters of the models. Further details on the training procedure can be found in Appendix A.2. All code is written in TensorFlow using GPFlow De G. Matthews et al. (2017).

3.6.1 Classifying digits in sequential MNIST

We start with a handwritten digit classification task, where writers were asked to draw the digits from 0 to 9. The instances are made up of 2-d trajectories of the pen traced across a digital screen. The trajectories are of length $\ell = 8$. The training and test sets are of size 7 494 and 3 498 respectively. We made use of $M = 500$ inducing features. In the results reported in Table 3.2, SigGPDE achieves even better accuracy and NLPP than the GPSig baselines, whilst being almost twice as fast than GPSig-IT.

Table 3.2: Classification for sequential MNIST (PenDigits). The higher the Mean Acc. and the lower the NLPP the better.

Model	Mean Acc.	NLPP	Time
GPSig-IS	97.42 ± 0.17	0.096 ± 0.005	0.186 (s/iter)
GPSig-IT	96.66 ± 0.59	0.115 ± 0.018	0.036 (s/iter)
SigGPDE	97.73 ± 0.13	0.085 ± 0.001	0.022 (s/iter)

3.6.2 Detecting whale call signals

In this example the task is to classify audio signals and distinguish one emitted from right whales from noise. The dataset (called RightWhaleCalls in the UEA archive) contains 10 934 train cases and 5 885 test cases. The signals are one-dimensional, sampled at 2kHz over 2 seconds, hence of length 4 000. We tackle this problem as a multivariate time-series classification task, by considering the spectrogram of the univariate audio signal. The resulting streams are made of 29 channels corresponding to selected frequencies and are 30 time steps long. The results in Table 3.3 are obtained with $M = 700$ and show the significant speed-up of SigGPDE by almost one order of magnitude compared to GPSig.

This speed-up is compensated by a minimal decrease in performance both in terms of accuracy and NLPP.

Table 3.3: Classification for whale call signals

Model	Mean Acc.	NLPP	Time
GPSig-IS	86.97 ± 0.11	0.367 ± 0.005	0.438 (s/iter)
GPSig-IT	87.70 ± 0.42	0.357 ± 0.003	0.048 (s/iter)
SigGPDE	86.76 ± 0.36	0.382 ± 0.002	0.008 (s/iter)

3.6.3 Large scale classification of satellite time-series

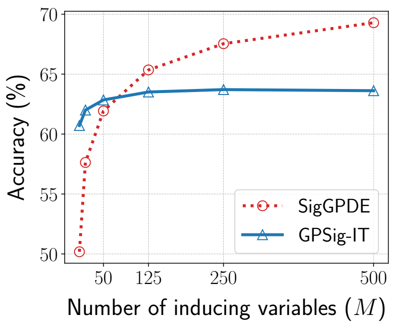
We now consider a large-scale classification example on 1 million time-series. The time-series in this dataset represent a vegetation index, calculated from remote sensing spectral data. The 24 classes represent different types of land cover (Petitjean et al., 2012). The aim in classifying these time-series of length $\ell = 46$ is to map different vegetation profiles to different types of crops and forested areas. Due to the sheer size of this dataset we only compare SigGPDE to GPSig-IT as GPSig-IS is not scalable to such large dataset. In Fig. 3.3 we report the accuracy, time per iteration and ELBO by progressively increasing the number of inducing variables. Compared to SigGPDE, GPSig-IT has additional variational parameters, namely the inducing tensors. This extra flexibility explains the better performance of GPSig-IT when few inducing variables are used. However, as the number of inducing features increases, SigGPDE catches up and outperforms GPSig-IT in all monitored metrics.

3.6.4 Weather forecast

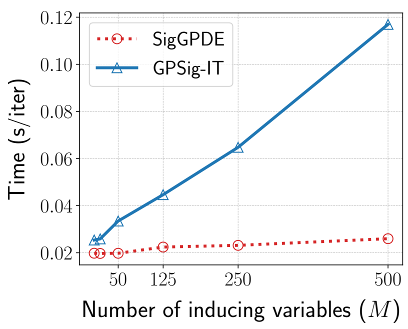
In this last example, we use a dataset of climatic variables recorded by the Max Planck Institute for Biogeochemistry[2] in the weather stations of WS Beutenberg and WS Saaleaue from 2004-2020. The data consists of 7-dimensional time-series recorded once per 10 minutes where each channel represents a weather feature such as temperature, pressure, humidity, etc. The goal is to predict whether it will rain in the next hour from the trajectory of all other features in the preceding 6 hours. To obtain binary labels for the classification task, we set the label to 1 if the precipitation is larger than 1mm and to 0 otherwise. The inference mechanism is depicted on Fig. 3.2.

A key feature, proper to our model SigGPDE, is its interpretability. Looking at the variational mean vector $\mathbf{m}$ in eq. (3.13), we can extract the terms with

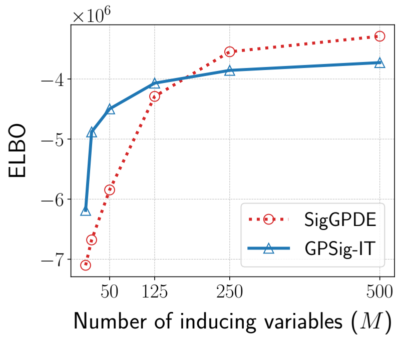
[2]`https://www.bgc-jena.mpg.de/wetter/`



(a)



(b)



(c)

Figure 3.3: Large scale (1M) classification of satellite time-series. Comparison of various metrics as functions of inducing variables in (a), (b) and (c).

highest relevance learned by the model. As discussed in Sec. 3.3.1, thanks to the corresponding signature features, it is possible to infer which signature features used by the GP are more responsible for the result produced. The most relevant predictive features for this weather forecast experiment are represented in Fig. 3.4.

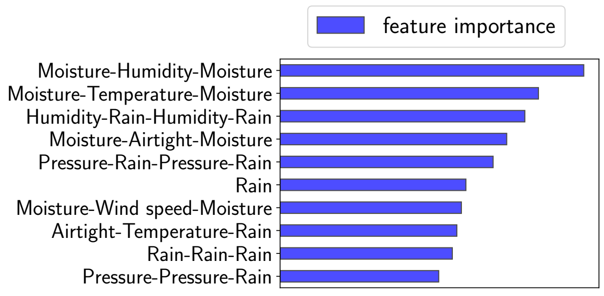


Figure 3.4: Top 10 signature features (by importance) used by SigGPDE to predict whether or not it will rain in the next hour from previous weather data. Each feature corresponds to an iterated integral $S^{i_1 \cdots i_n}(\boldsymbol{x})$ where each $i_k \in \{1, \ldots, 7\}$. The 7 channels of the input time-series represent Moisture, Humidity, Temperature, Rain, Airtight, Pressure, and Wind speed. For example we can read that the top feature corresponds to $n = 3$ and $(i_1, i_2, i_3) = (1, 2, 1)$.

3.7 Conclusion

In this chapter we have developed SigGPDE, a framework for performing variational inference for GP models on sequential data with orthogonal signature features. Firstly, we constructed inducing variables so that their covariance matrix is diagonal. Secondly, we showed that the gradients of the signature kernel are solutions of a hyperbolic PDE. As a result the ELBO is cheap to evaluate, as gradient descent does not require backpropagating through the operations of the PDE solver. We benchmarked SigGPDE against the state-of-the-art GPSig on different time-series classification tasks, showing a significant speed-up and similar performance.

Chapter 4

Distribution Regression on Sequential Data

In the previous chapter we discussed how large datasets of multivariate time-series lead to significant computational challenges when fitting Gaussian process models. We leveraged the properties of the signature kernel and its associated RKHS to scale up the learning algorithm. In some instances, one can instead reduce the size of the dataset by representing sets of inputs by empirical distributions. This is common practice in the context of *Distribution regression* (DR), supervised learning problems, where labels are only available for *sets* of inputs instead of individual inputs.

In this chapter, we develop a rigorous mathematical framework for DR where inputs are data streams. Leveraging properties of the *expected signature* and the recent *signature kernel trick*, we introduce two new learning techniques: 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 multivariate time-series, on both synthetic and real-world examples from thermodynamics, mathematical finance and agricultural science.

4.1 Introduction

DR on sequential data describes the task of learning a function from a set of data streams to a single scalar target. For example, in thermodynamics (Fig. 4.1) one may be interested in determining the temperature of a gas from the set of trajectories described by its particles (Hill, 1986; Reichl, 1999; Schrödinger, 1989). Similarly, in quantitative finance, practitioners may wish to estimate mean-reversion parameters from observed market dynamics (Balvers et al., 2000; Gatheral et al., 2018; Papavasiliou et al., 2011). Another example 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 (Dahikar and Rode, 2014; Panda et al., 2010; You et al., 2017).

DR techniques (Oliva et al., 2014; Póczos et al., 2013; Szabó et al., 2016) have been successfully applied to handle situations where the inputs in each set are vectors in $\mathbb{R}^d$. Recently, there has been an increased interest in extending these techniques to non-standard inputs such as images (Law et al., 2018b) or persistence diagrams (Kusano et al., 2016). However, DR for sequential data, such as multivariate time-series, has been largely ignored. The main challenges in this direction are the correlations of the variables in a sequence, which naturally come with an order, and the fact that in many real-world scenarios, the points in a sequence are irregularly distributed across time.

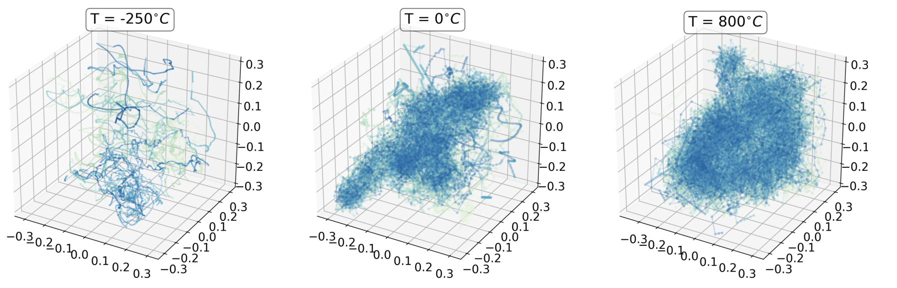


Figure 4.1: Simulation of the trajectories traced by 20 particles of an ideal gas in a 3-d box under different thermodynamic conditions. Higher temperatures equate to a higher internal energy in the system which increases the number of collisions resulting in different large-scale dynamics of the gas.

In this chapter, we propose a framework for DR that addresses precisely the setting in which the inputs within each set are data streams, mathematically thought of as *continuous paths of bounded variation* (Sec. 4.2). We formulate two distinct approaches, one feature-based and the other kernel-based, both relying on a recent tool from stochastic analysis known as the *expected signature* (Chevyrev and Oberhauser, 2018; Chevyrev et al., 2016; Lyons et al., 2015; Ni, 2012). First, we construct a new set of features that are universal, in the sense that any continuous function on distributions on paths can be uniformly well approximated by a linear combination of these features (Sec. 4.3.1). Secondly, we introduce a universal kernel on distributions on paths given by the composition of the expected signature and a Gaussian kernel (Sec. 4.3.2), which can be evaluated with a kernel trick. The former method is more suitable to datasets containing a large number of low-dimensional streams, whilst the latter is better for datasets with a low number of high-dimensional streams. We demonstrate the versatility of our methods to handle interacting trajectories like the ones in Fig. 4.1. We show how these two methods can be used to provide practical DR algorithms for time-series, which are robust to irregular sampling and achieve state-of-the-art performance on synthetic and real-world examples (Sec. 4.5).

4.1.1 Problem definition

Consider M input-output pairs $\{(\mathscr{S}_i, y_i)\}_{i=1}^M$, where each pair is given by a scalar target $y_i \in \mathbb{R}$ and a finite collection $\mathscr{S}_i = \{\underline{\mathbf{x}}_{i,p}\}_{p=1}^{N_i}$ of $(d-1)$-dimensional time-series (with $d > 1$), each of the form

$$\underline{\mathbf{x}} = ((t_1, \mathbf{x}_1), \ldots, (t_\ell, \mathbf{x}_\ell)),$$

of possibly unequal lengths $\ell \in \mathbb{N}_*$, with time-stamps $0 < t_1 < \ldots < t_\ell < T$ and values $\mathbf{x}_k \in \mathbb{R}^{d-1}$. Every time-series can be naturally embedded into a continuous path of bounded variation

$$\boldsymbol{x} : [0, T] \to \mathbb{R}^d,$$

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

4.2 Background

We begin by formally introducing the class of paths and the set of probability measures we are considering.

4.2.1 Paths and probability measures on paths

Let $T > 0$ and consider the closed time interval $[0, T]$. Let E be a Hilbert space with inner product $\langle \cdot, \cdot \rangle_E$. For applications we will take $E = \mathbb{R}^d$, with $d \in \mathbb{N}_*$. We denote by $\mathcal{X}(E)$ the Banach space (Friz and Victoir, 2010) of continuous functions of bounded variation $\boldsymbol{x} : [0, T] \to E$ equipped with the norm

$$\|\boldsymbol{x}\| = \mathcal{V}(\boldsymbol{x}) + \sup_{t \in [0,T]} \|\boldsymbol{x}(t)\|_E .$$

We will refer to any element $\boldsymbol{x} \in \mathcal{X}(E)$ as an E-valued *path*. Given a compact subset of paths $U \subset \mathcal{X}(E)$, with respect to the topology induced by $\|\cdot\|$, we denote by $\mathcal{P}(U)$ the set of (Borel) *probability measures* on U.

In the previous chapter, we leveraged the signature, which has been shown to be an ideal feature map for paths (Lyons, 2014). In this chapter, we make use of the expected signature, an appropriate feature map for probability measures on paths. Both feature maps take values in the same feature space. In the next section we introduce the necessary mathematical background to describe the structure of this space.

4.2.2 A canonical Hilbert space of tensors

In what follows, $\oplus$ and $\otimes$ will denote the direct sum and the tensor product of vector spaces, respectively. For example, $(\mathbb{R}^d)^{\otimes 2} = \mathbb{R}^d \otimes \mathbb{R}^d$ can be identified with the space of $d \times d$ matrices and $(\mathbb{R}^d)^{\otimes 3}$ can be identified wit the space of $d \times d \times d$ tensors. By convention $E^{\otimes 0} = \mathbb{R}$. Consider the following vector space

$$\mathscr{T}(E) = \bigoplus_{k=0}^{\infty} E^{\otimes k} = \mathbb{R} \oplus E \oplus E^{\otimes 2} \oplus \dots \ .$$

If $\{\mathbf{e}_i\}_{i\in\{1,\dots,d\}}$ is a basis of E, the elements $\{\mathbf{e}_{i_1} \otimes \dots \otimes \mathbf{e}_{i_k}\}_{(i_1,\dots,i_k)\in\{1,\dots,d\}^k}$ form a basis of $E^{\otimes k}$. For any $\mathcal{A} \in \mathscr{T}(E)$ we denote by $A_k \in E^{\otimes k}$ the k^{th}-tensor component of $\mathcal{A}$. If E is a Hilbert space with inner product $\langle \cdot, \cdot \rangle_E$, then there exists a canonical inner product $\langle \cdot, \cdot \rangle_{E^{\otimes k}}$ on each $E^{\otimes k}$ which extends by linearity to an inner product

$$\langle \mathcal{A}, \mathcal{B} \rangle_{H(E)} = \sum_{k\geq 0} \langle A_k, B_k \rangle_{E^{\otimes k}},$$

on $H(E) := \{\mathcal{A} \in \mathscr{T}(E) \mid \sum_{k=0}^{\infty} \|A_k\|^2_{E^{\otimes k}} < +\infty\}$ that thus becomes also a Hilbert space (Chevyrev and Oberhauser, 2018, Sec. 3).

4.2.3 The signature of a path

The *signature* (Chen, 1957; Lyons, 2014, 1998) turns the complex structure of a path $\boldsymbol{x}$ into a simpler vectorial representation given by an infinite sequence of iterated integrals. We recall that, in this thesis, the iterated integrals are defined in the classical *Riemann-Stieltjes* sense.

Definition 14. *The signature $S : \mathcal{X}(E) \to H(E)$ is the map defined in the following way: the 0^{th} component is always $S_0(\boldsymbol{x}) = 1$, whilst all the others are defined as*

$$S_k(\boldsymbol{x}) = \underset{0<t_1<\dots<t_k<T}{\int \dots \int} d\boldsymbol{x}(t_1) \otimes \dots \otimes d\boldsymbol{x}(t_k). \tag{4.1}$$

The k^{th} component $S_k(\boldsymbol{x})$ can be represented by its coordinates with respect to the canonical basis of $E^{\otimes k}$

$$S_k(\boldsymbol{x}) = \sum_{i_1 \dots i_k} S^{i_1 \dots i_k}(\boldsymbol{x}) \mathbf{e}_{i_1} \otimes \dots \otimes \mathbf{e}_{i_k},$$

where the coordinates are defined by

$$S^{i_1 \dots i_k}(\boldsymbol{x}) = \underset{0<t_1<\dots<t_k<T}{\int \dots \int} dx^{i_1}(t_1)\dots dx^{i_k}(t_k).$$

It is well known that any continuous function on a compact subset of $\mathbb{R}^d$ can be uniformly well approximated by polynomials (Conway, 2019, Thm. 8.1). In full analogy—as explained in more detail in Sec. 2.1—the collection of iterated integrals defined by the signature provides a basis for continuous functions on compact sets of paths as stated in the following result.

Theorem 4.2.1. *Let $U \subset \widetilde{\mathcal{X}}(\mathbb{R}^d)$ be a compact set of paths*[1] *with $d > 1$, and consider a continuous function $f : U \to \mathbb{R}$. Then for any $\epsilon > 0$, there exists a truncation level $n \geq 0$, such that for any path $\boldsymbol{x} \in U$*

$$\left| f(\boldsymbol{x}) - \theta_0 - \sum_{k=1}^{n} \sum_{i_1 \dots i_k} \theta_{i_1 \dots i_k} S^{i_1 \dots i_k}(\boldsymbol{x}) \right| < \epsilon,$$

where θ_0 and $\theta_{i_1 \dots i_k}$ are scalar coefficients, and the inner sum is taken over $i_1, \dots, i_k \in \{1, \dots, d\}$.

4.2.4 Truncating the signature

In light of the above, in view of numerical applications (Arribas et al., 2018; Bonnier et al., 2019; Graham, 2013; Kalsi et al., 2020; Moore et al., 2019), the signature $S(\boldsymbol{x})$ of a path $\boldsymbol{x}$ evolving in a d-dimensional vector space, might need to be truncated at a certain level $n \in \mathbb{N}_*$ which yields the approximation

$$\operatorname{sig}(\boldsymbol{x}) = \operatorname{vec}((1, S_1(\boldsymbol{x}), \dots, S_n(\boldsymbol{x}))).$$

This approximation is given by the collection of the first $(d^{n+1} - 1)/(d - 1)$ iterated integrals in eq. (4.1). Nonetheless, the resulting approximation is reasonable owing to Lyons et al. (2007, Proposition 2.2) which states that the absolute value of all neglected terms decays factorially as $|S^{i_1 \cdots i_n}(\boldsymbol{x})| = \mathcal{O}\left(\frac{1}{n!}\right)$. This factorial decay ensures that when the signature of $\boldsymbol{x}$ is truncated, only a negligible amount of information about $\boldsymbol{x}$ is lost (Bonnier et al., 2019, Sec. 1.3).

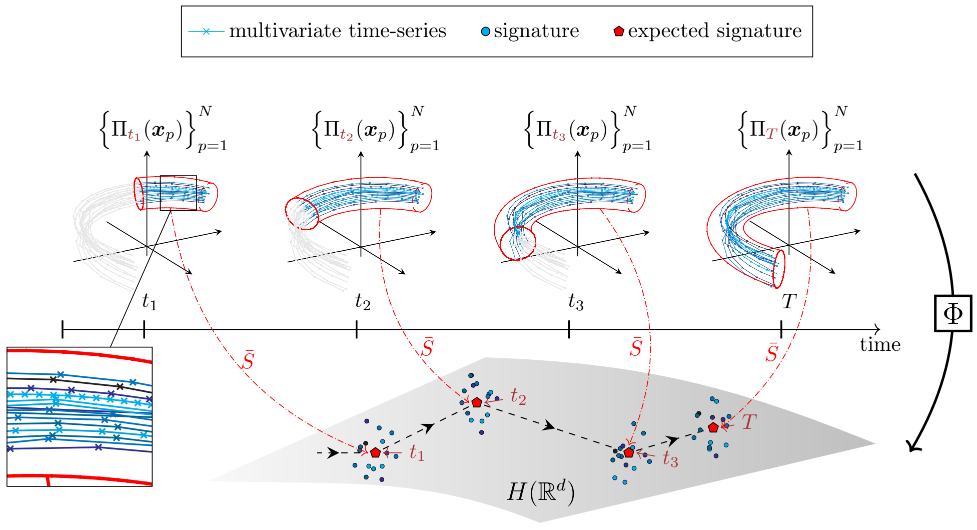
4.2.5 Robustness to irregular sampling

The invariance of the signature to a special class of transformations in the time domain of a path (Friz and Victoir, 2010, Proposition 7.10) called time reparameterizations, such as shifting $t \mapsto t + b$ and acceleration $t \mapsto t^b$ $(b > 0)$, partially explains its effectiveness to deal with irregularly sampled data streams

[1] $\widetilde{\mathcal{X}}(\mathbb{R}^d)$ is defined in terms of $\mathcal{X}(\mathbb{R}^{d-1})$ in Sec. 2.1.2

(Bonnier et al., 2019; Chevyrev and Kormilitzin, 2016). In effect, the iterated integrals in eq. (4.1) disregard the time parameterization of a path $\boldsymbol{x}$, but focus on describing its shape. To retain the information carried by time, it suffices to augment the state space of $\boldsymbol{x}$ by adding time t as an additional dimension, yielding $\tilde{\boldsymbol{x}} : t \mapsto (t, x^1(t), \ldots, x^{d-1}(t))$. This augmentation becomes particularly useful in the case of univariate time-series where the action of the signature becomes somewhat trivial, as there are no interesting dependencies to capture between the different path-coordinates (Chevyrev and Kormilitzin, 2016, Example 5). Furthermore, we use this augmentation in Thm. 4.2.1 in order to ensure the injectivity of the signature (as explained in Sec. 2.1.2).

4.3 Methodology



multivariate time-series
signature
expected signature
$\left\{\Pi_{t_1}(\boldsymbol{x}_p)\right\}_{p=1}^{N}$
$\left\{\Pi_{t_2}(\boldsymbol{x}_p)\right\}_{p=1}^{N}$
$\left\{\Pi_{t_3}(\boldsymbol{x}_p)\right\}_{p=1}^{N}$
$\left\{\Pi_{T}(\boldsymbol{x}_p)\right\}_{p=1}^{N}$
t_1
t_2
t_3
T
time
Φ
$\bar{S}$
$\bar{S}$
$\bar{S}$
$\bar{S}$
t_1
t_2
t_3
T
$H(\mathbb{R}^d)$

Figure 4.2: Schematic overview of the action of *pathwise expected signature* Φ on a set of paths $\{\boldsymbol{x}_p\}_{p=1}^{N}$. *Top*: Representation of the information about the set 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 $H(\mathbb{R}^d)$.

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

$$\left\{ \left(\{\boldsymbol{x}_{i,p}\}_{p=1}^{N_i}, y_i \right) \;\middle|\; \boldsymbol{x}_{i,p} \in \mathcal{X}(\mathbb{R}^d),\ y_i \in \mathbb{R} \right\}_{i=1}^{M}, \tag{4.2}$$

such that the finite set of paths $U = \bigcup_{i=1}^{M} \{\boldsymbol{x}_{i,p}\}_{p=1}^{N_i}$ is a compact subset of $\mathcal{X}(\mathbb{R}^d)$. As mentioned in Sec. 4.1.1, we can summarize the information carried by the collection of paths $\{\boldsymbol{x}_{i,p}\}_{p=1}^{N_i}$ in set i by considering the empirical measure $\mathbb{P}_i = \frac{1}{N_i} \sum_{p=1}^{N_i} \delta_{\boldsymbol{x}_{i,p}} \in \mathcal{P}(U)$, where $\delta_{\boldsymbol{x}_{i,p}}$ is the Dirac measure centered at $\boldsymbol{x}_{i,p}$.

This way, the input-output pairs in eq. (4.2) can be represented by

$$\left\{(\mathbb{P}_i, y_i) \;\middle|\; \mathbb{P}_i \in \mathcal{P}(U),\ y_i \in \mathbb{R}\right\}_{i=1}^{M}. \tag{4.3}$$

The sequence of moments $(\mathbb{E}[Z^{\otimes m}])_{m\geq 0}$ is classically known to characterize the law $\mathbb{P}_Z$ 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 (Chevyrev and Oberhauser, 2018). It says that one can fully characterize a probability measure on paths (provided that 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 15. *The expected signature is the map $\bar{S} : \mathcal{P}(U) \to H(E)$ defined for any $\mathbb{P} \in \mathcal{P}(U)$ by*

$$\bar{S}(\mathbb{P}) = \int_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x})\mathbb{P}(d\boldsymbol{x}).$$

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.

Theorem 4.3.1. *The expected signature map is injective and weakly continuous.*

Proof. The injectivity has been proved in Chevyrev and Oberhauser (2018, Thm. 5.3). We prove the weak continuity in Appendix B.1.1. □

4.3.1 A feature-based approach (SES)

As stated in Thm. 4.2.1, linear combinations of coordinate-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 on signatures. For any $t \in [0, T]$ consider the projection Π_t that maps any path $\boldsymbol{x}$ to its restriction to the subinterval $[0, t] \subset [0, T]$, such that $\Pi_t(\boldsymbol{x}) = \boldsymbol{x}|_{[0,t]}$ (see Fig. 4.2).

Definition 16 (Pathwise expected signature)**.** *The pathwise expected signature is the function $\Phi : \mathcal{P}(U) \to \mathcal{X}(H(E))$ that to a probability measure $\mathbb{P} \in \mathcal{P}(U)$ associates the path $\Phi(\mathbb{P}) : [0, T] \to H(E)$ defined for all $t \in [0, T]$ by*

$$\Phi(\mathbb{P})(t) = \mathbb{E}_{X\sim\mathbb{P}}\left[S\left(\Pi_t(X)\right)\right].$$

The action of Φ is illustrated on Fig. 4.2, and its implementation is outlined in Alg. 5.[2] In line 8 of Alg. 5, we use an algebraic property for fast computation of the signature, known as Chen's theorem (see Thm. 2.1.5 and Alg. 1).

Algorithm 5 PES — Computing the pathwise expected signature

```
1: Input: $N$ streams $\{\underline{\mathbf{x}}_p\}_{p=1}^N$ each of length $\ell$
2: Create array $\Phi$ to store the PES
3: Create array $S$ to store the signatures
4: Initialize $S[p] \leftarrow \mathbf{1}$ for $p \in \{1,\dots,N\}$
5: Initialize $\Phi[1] \leftarrow \mathbf{1}$
6: for each time-step $k \in \{2,\dots,\ell\}$ do
7:   for each stream $p \in \{1,\dots,N\}$ do
8:     // Compute the signature via Chen's relation
9:     $S[p] \leftarrow S[p] \otimes \exp\big(\underline{\mathbf{x}}_p[k] - \underline{\mathbf{x}}_p[k-1]\big)$
10:  end for
11:  $\Phi[k] \leftarrow \mathrm{avg}(S)$
12: end for
13: Output: The pathwise expected signature $\Phi$
```

We denote by $\Phi_k(\mathbb{P})$, the projection on $E^{\otimes k}$ of the pathwise expected signature of a probability measure $\mathbb{P}$, that is the path $\Phi_k(\mathbb{P}) : [0,T] \to E^{\otimes k}$, defined for all $t \in [0,T]$ by $\Phi_k(\mathbb{P})(t) = \mathbb{E}_{X\sim\mathbb{P}}\left[S_k\left(\Pi_t(X)\right)\right]$. In the sequel, $\widetilde{\Phi}$ denotes the pathwise expected signature augmented with time, such that $\widetilde{\Phi}(\mathbb{P})(t) = (t, \Phi(\mathbb{P})(t))$ for all $t \in [0,T]$.

The next theorem states that any weakly continuous function on $\mathcal{P}(U)$ can be uniformly well approximated by a linear combination of terms in the signature of the pathwise expected signature.

Theorem 4.3.2. *Let $U \subset \widetilde{\mathcal{X}}(\mathbb{R}\oplus E)$ be a compact set of paths, where E is a vector space of dimension $d \in \mathbb{N}_*$, and consider a weakly continuous function $F : \mathcal{P}(U) \to \mathbb{R}$. Then for any $\epsilon > 0$ there exists a truncation level $m \geq 0$, such that for any probability measure $\mathbb{P} \in \mathcal{P}(U)$,*

$$\left| F(\mathbb{P}) - \sum_{k=0}^{m} \mathcal{L}_k \circ S(\widetilde{\Phi}(\mathbb{P})) \right| < \epsilon$$

where $\mathcal{L}_k$ are linear functionals, that is, elements of the dual space of $(\mathbb{R}\oplus H(\mathbb{R}\oplus E))^{\otimes k}$, naturally extended to elements of the dual space of $H(\mathbb{R}\oplus H(\mathbb{R}\oplus E))$.

[2]Equivalently $\Phi(\mathbb{P})(t) = \bar{S}(\Pi_t\#\mathbb{P})$ where $\Pi_t\#\mathbb{P}$ is the push-forward measure of $\mathbb{P}$ by the measurable map Π_t.

Proof. $\mathcal{P}(U)$ is compact (see the proof of Thm. 4.3.3) and the image of a compact set by a continuous function is compact. Therefore, the image $U_\Phi = \Phi(\mathcal{P}(U))$ is a compact subset of $\mathcal{X}(H(\mathbb{R} \oplus E))$. Consider a weakly continuous function $F : \mathcal{P}(U) \to \mathbb{R}$. Given that Φ is injective (see Appendix B.1.2), Φ is a bijection when restricted to its image U_Φ. Therefore, there exists a continuous function $f : U_\Phi \to \mathbb{R}$ (w.r.t. $\|\cdot\|$) such that $F = f \circ \Phi$. Denote by τ the map that augments a path with a time coordinate. By Thm. 2.1.4 we know that for any $\epsilon > 0$, there exists a linear functional $\mathcal{L} : H(\mathbb{R} \oplus H(\mathbb{R} \oplus E)) \to \mathbb{R}$ of the form $\mathcal{L} = \sum_{k=0}^m \mathcal{L}_k$ such that $\|f - \mathcal{L} \circ S \circ \tau\|_\infty < \epsilon$. Thus, $\|F \circ \Phi^{-1} - \mathcal{L} \circ S \circ \tau\|_\infty < \epsilon$, implying $\|F - \mathcal{L} \circ S \circ \tau \circ \Phi\|_\infty < \epsilon$, that is, $\|F - \mathcal{L} \circ S \circ \widetilde{\Phi}\|_\infty < \epsilon$. $\square$

The practical consequence of this theorem is that the complex task of learning a highly non-linear function $F : \mathcal{P}(U) \to \mathbb{R}$ can be reformulated as a linear regression on the signature (truncated at level m) of the pathwise expected signature (truncated at level n). The model for the regression function takes the form

$$\widehat{F}(\mathbb{P}) = \theta_0 + \sum_{k=1}^{m} \sum_{i_1 \ldots i_k} \theta_{i_1 \ldots i_k} S^{i_1 \ldots i_k}(t \mapsto \mathrm{vec}(t, 1, \Phi_1(\mathbb{P})(t), \ldots, \Phi_n(\mathbb{P})(t))),$$

where $i_1, \ldots, i_k \in \{1, \ldots, (\tilde{d}^{n+1} - 1)/(\tilde{d} - 1)\}$ with $\tilde{d} = d + 1$. The resulting SES algorithm is outlined in Alg. 6, and has time complexity $\mathcal{O}(M\ell d^n(N + d^m))$, where M is the total number of sets, 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 set. The factorial decay mentioned in Sec. 4.2.4, also applies to the terms of the (pathwise) expected signature, hence low truncation levels $n, m \in \{2, 3\}$ will usually be sufficient in practice to achieve good predictive performances.

4.3.2 A kernel-based approach (KES)

The SES algorithm is well suited to datasets containing a possibly large number $M \times N$ 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 $K_{\mathrm{dr}} : \mathcal{P}(U) \times \mathcal{P}(U) \to \mathbb{R}$ by combining the expected signature with a Gaussian kernel and prove its universality to approximate weakly continuous functions on probability measures on paths. The resulting kernel-based algorithm (KES) for DR on sequential data is well adapted to the opposite data regime, i.e. when the dataset consists of few number $M \times N$ of high dimensional paths.

Algorithm 6 SES — DR on sequential data with SES

```
1: **Input:** $\{(\mathscr{S}_i, y_i)\}_{i=1}^M$ where $\mathscr{S}_i = \{\underline{\mathbf{x}}_{i,p}\}_{p=1}^{N_i}$
2: Create array $A$ to store $M$ signatures of the PES.
3: Initialize $A[:,i] \leftarrow \mathbf{1}$ for $i \in \{1,\dots,M\}$
4: for each set $i \in \{1,...,M\}$ do
5:     // Compute the pathwise expected signature using Alg. 5
6:     $\Phi = \mathsf{PES}(\mathscr{S}_i)$
7:     // Add a time coordinate to the PES
8:     Create array $\widetilde{\Phi}$
9:     $\widetilde{\Phi}[k] \leftarrow (k, \Phi[k])$ for $k \in \{1,\dots,\ell_i\}$
10:    for each time-step $k \in \{2,\dots,\ell_i\}$ do
11:        // Compute the signature of the (time-augmented) PES
12:        $A[:,i] \leftarrow A[:,i] \otimes \exp(\widetilde{\Phi}[k] - \widetilde{\Phi}[k-1])$
13:    end for
14: end for
15: $(\theta_0,\dots,\theta_c) = \text{LinearRegression}(A, (y_i)_{i=1}^M)$
16: **Output:** Regression coefficients $(\theta_0,\dots,\theta_c)$
```

Theorem 4.3.3. *Let $U \subset \widetilde{\mathcal{X}}(\mathbb{R} \oplus E)$ be a compact set of paths and $\sigma > 0$. The kernel $K_{dr} : \mathcal{P}(U) \times \mathcal{P}(U) \to \mathbb{R}$ defined by,*

$$K_{dr}(\mathbb{P}, \mathbb{Q}) = \exp\Big(-\sigma^2 \left\|\bar{S}(\mathbb{P}) - \bar{S}(\mathbb{Q})\right\|^2_{H(\mathbb{R}\oplus E)}\Big) \tag{4.4}$$

is universal, i.e. the associated RKHS is dense in the space of continuous functions from $\mathcal{P}(U)$ to $\mathbb{R}$.

Proof. By Christmann and Steinwart (2010, 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 $\|\cdot\|$, U is a compact metric space. Hence the set $\mathcal{P}(U)$ is weakly-compact (Walkden, 2014, Thm. 10.2). Given that (U, d_U)—where d_U is the topology induced by $\|\cdot\|$—is a compact metric space, the topology describing weak convergence of (Borel) probability measures can be metrized (e.g. by the Prohorov metric $d_{\mathcal{P}(U)}$). Therefore $(\mathcal{P}(U), d_{\mathcal{P}(U)})$ is also a compact metric space. By Thm. 4.3.1, the *expected signature* $\bar{S} : \mathcal{P}(U) \to H(\mathbb{R} \oplus E)$ is injective and weakly continuous. Furthermore, $H(\mathbb{R}\oplus E)$ is a Hilbert space with a countable basis, hence it is separable. Setting $K = \mathcal{P}(U)$, $H = H(\mathbb{R} \oplus E)$ and $\rho = \bar{S}$ concludes the proof. □

We note that Thm. 4.3.3 holds more generally for any Taylor-type kernel of the form,

$$K_{\mathrm{dr}}(\mathbb{P}, \mathbb{Q}) = \sum_{n=0}^{\infty} a_n ||\bar{S}(\mathbb{P}) - \bar{S}(\mathbb{Q})||^{2n}, \quad a_n > 0 \tag{4.5}$$

including the Gaussian-type kernel in eq. (4.4).

4.3.3 Evaluating the distribution regression kernel

When the input measures are two empirical measures $\mathbb{P}_1 = \frac{1}{N_1}\sum_{p=1}^{N_1} \delta_{\boldsymbol{x}_{1,p}}$ and $\mathbb{P}_2 = \frac{1}{N_2}\sum_{q=1}^{N_2} \delta_{\boldsymbol{x}_{2,q}}$, the evaluation of the kernel K_{dr} in Equation (4.4) requires the ability to compute the tensor norm

$$\left\|\bar{S}(\mathbb{P}_1) - \bar{S}(\mathbb{P}_2)\right\|^2 = m_{11} + m_{22} - 2 \times m_{12}, \tag{4.6}$$

where for $i, j \in \{1, 2\}$, the term m_{ij} is given by

$$m_{ij} = \frac{1}{N_i N_j} \sum_{p=1}^{N_i} \sum_{q=1}^{N_j} \big\langle S(\boldsymbol{x}_{i,p}), S(\boldsymbol{x}_{j,q}) \big\rangle.$$

Each of these inner products defines another recent object from stochastic analysis called the signature kernel K (Király and Oberhauser, 2019). Recently, Cass et al. (2020) have shown that K is actually the solution of a surprisingly simple partial differential equation (PDE). This result provides us with a kernel trick for computing the inner products in Equation (4.6) by a simple call to any numerical PDE solver.

Theorem 4.3.4. *(Cass et al., 2020, Thm. 2.2) The signature kernel*

$$K(\boldsymbol{x}, \boldsymbol{y}) := \langle S(\boldsymbol{x}), S(\boldsymbol{y}) \rangle, \tag{4.7}$$

is the solution $u : [0, T] \times [0, T] \to \mathbb{R}$ at $(s, t) = (T, T)$ of the following linear hyperbolic PDE

$$\frac{\partial^2 u}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle \, u \tag{4.8}$$

with boundary conditions $u(0, \cdot) = 1$ and $u(\cdot, 0) = 1$.

In light of Thm. 4.3.3, DR on paths with KES can be performed using any kernel method (Drucker et al., 1997; Quiñonero-Candela and Rasmussen, 2005) available within popular libraries (De G. Matthews et al., 2017; Gardner et al., 2018; Pedregosa et al., 2011) using the Gram matrix computed using Alg. 7 and leveraging the aforementioned kernel trick. When using a finite difference scheme (referred to as PDESolve in Alg. 7) to approximate the solution of the PDE, the resulting time complexity of KES is $\mathcal{O}(M^3 + M^2N^2\ell^2 d)$.

Algorithm 7 GramKES — Gram matrix for KES

```
1: **Input:** $\{\boldsymbol{x}_{i,p}\}_{p=1}^{N_i}$, for $i = 1, \dots, M$ and $\sigma > 0$.
2: Initialize 0-array $G \in \mathbb{R}^{M \times M}$
3: **for** each pair of sets $(i, j)$ such that $i \leq j$ **do**
4:     Initialize 0-array $K_{ij} \in \mathbb{R}^{N_i \times N_j}$
5:     Similarly initialize $K_{ii}, K_{jj} \in \mathbb{R}^{N_i \times N_i}, \mathbb{R}^{N_j \times N_j}$
6:     **for** $p, p'$ in set $i$ and $q, q'$ in set $j$ **do**
7:         $K_{ii}[p, p'] \leftarrow \mathsf{PDESolve}(\boldsymbol{x}_{i,p}, \boldsymbol{x}_{i,p'})$
8:         $K_{jj}[q, q'] \leftarrow \mathsf{PDESolve}(\boldsymbol{x}_{j,q}, \boldsymbol{x}_{j,q'})$
9:         $K_{ij}[p, q] \leftarrow \mathsf{PDESolve}(\boldsymbol{x}_{i,p}, \boldsymbol{x}_{j,q})$
10:     **end for**
11:     $G[i, j] \leftarrow \text{avg}(K_{ii}) + \text{avg}(K_{jj}) - 2 \times \text{avg}(K_{ij})$
12:     $G[j, i] \leftarrow G[i, j]$
13: **end for**
14: $G \leftarrow \exp(-\sigma^2 G)$                    // *elementwise exponential*
15: **Output:** The gram matrix $G$.
```

Remark In the case where the observed paths are assumed to be i.i.d. samples $\{X_p\}_{p=1}^N \sim \mathbb{P}$ from the law of an underlying random process, one would expect that the larger the sample size N, the better the approximation of $\mathbb{P}$, and therefore of its expected signature $\bar{S}(\mathbb{P})$. Indeed, for an arbitrary multi-index $(i_1, \dots, i_k)$, the *Central Limit Theorem* yields the convergence (in distribution)

$$\sqrt{N}\Big(\mathbb{E}_{X\sim\mathbb{P}}\big[S^{i_1\dots i_k}(X)\big] - \frac{1}{N}\sum_{p=1}^{N} S^{i_1\dots i_k}(X_p)\Big) \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, \sigma^2_{i_1\dots i_k}\right),$$

as the variance $\sigma^2_{i_1\dots i_k} = \mathbb{E}_{X\sim\mathbb{P}}\big[S^{i_1\dots i_k}(X)^2\big] - \big(\mathbb{E}_{X\sim\mathbb{P}}\big[S^{i_1\dots i_k}(X)\big]\big)^2$ is always finite; in effect, the product $S^{i_1\dots i_k}(X)S^{i_1\dots i_k}(X)$ can be expressed as a finite sum of higher-order terms $S^{j_1\dots j_{2k}}(X)$ according to the shuffle identity (Thm. 2.1.2). However, we note that Monte Carlo sampling is only one way of estimating the expected signature. There are stochastic processes, such as Brownian motion, for which the expected signature can be computed by solving a PDE (Ni, 2012).

4.4 Related work

Recently, there has been a growing interest in extending regression algorithms to the case where inputs are sets of numerical arrays (Hamelijnck et al., 2019; Law et al., 2018a; Musicant et al., 2007; Skianis et al., 2020; Wagstaff et al., 2008). Here we highlight the previous work most closely related to our approach.

Deep learning techniques DeepSets (Zaheer et al., 2017) are examples of neural networks designed to process each element 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. The aggregation makes it possible to encode permutation invariance. However, these models depend on a large number of parameters and the results may vary greatly depending on the choice of architecture and activation functions (Wagstaff et al., 2019)

Kernel-based techniques In the setting of DR, elements of a set are viewed as samples from an underlying probability distribution (Flaxman, 2015; Law et al., 2018b; Muandet et al., 2012; Smola et al., 2007; Szabó et al., 2016). This framework can be intuitively summarized as a two-step procedure. Firstly, a probability measure μ is mapped to a point in an RKHS $\mathcal{H}_k$ by means of a *kernel mean embedding* $\mu = \int_{x\in\mathcal{X}} k(\cdot, x)\mathbb{P}(dx)$, where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the associated reproducing kernel. Secondly, the regression is finalized by approximating a function $F : \mathcal{H}_k \to \mathbb{R}$ by minimizing $F \approx \arg\min_{g\in\mathcal{H}_\kappa} \sum_{i=1}^{M} c(y_i, g(\mu_i))$, where c is a loss function, resulting in a procedure involving a second kernel $\kappa : \mathcal{H}_k \times \mathcal{H}_k \to \mathbb{R}$. In Sec. 4.5 we denote by DR-k the models produced by choosing κ as a Gaussian-type kernel. Despite the theoretical guarantees of these methods (Szabó et al., 2016), the feature map $k(\cdot, x)$ 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.

The signature method As explained in Sec. 2.1, the signature method consists in using the terms of the signature as features to solve supervised learning problems on time-series, with successful applications, such as bipolar disorder detection (Arribas et al., 2018) or human action recognition (Yang et al., 2017) to name a few. The signature features have also been used to construct neural network layers (Bonnier et al., 2019; Graham, 2013) in deep architectures. The feature-based method (SES) developed in this chapter can be viewed as a generalization of the signature method when inputs are *sets* of multivariate time-series.

4.5 Experiments

We benchmark our feature-based (SES) and kernel-based (KES) methods against DeepSets and the existing kernel-based DR techniques discussed in Sec. 4.4 on various simulated and real-world examples from physics, mathematical finance and agricultural science. With these examples, we show the ability of our methods to handle challenging situations where only a few number of labeled sets of multivariate time-series are available. We consider the kernel-based techniques DR-k with $k \in \{\mathrm{RBF}, \mathrm{Matern32}, \mathrm{GA}\}$, where GA refers to the Global Alignment kernel for time-series from Cuturi et al. (2007). Unlike our methods, DeepSets, DR-RBF and DR-Matern32 are all for static arrays on $\mathbb{R}^d$. This is why we also construct the DR-GA method, which can be seen as a simplification of KES, where some smaller terms are deleted in the signature (see Király and Oberhauser (2019, Sec 5.)).

For KES and DR-k we perform Kernel Ridge Regression, while 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 are selected by cross-validation via a grid search on the training set of each run. Additional details about hyperparameters search and model architecture can be found in Appendix B.2. The code is available at `https://github.com/maudl3116/Distribution_Regression_Streams`.

4.5.1 A defective electronic device

We start with a toy example to show the robustness of our methods to irregularly sampled time-series. To this aim, we propose to infer the phase φ of an electronic

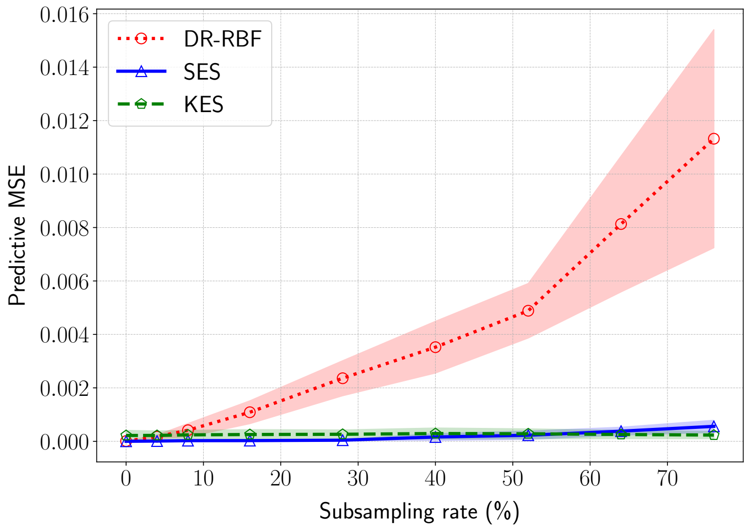


Figure 4.3: Predictive MSE at various subsampling rates for $M = 50$ circuits and $N = 15$ devices. The shaded area indicates the standard deviation.

circuit from multiple recordings of its voltage $v^{\varphi}(t) = \sin(\omega t)$ and current $i^{\varphi}(t) = \sin(\omega t - \varphi)$. The data consist 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 Fig. 4.3, the predictive performances of DR-RBF drastically deteriorate when the subsampling rate increases, whilst results for KES and SES are stable.

4.5.2 Inferring the temperature of an ideal gas

Table 4.1: Ideal gas dataset. Radii of the particles in the 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).

Model	Predictive MSE $\times 10^{-2}$ (standard deviation)	
	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)

The thermodynamic properties of an *ideal gas* of N particles inside a 3-d box of volume V (3 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 $U = c_VNk_BT$, where k_B is the *Boltzmann constant* (Adkins and Adkins, 1983), 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 $U = \frac{1}{2}\sum_{p=1}^{N} m_p|\overrightarrow{v_p}|^2$. The complexity of the large-scale dynamics of the gas depends on T (see Fig. 4.1) and 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, randomly initializing all velocities and letting the particles evolve at constant speed.[3] 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 Table 4.1 we report the results of two experiments, one where the particles have a small radius (few collisions) and another where they

[3]We assume (Chang, 2015) that the environment is frictionless, and that the particles are not subject to other forces such as gravity. We use the Python code from `https://github.com/labay11/ideal-gas-simulation`.

have a larger radius (many collisions). The performance of DR-k is comparable to the ones 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, whereas the performance of DR-k drops with increasing the system complexity. With a total number of $MN = 1\,000$ time-series of dimension $d = 7$ (after path augmentation discussed in Sec. 4.2.5 and in Appendix B.2), KES runs in 50 seconds, three times faster than SES on a 128 cores CPU.

4.5.3 Parameter estimation in a pricing 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 (Arribas et al., 2020). This setting is often too simple to match the volatility observed in the market, especially since the advent of electronic trading (Gatheral et al., 2018). 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 regularity of the trajectories (Decreusefond et al., 1999).[4] In line with the findings in Gatheral et al. (2018) 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 $[10^{-6}, 1]$. 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. As shown in Table 4.2, KES and SES systematically yield the best MSE among all the models compared. Furthermore, the performance of KES and SES progressively improves with the number of time-series in each set, according to the remark at the end of Sec. 4.3, while this pattern is not observed for DR-RBF, DR-Matern32, and DeepSets. Both KES and SES give comparable results. However, while the running time of SES remains stable (≈ 1 min) when $M \times N$ increases from $1\,000$ to $5\,000$, the running time of KES increases from ≈ 1 min to 15 min (on 128 cores).

4.5.4 Crop yield prediction from GLDAS data

Finally, we evaluate our methods on a crop yield prediction task. The challenge consists in predicting the yield of wheat crops in a region from longitudinal measurements of climatic variables recorded at different locations in the region.

[4]We note that sample-paths of fBM are not in $\mathscr{C}([0, T], \mathbb{R})$ but we can assume that the interpolations obtained from market high-frequency data provide a sufficiently refined approximation of the underlying process.

Table 4.2: Predictive MSE (standard deviation) on the rough volatility dataset. N is the number of rough volatility trajectories and $(M, d, \ell) = (50, 2, 200)$.

Model	Predictive MSE $\times 10^{-3}$ (standard deviation)		
	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)

We use the publicly available Eurostat dataset (available at `http://ec.europa.eu/eurostat/data/database`) 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 (Rodell et al., 2004), are recorded every 6 hours at a spatial resolution of 0.25° × 0.25°, and their number varies across regions. We further subsample at random 50% of the measurements. As can be seen in Table 4.3, SES and KES are the two methods that improve the most against the baseline which consists in predicting the average yield on the train set.

Table 4.3: MSE and MAPE (mean absolute percentage error) on the Eurostat/GLDAS dataset

Model	MSE (std)	MAPE (std)
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)

4.6 Conclusion

In this chapter, we have developed two techniques for DR on sequential data. In the first technique, we have introduced the pathwise expected signature and have constructed a universal feature map for probability measures on paths. In the second technique, we have defined a universal kernel based on the expected signature. We have shown the robustness of the proposed methodologies to irregularly sampled multivariate time-series, achieving state-of-the-art performances on various DR problems for sequential data.

Chapter 5

Kernel Mean Embeddings for Stochastic Processes

In the previous chapter, we have constructed a feature map and a kernel for representing and learning on (discrete or empirical) probability measures supported on some space of paths, representing laws of stochastic processes. This allowed us to perform regression analysis on sets of multivariate time-series.

However, reducing a stochastic process to a path–valued random variable ignores its *filtration*, that is, the flow of information carried by the process through time. In this chapter, by conditioning the process on its filtration, we introduce a family of *higher order kernel mean embeddings* (KMEs) that generalizes the notion of KME and captures additional information related to the filtration. We derive empirical estimators for the associated *higher order maximum mean discrepancies* (MMDs) and prove their consistency. We then construct a filtration-sensitive kernel two-sample test able to pick up information that gets missed by the standard MMD test. In addition, leveraging the higher order MMDs we construct a family of universal kernels on stochastic processes that allows one to solve real-world calibration and optimal stopping problems in quantitative finance (such as the pricing of American options) via classical kernel-based regression methods. Finally, adapting existing tests for conditional independence to the case of stochastic processes, we design a causal discovery algorithm to recover the causal graph of structural dependencies among interacting bodies solely from observations of their multidimensional trajectories.

5.1 Introduction

The idea of embedding probability distributions into a reproducing kernel Hilbert space (RKHS) via kernel mean embeddings (KMEs) has become ubiquitous in many areas of statistics and data science such as hypothesis testing (Gretton et al., 2012a; Zhang et al., 2012), non-linear regression (Hofmann et al., 2008; Schölkopf et al., 2002), distribution regression (Szabó et al., 2016) etc. Despite the strong progress in the study of KMEs, most of the examples considered in the literature tend to focus on random variables supported on some finite (possibly high) dimensional Euclidean spaces like $\mathbb{R}^d$. The study of KMEs for function-valued random variables has been largely ignored.

Stochastic processes are random variables with values in some space of paths. However, reducing a stochastic process to a path-valued random variable ignores its *filtration*, which can informally be thought of as the *flow of information carried by the process through time*. A question that naturally emerges from the study of many random, time-evolving systems like financial markets is how does the information available up to present time affect the future evolution of the system?

Formally, this question can be addressed by conditioning a process on its filtration (Sec. 5.3.1 and Sec. 5.3.2). In this chapter, we introduce a family of *higher order KMEs* that generalizes the notion of KME to capture additional, filtration-related information (Sec. 5.3.3 and Sec. 5.3.5). In view of concrete applications, we derive empirical estimators for the associated *higher order MMDs* and use one of them to construct a filtration-sensitive kernel two-sample test (Sec. 5.3.4) demonstrating with simulated data its ability to capture information that otherwise gets missed by the standard MMD test (Sec. 5.4.1). Furthermore, we construct a family of universal kernels on stochastic processes (Sec. 5.3.6) that allows us to solve challenging, real-world optimization problems in quantitative finance via classical kernel-based regression methods (Sec. 5.4.2). Finally, we adapt existing tests for conditional independence to the case of stochastic processes in order to design a causal-discovery algorithm able to recover the causal graph of structural dependencies among interacting bodies solely from observations of their multidimensional trajectories (Sec. 5.4.3).

5.1.1 Related work

The notion of conditioning is a powerful probabilistic tool allowing one to understand possibly complex, non-linear interactions between random variables. As their unconditional counterparts, conditional distributions can also be embedded into RKHSs (Song et al., 2013). Recently, conditional KMEs have received increased attention, especially in the context of graphical models (Song et al., 2010), state-space models (Fukumizu et al., 2013), dynamical systems (Song et al., 2009), causal inference (Mitrovic et al., 2018; Sun et al., 2007; Tillman et al., 2009), two-sample and conditional independence hypothesis testing (Fukumizu et al., 2007; Park and Muandet, 2020; Sun et al., 2007). Embeddings of distributions via KMEs have also shown their success in the

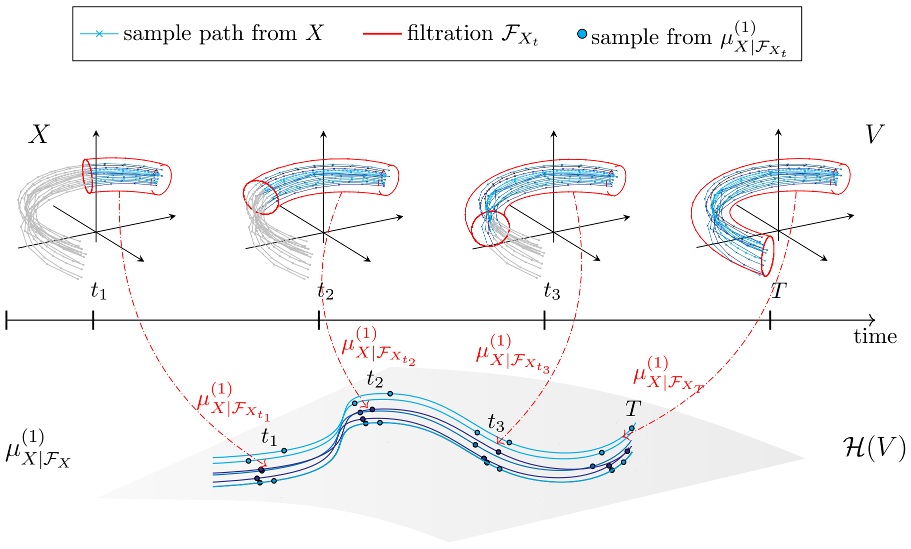

sample path from X
filtration $\mathcal{F}_{X_t}$
sample from $\mu^{(1)}_{X|\mathcal{F}_{X_t}}$
X
V
t_1
t_2
t_3
T
time
$\mu^{(1)}_{X|\mathcal{F}_{X_{t_1}}}$
$\mu^{(1)}_{X|\mathcal{F}_{X_{t_2}}}$
$\mu^{(1)}_{X|\mathcal{F}_{X_{t_3}}}$
$\mu^{(1)}_{X|\mathcal{F}_{X_T}}$
$\mu^{(1)}_{X|\mathcal{F}_X}$
$\mathcal{H}(V)$

Figure 5.1: Schematic overview of the construction in (Sec. 5.3.2) of the 1st order predictive KME $\mu^{(1)}_{X|\mathcal{F}_X}$. Here X is a stochastic process with sample paths taking their values in V. The red contours indicate the portion of its filtration $\mathcal{F}_X$ upon which the conditioning is applied, i.e. the available information about X from start up to time t. As explained in Sec. 5.3.2, the 1st order predictive KME $\mu^{(1)}_{X|\mathcal{F}_X}$ is a path whose value at time t is a $\mathcal{H}(V)$-valued random variable representing the law of X conditioned on its filtration $\mathcal{F}_{X_t}$. Equivalently $\mu^{(1)}_{X|\mathcal{F}_X}$ is a stochastic process with sample paths taking their values in $\mathcal{H}(V)$.

context of distribution regression (DR), which is the task of learning a function mapping a collection of samples from a probability distribution to scalar targets (Law et al., 2018b; Muandet et al., 2012; Smola et al., 2007). In Chapter 4, we developed a framework for DR that addresses the setting in which inputs are sample paths from an underlying stochastic process. We used the *signature transform* (Bonnier et al., 2019; Lyons, 2014) and the *signature kernel* (Cass et al., 2020; Király and Oberhauser, 2019), two well-established tools in stochastic analysis.

When it comes to stochastic processes, it was first shown in Aldous (1981) that weak convergence of random variables does not always account for the information contained in the filtration, as highlighted by means of numerous numerical examples in Backhoff-Veraguas et al. (2021); Pflug and Pichler (2012). This limitation is addressed in Aldous (1981); Hoover and Keisler (1984) through the construction of a sequence of so–called *adapted topologies*[1] $(\tau_n)_{n\geq 1}$ that become progressively finer[2] as n increases (with τ_1 coinciding with the weak topology). In particular, higher order adapted topologies are shown to

[1]We say that a sequence of random variables $\{X_n\}_{n\in\mathbb{N}}$ converges to a random variable X in the topology τ if and only if for every τ-open neighbourhood $\mathbb{U}$ of X there exists $N \in \mathbb{N}$ such that $X_n \in \mathbb{U}$ as soon as $n \geq N$.

[2]A topology τ_1 is said to be finer than a topology τ_2 if every τ_2-open set is also τ_1-open.

capture more filtration-related information than their weak counterpart. This characteristic becomes relevant for example in some optimal stopping problems such as the pricing of American options, where the pricing function can be shown to be discontinuous with respect to the weak topology, but is continuous with respect to the second order adapted topology[3] (Backhoff-Veraguas et al., 2019, 2020; Pflug and Pichler, 2012). Leveraging properties of the signature transform, it has been shown that adapted topologies are intimately related to a family of higher order MMDs (Bonnier et al., 2020). However, providing empirical estimators for these discrepancies that can be deployed on real-world tasks remains a challenge. In this chapter, we propose to address this challenge by presenting an alternative construction to this higher order MMDs using the formalism of kernels and KMEs. The results in Bonnier et al. (2020) serve as a strong theoretical background for this chapter.

5.2 Preliminaries

We begin with a brief summary of tools from stochastic analysis needed to define higher order KMEs. Let $\mathcal{X}(\mathbb{R}^d) = \{\boldsymbol{x} : [0,T] \to \mathbb{R}^d\}$ be a compact set of continuous, piecewise linear, $\mathbb{R}^d$-valued paths defined over a common time interval $[0,T]$, obtained for example by linearly interpolating a multivariate time-series. More generally we denote by $\mathcal{X}(V) = \{\boldsymbol{x} : [0,T] \to V\}$ a compact set of continuous, piecewise linear paths with values in a Hilbert space V with a countable basis.

5.2.1 The signature transform and the signature kernel

The *signature transform* $S : \mathcal{X}(V) \to H(V)$ is a *feature map* which associates to any path $\boldsymbol{x} \in \mathcal{X}(V)$ a sequence of tensors in the Hilbert space $H(V)$. The *signature kernel* $K : \mathcal{X}(V) \times \mathcal{X}(V) \to \mathbb{R}$ is a characteristic kernel defined for any pair of paths $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}(V)$ as the following inner product in $H(V)$

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle S(\boldsymbol{x}), S(\boldsymbol{y}) \rangle . \tag{5.1}$$

Theorem 5.2.1. *(Cass et al., 2020, Thm. 2.5) For any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}(V)$ the signature kernel satisfies the equation $K(\boldsymbol{x}, \boldsymbol{y}) = u(T,T)$, where $u : [0,T] \times [0,T] \to \mathbb{R}$ is the solution of the hyperbolic PDE*

$$\frac{\partial^2 u}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle_V \, u, \tag{5.2}$$

with boundary conditions $u(0,\cdot) = u(\cdot,0) = 1$ and where $\dot{\boldsymbol{z}}(s) = \frac{d\boldsymbol{z}(r)}{dr}\big|_{r=s}$.

[3]The second order adapted topology τ_2 is equivalent to the *adapted Wasserstein distance* (Backhoff-Veraguas et al., 2019).

Hence, evaluating K at a pair of paths $(\boldsymbol{x}, \boldsymbol{y})$ is equivalent to solving the PDE in eq. (5.2); in this thesis we solve PDEs numerically via a finite difference scheme (see Appendix C.2 for additional details). In what follows, we denote by $\mathcal{H}(V)$ the RKHS associated to K.

5.2.2 Stochastic processes and filtrations

We take $(\Omega, \mathcal{F}, \mathbb{P})$ as the underlying probability space. A *(discrete time) stochastic process* X is a random variable with values on $\mathcal{X}(V)$. We denote by $\mathbb{P}_X = \mathbb{P} \circ X^{-1}$ the *law* of X. Assuming the integrability condition $\mathbb{E}_X[K(X,X)] < \infty$, the 1st *order kernel mean embedding* (KME) of X is defined as[4] the following point in $\mathcal{H}(V)$

$$\mu_X^{(1)} = \mathbb{E}_X[K(\cdot, X)] = \int_{\boldsymbol{x} \in \mathcal{X}(V)} K(\cdot, \boldsymbol{x}) \mathbb{P}_X(d\boldsymbol{x}).$$

Accordingly, given two stochastic processes X, Y, their 1st *order maximum mean discrepancy* (MMD) is the standard MMD distance with kernel K given by the following expression

$$\mathscr{D}_K^{(1)}(X,Y) = \left\| \mu_X^{(1)} - \mu_Y^{(1)} \right\|_{\mathcal{H}(V)}.$$

Because the signature kernel K is characteristic, it is a classical result (Chevyrev and Oberhauser, 2018; Gretton et al., 2012a) that the 1st order MMD is a sufficient statistics to distinguish between the laws of X and Y, in other words

$$\mathscr{D}_K^{(1)}(X,Y) = 0 \iff \mathbb{P}_X = \mathbb{P}_Y. \tag{5.3}$$

Despite the fact that stochastic processes are path-valued random variables, they encode a much richer structure compared to standard $\mathbb{R}^d$-valued random variables, that goes well beyond their laws. This additional structure is described mathematically by the concept of *filtration* of a process X, defined as the following family of σ-algebras

$$\mathcal{F}_X = (\mathcal{F}_{X_t})_{t \in [0,T]},$$

where for any $t \in [0,T]$, $\mathcal{F}_{X_t}$ is the σ-algebra generated by the variables $\{X_s\}_{s \in [0,t]}$. Note that $\mathcal{F}_X$ is totally ordered in the sense that $\mathcal{F}_{X_s} \subset \mathcal{F}_{X_t}$ for all $s < t$, which naturally explains why filtrations are good mathematical descriptions to model the flow information carried by the process X.

In the next section, we will present our main findings and introduce a family of *higher order KMEs* and corresponding *higher order MMDs* as generalizations

[4]The 1st order KME is the standard KME with the signature kernel K.

of the standard KME and MMD respectively. We will do so by conditioning stochastic processes on elements of their filtrations.

5.3 Higher order kernel mean embeddings

We begin by describing how KMEs can be extended to conditional laws of stochastic processes.

5.3.1 Conditional kernel mean embeddings for stochastic processes

Let X, Y be two stochastic processes. For a given path $\boldsymbol{x} \in \mathcal{X}(V)$, the 1^{st} *order conditional kernel mean embeddings* $\mu^{(1)}_{Y|X=\boldsymbol{x}}$ and $\mu^{(1)}_{Y|X}$ are defined as follows

$$\mu^{(1)}_{Y|X=\boldsymbol{x}} = \mathbb{E}[K(\cdot, Y) \,|\, X = \boldsymbol{x}] = \int_{\boldsymbol{y} \in \mathcal{X}(V)} K(\cdot, \boldsymbol{y}) \mathbb{P}_{Y|X=\boldsymbol{x}}(d\boldsymbol{y}), \tag{5.4}$$

$$\mu^{(1)}_{Y|X} = \mathbb{E}[K(\cdot, Y) \,|\, X] = \int_{\boldsymbol{y} \in \mathcal{X}(V)} K(\cdot, \boldsymbol{y}) \mathbb{P}_{Y|X}(d\boldsymbol{y}). \tag{5.5}$$

Note that whilst $\mu^{(1)}_{Y|X=\boldsymbol{x}}$ is a single point in $\mathcal{H}(V)$, the 1^{st} order conditional KME $\mu^{(1)}_{Y|X}$ describes a cloud of points on $\mathcal{H}(V)$. Each point in this cloud is indexed by a path $\boldsymbol{x} \in \mathcal{X}(V)$. Equivalently, $\mu^{(1)}_{Y|X}$ constitutes a $\mathbb{P}_X$-measurable, $\mathcal{H}(V)$-valued random variable.

These embeddings make it possible to extend the applications of conditional KMEs to the case where the random variables are (possibly multidimensional) stochastic processes. In particular, one can directly obtain conditional independence criteria for stochastic processes (see Appendix C.1.1), enabling to deploy standard kernel-based causal learning algorithms Sun et al. (2007), as we demonstrate in Sec. 5.4. Next we describe how in the case of stochastic processes, conditioning on filtrations is an important mathematical operation to model real-world time-evolving systems.

5.3.2 Conditioning stochastic processes on their filtrations

Financial markets are examples of complex dynamical systems that evolve under the influence of randomness. An important objective for financial practitioners is to determine how actionable information available up to present could affect the future market trajectories. The task of *conditioning on the past to describe the future* of a stochastic process X can be formulated mathematically by conditioning X on its filtration $\mathcal{F}_{X_t}$ for any time $t \in [0, T]$.

More precisely, consider the 1^{st} order KME of the conditional law $\mathbb{P}_{X|\mathcal{F}_{X_t}}$, which is defined as the following $\mathcal{F}_{X_t}$-measurable, $\mathcal{H}(V)$-valued random variable

$$\mu^{(1)}_{X|\mathcal{F}_{X_t}} = \mathbb{E}[K(\cdot, X)\,|\,X_{0,t}] = \int_{\boldsymbol{x}\in\mathcal{X}(V)} K(\cdot, \boldsymbol{x})\mathbb{P}_{X|\mathcal{F}_{X_t}}(d\boldsymbol{x}), \tag{5.6}$$

where $X_{0,t}$ denotes the stochastic process X restricted to the sub-interval $[0,t] \subset [0,T]$. By varying the time index t, we can form the following ordered collection of 1st order KMEs

$$\mu^{(1)}_{X|\mathcal{F}_X} = \left(\mu^{(1)}_{X|\mathcal{F}_{X_t}}\right)_{t\in[0,T]}, \tag{5.7}$$

that we term 1^{st} *order predictive KME* of the process X. By construction, $\mu^{(1)}_{X|\mathcal{F}_X}$ describes a path taking its values in the space of $\mathcal{H}(V)$-valued random variables, in other words it is itself a stochastic process [5] (see Fig. 5.1). Hence, the law of $\mu^{(1)}_{X|\mathcal{F}_X}$ can itself be embedded via KMEs into a "higher order RKHS" (see next section), making the full procedure iterable, as we shall discuss next.

We note that for each time t, the random variable $\mu^{(1)}_{X|\mathcal{F}_{X_t}}$ in eq. (5.6) is the Bochner integral of $K(\cdot, \boldsymbol{x})$ with respect to the probability measure $\mathbb{P}_{X|\mathcal{F}_{X_t}}$. Since we assumed that V is a compact set, the path space $\mathcal{X}(V)$ is also compact. Hence, the function $\boldsymbol{x} \mapsto K(\cdot, \boldsymbol{x})$ is continuous, the set $U = \{K(\cdot, \boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}(V)\}$ is compact as continuous image of a compact set, and therefore its Bochner integral $\mu^{(1)}_{X|\mathcal{F}_{X_t}}$ takes values in the closed convex hull of U, which is again a compact subset in the RKHS $\mathcal{H}(V)$. Consequently the path $t \mapsto \mu^{(1)}_{X|\mathcal{F}_{X_t}}$ belongs to a compact subset of $\mathcal{X}(\mathcal{H}(V))$, which satisfies the assumptions introduced in Sec. 5.2.

5.3.3 Second order kernel mean embedding and maximum mean discrepancy

The 2^{nd} *order KME* is the point in $\mathcal{H}(\mathcal{H}(V))$ defined as the KME of the 1st order predictive KME

$$\mu^{(2)}_X = \int_{\boldsymbol{x}\in\mathcal{X}(\mathcal{H}(V))} K(\cdot, x)\mathbb{P}_{\mu^{(1)}_{X|\mathcal{F}_X}}(d\boldsymbol{x}).$$

The 2^{nd} *order MMD* of X, Y is the norm of the difference in $\mathcal{H}(\mathcal{H}(V))$ of their 2nd order KMEs, that is

$$\mathscr{D}^{(2)}_K(X,Y) = \left\| \mu^{(2)}_X - \mu^{(2)}_Y \right\|_{\mathcal{H}(\mathcal{H}(V))}. \tag{5.8}$$

The next theorem states that the 2nd order MMD of two stochastic processes X, Y is a stronger discrepancy measure than the 1st order MMD.

[5] Because $\mathbb{P}_X = \mathbb{P}_{X|\mathcal{F}_{X_0}}$, all the information about the law of X is contained in just the initial point of the trajectory traced by $\mu^{(1)}_{X|\mathcal{F}_X}$ (Fig. 5.1).

Theorem 5.3.1. *Given two stochastic processes X, Y*

$$\mathscr{D}_K^{(2)}(X,Y) = 0 \iff \mathbb{P}_{X|\mathcal{F}_X} = \mathbb{P}_{Y|\mathcal{F}_Y}. \tag{5.9}$$

Furthermore

$$\mathscr{D}_K^{(2)}(X,Y) = 0 \implies \mathscr{D}_K^{(1)}(X,Y) = 0, \tag{5.10}$$

but the converse is not generally true.

Proof. All proofs are given in Appendix C.4. □

Next, we make use of Thm. 5.3.1 in the context of two-sample hypothesis testing (Chevyrev and Oberhauser, 2018; Gretton et al., 2012a) for stochastic processes. In Sec. 5.4 we will show by means of a numerical example that the 2nd order MMD is able to capture filtration-related information otherwise ignored by the 1st order MMD.

5.3.4 A filtration-sensitive kernel two-sample test

Suppose that we are given m realizations $\{\boldsymbol{x}_i\}_{i=1}^m$ from X and n realizations $\{\boldsymbol{y}_i\}_{i=1}^n$ from Y. A classical two-sample test (Gretton et al., 2012a) for X, Y tests a null-hypothesis

$$H_0 : \mathbb{P}_X = \mathbb{P}_Y \text{ against the alternative } H_A : \mathbb{P}_X \neq \mathbb{P}_Y.$$

The probability of falsely rejecting the null is called the *type I error* (and similarly the probability of falsely accepting the null is called the *type II error*). If the type I error can be bounded from above by a constant α, then we say that the test is of level α. In Chevyrev and Oberhauser (2018, Sec. 8) it is shown that rejecting the null if $\left(\widehat{\mathscr{D}}_K^{(1)}(X,Y)\right)^2 > c_\alpha$ (for some c_α that depends on m, n and α) gives a test of level α, where $\widehat{\mathscr{D}}_K^{(1)}(X,Y)$ denotes the classical unbiased estimator of the 1st order MMD (Gretton et al., 2012a). This choice of threshold is conservative and can be improved by using data-dependent bounds such as in permutation tests (we refer to the MMD testing literature for extra details (Gretton et al., 2012a; Sejdinovic et al., 2013; Sriperumbudur et al., 2010)).

However, as discussed in Sec. 5.3.3, comparing the laws $\mathbb{P}_X, \mathbb{P}_Y$ via the estimator above might be insufficient to capture filtration-related information about X, Y. To overcome this limitation we propose instead to test the null-hypothesis

$$H_0 : \mathbb{P}_{X|\mathcal{F}_X} = \mathbb{P}_{Y|\mathcal{F}_Y} \text{ against the alternative } H_A : \mathbb{P}_{X|\mathcal{F}_X} \neq \mathbb{P}_{Y|\mathcal{F}_Y}. \tag{5.11}$$

Using Thm. 5.3.1, one can immediately construct a filtration-sensitive kernel two-sample test for eq. (5.11) provided one can build an empirical estimator of the 2nd order MMD $\mathscr{D}_K^{(2)}(X,Y)$. In the rest of this section we explain how to obtain such an estimator and ultimately show its consistency.

Assuming availability of m realizations $\{\widetilde{\boldsymbol{x}}_i\}_{i=1}^m$ from the stochastic process $\mu^{(1)}_{X|\mathcal{F}_X}$ and n realizations $\{\widetilde{\boldsymbol{y}}_i\}_{i=1}^n$ from $\mu^{(1)}_{Y|\mathcal{F}_Y}$, an estimate of the squared 2nd order MMD is given by

$$\begin{aligned}\left(\widehat{\mathscr{D}}_K^{(2)}(X,Y)\right)^2 =& \frac{1}{m(m-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{m} K(\widetilde{\boldsymbol{x}}_i,\widetilde{\boldsymbol{x}}_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} K(\widetilde{\boldsymbol{x}}_i,\widetilde{\boldsymbol{y}}_j)\\ &+ \frac{1}{n(n-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{n} K(\widetilde{\boldsymbol{y}}_i,\widetilde{\boldsymbol{y}}_j).\end{aligned}$$

Computing this estimate boils down to evaluating the signature kernel $K(\widetilde{\boldsymbol{x}},\widetilde{\boldsymbol{y}})$ on multiple realizations $\widetilde{\boldsymbol{x}}$ from $\mu^{(1)}_{X|\mathcal{F}_X}$ and $\widetilde{\boldsymbol{y}}$ from $\mu^{(1)}_{Y|\mathcal{F}_Y}$. By Thm. 5.2.1, the signature kernel solves the following PDE

$$\frac{\partial^2 u}{\partial s\partial t} = \left\langle \frac{\widetilde{\boldsymbol{x}}(s-\delta)-\widetilde{\boldsymbol{x}}(s)}{\delta}, \frac{\widetilde{\boldsymbol{y}}(t-\delta)-\widetilde{\boldsymbol{y}}(t)}{\delta}\right\rangle_{\mathcal{H}(V)} u,$$

where the two derivatives in eq. (5.2) have been approximated by finite difference with time increment δ. It remains to explain how to estimate, for any $s,t\in[0,T]$, the inner product $\langle \widetilde{\boldsymbol{x}}(s),\widetilde{\boldsymbol{y}}(t)\rangle_{\mathcal{H}(V)}$ from sample paths of X and Y. This can be achieved using the formalism of *cross-covariance operators* (Muandet et al., 2016) as thoroughly explained in Appendix C.1, which yields to the following approximation

$$\langle \widetilde{\boldsymbol{x}}(s),\widetilde{\boldsymbol{y}}(t)\rangle_{\mathcal{H}(V)} \approx \mathbf{k}_X^s(\boldsymbol{x})'(\mathbf{K}_X^s+m\lambda\mathbf{I}_m)^{-1}\mathbf{K}_{X,Y}(\mathbf{K}_Y^t+n\lambda\mathbf{I}_n)^{-1}\mathbf{k}_Y^t(\boldsymbol{y}), \quad (5.12)$$

where $\mathbf{k}_X^s(\boldsymbol{x})\in\mathbb{R}^m$ and $\mathbf{k}_Y^t(\boldsymbol{y})\in\mathbb{R}^n$ are the vectors (here we use the notation $\boldsymbol{x}|_{[0,s]}$ to denote the restriction of the path $\boldsymbol{x}$ to the sub-interval $[0,s]\subset[0,T]$)

$$[\mathbf{k}_X^s]_i = K\left(\boldsymbol{x}_i|_{[0,s]},\boldsymbol{x}|_{[0,s]}\right) \quad \left[\mathbf{k}_Y^t\right]_i = K\left(\boldsymbol{y}_i|_{[0,t]},\boldsymbol{y}|_{[0,t]}\right),$$

$\mathbf{K}_X^s\in\mathbb{R}^{m\times m}$, $\mathbf{K}_{X,Y}\in\mathbb{R}^{m\times n}$, and $\mathbf{K}_Y^t\in\mathbb{R}^{n\times n}$ are the matrices defined element-wise by

$$[\mathbf{K}_X^s]_{i,j} = K(\boldsymbol{x}_i|_{[0,s]},\boldsymbol{x}_j|_{[0,s]}) \quad [\mathbf{K}_{X,Y}]_{i,j} = K(\boldsymbol{x}_i,\boldsymbol{y}_j),$$

$$[\mathbf{K}_Y^t]_{i,j} = K(\boldsymbol{y}_i|_{[0,t]},\boldsymbol{y}_j|_{[0,t]}),$$

where $\mathbf{I}_m$ (resp. $\mathbf{I}_n$) is the $m\times m$ (resp. $n\times n$) identity matrix. The corresponding algorithm and its complexity analysis are provided in Appendix C.2. The next theorem ensures that the estimator $\widehat{\mathscr{D}}_K^{(2)}(X,Y)$ is consistent for the

2nd order MMD.

Theorem 5.3.2. *$\widehat{\mathscr{D}}_K^{(2)}(X,Y)$ is a consistent estimator for the 2^{nd} order MMD*

$$\left|\widehat{\mathscr{D}}_K^{(2)}(X,Y) - \mathscr{D}_K^{(2)}(X,Y)\right| \xrightarrow{p} 0 \quad \text{as} \quad m,n \to \infty,$$

with $\{X_i\}_{i=1}^m \sim X$, $\{Y_i\}_{i=1}^n \sim Y$ and where convergence is in probability.

We now iterate the procedure presented so far to define higher order KMEs and MMDs.

5.3.5 Higher order kernel mean embeddings and maximum mean discrepancies

One can iterate the procedure described in Sec. 5.3.3 and recursively define, for any $n \in \mathbb{N}_*$, the n^{th} *order KME of* X as the following point in $\mathcal{H}^{(n)}(V)$

$$\mu_X^{(n)} = \int_{\boldsymbol{x}\in\mathcal{X}(\mathcal{H}^{(n-1)}(V))} K(\cdot,\boldsymbol{x})\mathbb{P}_{\mu^{(n-1)}_{X|\mathcal{F}_X}}(d\boldsymbol{x}), \tag{5.13}$$

where $\mu^{(n-1)}_{X|\mathcal{F}_X}$ is the $(n-1)^{\text{st}}$ predictive KME of X and $\mathcal{H}^{(n)}(V)$ is defined by

$$\mathcal{H}^{(n)}(V) = \underbrace{\mathcal{H}(\mathcal{H}(\ldots\mathcal{H}}_{n \text{ times}}(V)\ldots)),$$

and $\mu^{(0)}_{X|\mathcal{F}_X} = X$ and $\mathcal{H}^{(0)}(V) = V$. The associated n^{th} *order MMD* between two processes X, Y is then defined as the norm of the difference in $\mathcal{H}^{(n)}(V)$ of the two n^{th} order KMEs

$$\mathscr{D}_K^{(n)}(X,Y) = \left\|\mu_X^{(n)} - \mu_Y^{(n)}\right\|_{\mathcal{H}^{(n)}(V)}. \tag{5.14}$$

The following result generalizes Thm. 5.3.1 in that it shows that the n^{th} order MMD is a stronger (i.e. finer) discrepancy measure than all the k^{th} order MMDs of lower order $1 < k < n$.

Theorem 5.3.3. *Given two stochastic processes X, Y*

$$\mathscr{D}_K^{(n)}(X,Y) = 0 \implies \mathscr{D}_K^{(k)}(X,Y) = 0 \quad \text{for any} \quad 1 < k < n, \tag{5.15}$$

but the converse is not generally true.

Other than hypothesis testing, another important application relying on the ability of distinguishing random variables is *distribution regression* (DR) Szabó et al. (2016). In the next section we make use of the n^{th} order MMD in the setting of DR on path-valued random variables (presented in the previous chapter) and propose a family of kernels on stochastic processes whose RKHSs contains richer classes of functions than the RKHS associated to the universal kernel proposed in the previous chapter.

We note that since V is a Polish space (i.e., a separable, complete metric space) and the signature maps is continuous, in view of Christmann and Steinwart (2008, Lemma 4.33), one can easily check that all RKHSs appearing in this chapter are separable Hilbert spaces by an induction argument and therefore all regular conditional distributions are well–defined.

5.3.6 Higher order distribution regression

As we have seen in Chapter 4, DR on stochastic processes describes the supervised learning problem where the input is a collection of sample paths and the output is a vector of scalars. Denote by $\mathcal{P}(\mathcal{X}(V))$ the set of stochastic processes with sample paths on $\mathcal{X}(V)$. The goal is to learn a function $F : \mathcal{P}(\mathcal{X}(V)) \to \mathbb{R}$ from a training set of input-output pairs $\{(X_i, y_i)\}_{i=1}^n$ with $X_i \in \mathcal{P}(\mathcal{X}(V))$ and $y_i \in \mathbb{R}$, by means of a classical two-step procedure (Law et al., 2018b; Muandet et al., 2012; Smola et al., 2007).

Firstly, a stochastic process $X \in \mathcal{P}(\mathcal{X}(V))$ is embedded into its KME $\mu_X^{(1)} \in \mathcal{H}(V)$ via the signature kernel K. Secondly, another function $g : \mathcal{H}(V) \to \mathbb{R}$ is learnt by solving the minimization $\arg\min_{g\in\mathcal{H}_{\mathrm{RBF}}} \sum_i c(g(\mu_{X_i}^{(1)}), y_i)$, where c is a loss function, and $\mathcal{H}_{\mathrm{RBF}}$ is the RKHS associated to the classical Gaussian kernel $k_{\mathrm{RBF}} : \mathcal{H}(V) \times \mathcal{H}(V) \to \mathbb{R}$. This procedure materialises into a kernel on stochastic processes whose RKHS is shown to be dense in the space of functions $F : \mathcal{P}(\mathcal{X}(V)) \to \mathbb{R}$ that are continuous with respect to the weak topology.

However, a class of approximators that is universal with respect to some topology is not guaranteed to well approximate functions that are discontinous with respect to that topology (but potentially continuous with respect to a finer topology). For example, financial practitioners are often interested in calibrating financial models to market data or pricing financial instruments from observations of market dynamics. These tasks can be formulated as DR problems on stochastic processes (see experiments in Sec. 5.4.2), but the resulting learnable functions are discontinuous with respect to the 1st order MMD whilst being continuous with respect to the 2nd order MMD (Backhoff-Veraguas et al., 2019). This motivates the need to extend the kernel-based DR technique proposed in Chapter 4 to situations where the target functions are not weakly continuous, which is what Thm. 5.3.4 addresses. Before stating the result, we recall that a function $f : \mathbb{R} \to \mathbb{R}$ is called *globally analytic with non-negative coefficients* if it admits everywhere a Taylor expansion where all the coefficients are strictly positive, i.e. for any $x \in \mathbb{R}$ we have $f(x) = \sum_{i=0}^{\infty} a_i x^i$ with $a_i > 0$.

Theorem 5.3.4. *Let $f : \mathbb{R} \to \mathbb{R}$ be a globally analytic function with non-negative* *coefficients. Define the family of kernels* $K_{dr}^{(n)} : \mathcal{P}(\mathcal{X}(V)) \times \mathcal{P}(\mathcal{X}(V)) \to \mathbb{R}$ *as follows*

$$K_{dr}^{(n)}(X, Y) = f\left(\mathscr{D}_K^{(n)}(X, Y)\right), \quad n \in \mathbb{N}_* \tag{5.16}$$

Then the RKHS associated to $K_{dr}^{(n)}$ *is dense in the space of functions from* $\mathcal{P}(\mathcal{X}(V))$ *to* $\mathbb{R}$ *which are continuous with respect to the* k^{th} *order MMD for any* $1 < k \leq n$.

In Sec. 5.4 we will take $f(x) = \exp(-x^2/\sigma)$ with $\sigma > 0$. This result marks the end of our analysis. Next we apply our theoretical results in the contexts of two-sample testing, DR and causal inference.

5.4 Applications

Here we demonstrate the practical advantage of using 2nd order kernel mean embeddings, and evaluate the conditional kernel mean embedding for stochastic processes on a causal discovery task. Additional experimental details can be found in Appendix C.3 and the code is available at `https://github.com/maudl3116/higherOrderKME`.

5.4.1 Hypothesis testing on filtrations

We start by considering two processes X_n and X with transition probabilities depicted in Fig. 5.2. Although the laws $\mathbb{P}_n$ and $\mathbb{P}$ get arbitrarily close for large n, their filtrations are very different. Indeed, the two processes have different information structures available before time $t = 1$: for any $0 < t \leq 1$, the trajectory of X_n is deterministic, whilst the progression of X remains random until $t = 1$. Being able to distinguish two such stochastic processes is crucial in

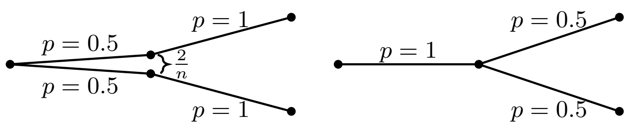
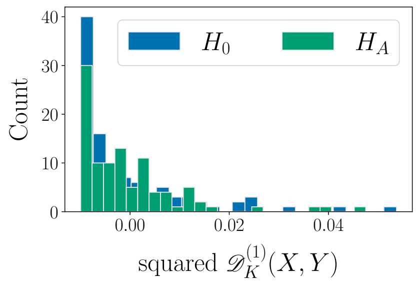


Figure 5.2: Two stochastic processes X_n (left) and X (right).

quantitative finance: if $\mathbb{P}_n$ and $\mathbb{P}$ are the laws of two traded assets, $\mathbb{P}_n$ gives an arbitrage opportunity. As shown in Fig. 5.3, the 2nd order MMD can distinguish these two processes with similar laws ($n = 5 \cdot 10^5$) but different filtrations, while the 1st order MMD fails to do so. We refer the interested reader to Bonnier et al. (2020, Appendix A) for examples of sequences of processes X_n,Y_n such that the 1st and 2nd order MMDs converge to zero, but not the 3rd order MMD.



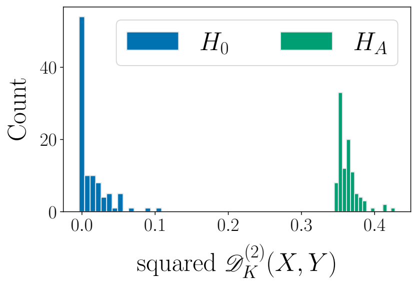


Figure 5.3: *Left:* Empirical distribution of the 1st order MMD. Under H_0 the two measures are both equal to $\mathbb{P}$ and we use 500 samples from each. Under H_A with $\mathbb{P}$ and $\mathbb{P}_n$ where $n = 5 \cdot 10^5$, and we use 500 samples. *Right:* Same for the 2nd order MMD. Histograms are obtained by computing 500 independent instances of the MMD.

5.4.2 Applications of higher order distribution regression to quantitative finance

In this section we use kernel Ridge regression and support vector machine (SVM) classification equipped with the kernel $K^{(2)}_{\text{dr}}$ from Thm. 5.3.4 to address two real-world problems arising in quantitative finance, notably the calibration of the *rough Bergomi model* (Bayer et al., 2016) and the approximation of solutions of optimal stopping problems (Becker et al., 2019). We benchmark our filtration-sensitive kernel $K^{(2)}_{\text{dr}}$ against a range of kernels, including $K^{(1)}_{\text{dr}}$.

The rough Bergomi model is a rough volatility model (Gatheral et al., 2018) satisfying the following stochastic dynamics

$$dS_t = \sqrt{V_t} S_t dW^1_t, \quad V_t = \int_0^t \kappa(s,t) dZ_s, \quad Z_t = \rho W^1_t + \sqrt{1-\rho^2} W^2_t, \quad (5.17)$$

where W^1, W^2 are two independent Brownian motions and $\kappa(s,t) = (t-s)^{h-0.5}$ where here we take $h = 0.2$. The model in eq. (5.17) is non-Markovian in the sense that the conditional law of $S \mid \mathcal{F}_{S_t}$ depends pathwise on the past history of the process. Of particular importance is the correct retrieval of the sign of the correlation parameter ρ (Gassiat et al., 2019). We consider 50 parameter values $\{\rho_i\}_{i=1}^{50}$ chosen uniformly at random from $[-1,1]$. Each ρ_i is regressed on a collection of $m = 200$ sample trajectories. We use an SVM classifier endowed with different kernels (Table 5.1).

One of the most studied optimal stopping problems is the pricing of an American option with a non-negative payoff function $g : \mathbb{R}^d \to \mathbb{R}$. Stock prices are assumed to follow a d-dimensional stochastic process X. The price of the corresponding option is the solution of the optimal stopping problem $V(x) = \sup_\tau \mathbb{E}[g(X_\tau) \,|\, X_0 = x]$, where the supremum is taken over stopping times τ. Despite significant advances, pricing American options remains one of the most computationally challenging problems in financial optimization,

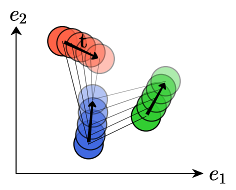
Table 5.1: Quantitative finance examples. Average performances with standard errors in parenthesis.

Kernel	Rough Bergomi model calibration (Acc.)	Optimally stopping fBms (MSE $\times 10^{-3}$)
RBF	87% (5%)	1.07 (0.75)
Matérn	87% (3%)	2.75 (3.05)
$K_{\text{dr}}^{(1)}$	91% (3%)	0.90 (0.34)
$K_{\text{dr}}^{(2)}$	**93%** (3%)	**0.52** (0.07)

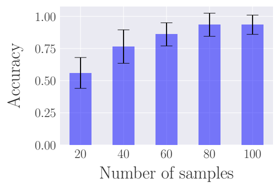
in particular when the underlying process X is non-Markovian. This is the setting we consider here, by modelling X as fractional Brownian motion (fBm) (Duncan et al., 2000) all starting at $X_0 = 0$, with different Hurst exponents $h \in (0, 1)$. We follow the setup of Becker et al. (2019); Herrera et al. (2021) and use $g = id$. We note that the resulting optimal stopping problem falls outside the standard American option pricing problem setting, because $g(X_\tau)$ is allowed to be negative and fBm is not typically used as a stock price model in quantititative finance. It is nevertheless considered a respected challenging example for optimal stopping algorithms (Becker et al., 2019; Herrera et al., 2021). True target optimal values ("prices") are obtained via expensive Monte Carlo simulations (Longstaff and Schwartz, 2001). We consider 25 values of $\{h_i\}_{i=1}^{25}$ sampled uniformly at random in $[0.2, 0.8]$ and use 500 samples from each fBm. As shown in Table 5.1 our kernel $K_{\text{dr}}^{(2)}$ yields the best results on both tasks (rough Bergomi model calibration and optimally stopping fBms), systematically outperforming other classical kernels as well as the kernel $K_{\text{dr}}^{(1)}$ introduced in Chapter 4.

5.4.3 Inferring causal graph for interacting bodies

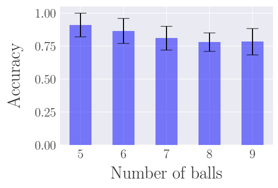
Finally, we consider the task of recovering the causal relationships between interacting bodies solely from observations of their multidimensional trajectories. We employ the multi-body interaction simulator from Li et al. (2020a) in order to simulate an environment where N balls are connected by invisible physical relations (e.g. a spring) and describe 2D trajectories (see Fig. 5.4a with $N = 3$ and 2 springs). At the beginning of a simulated episode, the initial positions of the balls are generated at random, and during the episode, the balls are subject to forces with random intensity and direction. By simulating m episodes we end up with m sample trajectories for each of the N balls. We use the kernel PC (kPC) algorithm (Sun et al., 2007)—which relies on conditional independence testing— with the signature kernel and evaluate its ability to recover whether any two balls are connected or not. We vary m and N and report the results in Figs. 5.4b and 5.4c. Each experiment is run 15 times, 30% of the runs are used to choose the hyperparameters, and the reported results have been obtained on the remaining runs. We note that for finite datasets conditional independence testing is hard without additional assumptions, as discussed in Lundborg et al. (2021); Shah and Peters (2020).



(a) 3 interacting balls describing trajectories in the 2D plane over time.



(b) Accuracy on binary classification of edges with a varying number of sample episodes (5 balls)



(c) Accuracy on binary classification of edges with a varying number of balls (100 samples)

Figure 5.4: Inferring the graph structure of interacting bodies: results of the kPC algorithm for multidimensional stochastic processes.

5.5 Conclusion

In this chapter, we have introduced a family of higher order KMEs by conditioning a stochastic process on its filtration, generalizing the classical notion of KME. We have derived an empirical estimator for the 2^{nd} order MMD and proved its consistency. Then, we have proposed a filtration-sensitive kernel two-sample test and showed with simulations its ability to capture information that gets missed by the standard MMD test. In addition, we have constructed a family of universal kernels on stochastic processes that can be used to solve real-world calibration and optimal stopping problems in quantitative finance via kernel Ridge regression. Finally, we have designed a causal-discovery algorithm using conditional independence tests to recover the causal graph of structural dependencies among interacting bodies solely from observations of their multidimensional trajectories.

Chapter 6

Resolution-Invariant Learning of Spatio-temporal Dynamics

So far, we have developed methodologies to perform regression (or classification) analysis on multivariate time-series (Chapter 3) and on *sets* of multivariate time-series (Chapters 4 and 5). Although in theory, the signature (used in Chapter 3) can be defined on spaces of paths valued in an infinite-dimensional state space, their application to spatio-temporal data remains challenging. Furthermore, the methodologies developed in Chapters 4 and 5 are best suited when the input set can be modelled as i.i.d. samples from a stochastic process. This assumption breaks down for spatio-temporal data which are sets of multivariate streams, but which exhibit spatial dependencies.

In this chapter, we propose to resolve this issue by leveraging *Stochastic partial differential equation*s (SPDEs), which are the mathematical tool of choice for modelling spatio-temporal dynamics under the influence of randomness. Based on the notion of a mild solution of an SPDE, we introduce a neural architecture to learn the solution operators of PDEs with (possibly stochastic) forcing from partially observed data. The proposed *Neural SPDE* model provides an extension to two popular classes of physics-inspired architectures. On the one hand, it extends Neural CDEs and variants—continuous-time analogues of RNNs—in that it is capable of processing incoming sequential information arriving irregularly in time and observed at arbitrary spatial resolutions. On the other hand, it extends Neural Operators—generalizations of neural networks to model mappings between spaces of functions—in that it can parameterize solution operators of SPDEs depending simultaneously on the initial condition and a realization of the driving noise. By performing operations in the spectral domain, we show how a Neural SPDE can be evaluated in two ways, either by calling an ODE solver (emulating a spectral Galerkin scheme), or by solving a fixed point problem. Experiments on various semilinear SPDEs, including the stochastic Navier-Stokes equations, demonstrate how the

Neural SPDE model is capable of learning complex spatio-temporal dynamics in a resolution-invariant way, with better accuracy and lighter training data requirements compared to alternative models, and up to 3 orders of magnitude faster than traditional solvers.

In this chapter, $\mathbf{x}$ denotes a spatial variable. The functions considered in this chapter depend on t and $\mathbf{x}$. To unclutter the notations, we will drop the dependence on space and write $u(t)$ for $u(t, \mathbf{x})$ where no confusion is caused by doing so.

6.1 Introduction

Motivated by the fact that many classical deep learning architectures can be interpreted as approximations to differential equations, there has been an increased interest in combining neural networks and differential equations into a hybrid approach, dubbed *neural differential equations* (Chen et al., 2018). When the data are sequential, recurrent neural networks (RNNs) and variants (GRU, LSTM etc.) are a popular modelling choice that can be regarded as a discrete approximation to some function of an underlying continuous signal.

Neural controlled differential equations (Neural CDEs) (Kidger et al., 2020) and variants (Neural SDEs (Kidger et al., 2021), Neural RDEs (Morrill et al., 2021)) can be viewed as continuous-time analogues of RNNs. In essence, Neural CDEs embed a neural network f_θ as the vector field of a differential equation $d\boldsymbol{z}(t) = f_\theta(\boldsymbol{z}(t))d\boldsymbol{x}(t)$ driven by a control path $\boldsymbol{x}(t)$, and the parameters θ are then learned from the data to optimize a given loss function. The term "$d\boldsymbol{x}(t)$" means that the solution $\boldsymbol{z}(t)$ of the equation (e.g. attention required from a doctor) can change in response to a change of an external stream of information $\boldsymbol{\xi}(t)$ (e.g. heart rate of a patient). Despite offering many advantages for modelling temporal dynamics, Neural CDEs are not designed to process signals varying both in space and in time such as videos or physical fields arising from PDE-dynamics.

Neural Operators (Kovachki et al., 2021) are generalizations of neural networks capable of modelling mappings between spaces of functions, therefore offering an attractive option for learning with spatio-temporal data. However, Neural Operators architectures generally fail to incorporate randomness into the systems they describe. In many cases the presence of noise leads to new phenomena, both at the mathematical and the physical level, often describing more complex and realistic dynamics than the ones arising from deterministic PDEs.

SPDEs are the mathematical tool of choice to model many physical, biological, and economic systems subject to the influence of randomness, be it intrinsic (e.g. quantifying uncertainty) or extrinsic (e.g. modelling environmental random perturbations). Examples include the *Kardar–Parisi–Zhang (KPZ) equations* for random interface growth modelling, for instance, the propagation of a forest fire from a burnt region to an unburnt region (Hairer, 2013), the *Ginzburg-Landau equation* describing phase transitions of ferromagnets and superconductors (Temam, 2012), or the *stochastic Navier-Stokes equations* modelling the dynamics of a turbulent fluid flow under the presence of local random fluctuations (Mikulevicius and Rozovskii, 2004). For an introduction to SPDEs, see Hairer (2009); a comprehensive textbook is Holden et al. (1996).

Similarly to a *stochastic differential equation* (SDE), the solution $\boldsymbol{u}$ to an SPDE is characterized by an initial condition $\boldsymbol{u}_0$ and a driving noise $\boldsymbol{\xi}$. However, in the case of SDEs $(\boldsymbol{u}_0, \boldsymbol{\xi}(t), \boldsymbol{u}(t))$ are vectors, while in the case of SPDEs they are functions. Thus, Neural CDEs are not adapted to learn solution operators of SPDEs from data, due to the non-linear interactions between the various space-time points. While Neural Operators can model operators mapping the initial condition $\boldsymbol{u}_0$ or a realization of the forcing noise $\boldsymbol{\xi}$ to the solution $\boldsymbol{u}$ of a PDE, they do not offer a natural mechanism to handle SPDEs solution operators, where the solution $\boldsymbol{u}$ depends simultaneously on $\boldsymbol{u}_0$ and $\boldsymbol{\xi}$.

In this chapter, we introduce the *neural stochastic partial differential equation* (Neural SPDE) model capable of learning solution operators of SPDEs of the form $(\boldsymbol{u}_0, \boldsymbol{\xi}) \mapsto \boldsymbol{u}$ from partially observed data. A Neural SPDE can be used to process (continuously in time) incoming sequential information arriving at an arbitrary spatial resolution. We propose two possible ways to evaluate a Neural SPDE, either by calling an ODE solver (emulating a spectral Galerkin scheme) or by solving a fixed-point problem. We perform extensive experiments on various semilinear SPDEs, including the stochastic Ginzburg-Landau, Korteweg-De Vries and Navier-Stokes equations. The outcomes illustrate several aspects of our model: 1) it is space and time resolution-invariant, meaning that even if trained on a lower resolution, it can be directly evaluated on a higher resolution without sacrificing performance; 2) it requires a lower amount of training data compared to alternative models; 3) for both choices of evaluation, it inherits memory-efficient backpropagation capabilities provided by existing adjoint-based and implicit-differentiation-based methods; 4) its evaluation is up to 3 orders of magnitude faster than traditional numerical solvers, so that once trained, the model can be used as a fast surrogate solver.

In Sec. 6.2 we provide a brief introduction to SPDEs which will help us to define our Neural SPDE model in Sec. 6.3. In Sec. 6.4 we present related work on Neural CDEs and Neural Operators, followed by numerical experiments in Sec. 6.5. In Appendix D.1, we provide an overview of the computational aspects of SPDEs used to design the model and solve SPDEs numerically. Additional experiments can be found in Appendix D.2.

6.2 Background on SPDEs

Let $T > 0$ and $d, d_u, d_\xi \in \mathbb{N}_*$. Let $\mathcal{D} \subset \mathbb{R}^d$ be a bounded domain. Let $\mathcal{H}_u = \{f : \mathcal{D} \to \mathbb{R}^{d_u}\}$ and $\mathcal{H}_\xi = \{f : \mathcal{D} \to \mathbb{R}^{d_\xi}\}$ be two Hilbert spaces of functions on $\mathcal{D}$ with values in $\mathbb{R}^{d_u}$ and $\mathbb{R}^{d_\xi}$, respectively.

We consider a large class of SPDEs of the type

$$d\boldsymbol{u} = (\mathcal{L}\boldsymbol{u} + F(\boldsymbol{u}))\, dt + G(\boldsymbol{u})d\mathbf{W}(t), \qquad \boldsymbol{u}(0) = \boldsymbol{u}_0 \in \mathcal{H}_u, \tag{6.1}$$

where $\mathbf{W}(t)$ is an infinite-dimensional Q-Wiener process (Lord et al., 2014, Definition 10.6) or a cylindrical Wiener process (Hairer, 2009, Definition 3.54), $F : \mathcal{H}_u \to \mathcal{H}_u$ and $G : \mathcal{H}_u \to L(\mathcal{H}_\xi, \mathcal{H}_u)$ are two continuous operators[1], and where $\mathcal{L}$ is a linear differential operator generating a *semigroup* $e^{t\mathcal{L}} : \mathcal{H}_u \to \mathcal{H}_u$ (Hairer, 2009, Section 4).

A function $\boldsymbol{u} : [0, T] \to \mathcal{H}_u$ is defined to be a *mild solution* of the SPDE in eq. (6.1) if for any $t \in [0, T]$ it satisfies

$$\boldsymbol{u}(t) = e^{t\mathcal{L}}\boldsymbol{u}_0 + \int_0^t e^{(t-s)\mathcal{L}} F(\boldsymbol{u}(s))ds + \int_0^t e^{(t-s)\mathcal{L}} G(\boldsymbol{u}(s))d\mathbf{W}(s),$$

where the second integral is a stochastic integral (Hairer, 2009, Definition 3.57). An SPDE can be informally thought of as an SDE with solutions evolving in $\mathcal{H}_u$ and driven by an infinite-dimensional Brownian motion $\mathbf{W}$. Assuming global Lipschitz regularity on F and G, a mild solution $\boldsymbol{u}$ to eq. (6.1) exists and is unique (Hairer, 2009, Theorem 6.4) at least for short times.

In view of machine learning applications, one should think of $\mathbf{W}$ as a continuous space-time embedding of an underlying spatio-temporal data stream. In this chapter we are only going to consider $\mathbf{W}$ to be a discretized sample path of a Wiener process, but we emphasize that the Neural SPDE model extends, in principle, beyond the scope of SPDEs and could be used, for example, to process videos in computer vision applications.

We follow Friz and Hairer (2020) and consider the regularization $\boldsymbol{\xi}^\epsilon := \delta^\epsilon * \boldsymbol{\xi}$ of white noise $\boldsymbol{\xi}$ with a compactly supported smooth mollifier δ^ϵ. It is a classical result in rough path theory (Wong-Zakai) that the sequence of random ODEs driven by the mollification of Brownian motion converges in probability to a limiting process that does not depend on the choice of mollifier and agrees with the Stratonovich solution of the SDE. Furthermore, the solution map is continuous in an appropriate rough path topology. This result nicely extends to the setting of SPDEs driven by a finite-dimensional noise (Friz and Hairer, 2020, Theorem 1.3): if $\boldsymbol{u}^\epsilon$ denotes the random PDE solutions driven by $\boldsymbol{\xi}^\epsilon dt$ (instead of $\circ d\mathbf{W}_t$), then $\boldsymbol{u}^\epsilon$ converges in probability to a limiting process given

[1] $L(\mathcal{H}_\xi, \mathcal{H}_u)$ denotes the space of bounded linear operators from $\mathcal{H}_\xi$ to $\mathcal{H}_u$.

by Stratonovich SPDE solution.

In our setting though, we consider dynamics driven by an infinite-dimensional noise. In this case, there is no Stratonovich formulation for such class of equations, since the corresponding Itô-Stratonovich correction would be infinite. Therefore, the stochastic integral $(d\mathbf{W}_t)$ is to be interpreted in the Itô sense. Hairer and Pardoux (2015, Theorem 1.1) show that for the case of the heat operator and under appropriate renormalization and drift correction, the random PDE solution $\boldsymbol{u}^\epsilon$ converges in probability to the Itô solution of the SPDE. Furthermore, the solution map is continuous in an appropriate regularity structures topology.

Hence, computationally it is reasonable to assume some degree of smoothness of $\mathbf{W}$.[2] As done in Kidger et al. (2020) for Neural CDEs, we can rewrite the mild solution of the mollified version of eq. (6.1) as the following stochastically forced PDE

$$\boldsymbol{u}(t) = e^{t\mathcal{L}}\boldsymbol{u}_0 + \int_0^t e^{(t-s)\mathcal{L}} H_\xi(\boldsymbol{u}(s))ds \tag{6.2}$$

where $H_\xi(\boldsymbol{u}(t)) := F(\boldsymbol{u}(t)) + G(\boldsymbol{u}(t))\boldsymbol{\xi}(t)$ for $\boldsymbol{\xi} = \dot{\mathbf{W}}$. We also assume enough spatial regularity and take $\mathcal{H}_u = L^2(\mathcal{D}, \mathbb{R}^{d_u})$ and $\mathcal{H}_\xi = L^2(\mathcal{D}, \mathbb{R}^{d_\xi})$. We will refer to $\boldsymbol{\xi}$ as white noise if is $\mathbf{W}$ a cylindrical Wiener process (see Appendix D.1.2 for further details on Wiener processes).

In the next section we introduce the Neural SPDE model.

6.3 Neural SPDEs

For a large class of differential operators $\mathcal{L}$, the action of the semigroup $e^{t\mathcal{L}}$ can be written as an integral against a kernel function $\mathcal{K}(t) : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^{d_u \times d_u}$ such that

$$(e^{t\mathcal{L}}\boldsymbol{h})(\mathbf{x}) = \int_{\mathcal{D}} \mathcal{K}(t, \mathbf{x}, \mathbf{y})\boldsymbol{h}(\mathbf{y})\mu(d\mathbf{y}), \tag{6.3}$$

for any $\boldsymbol{h} \in \mathcal{H}_u$, any $\mathbf{x} \in \mathcal{D}$ and $t \in [0, T]$, and where μ is a Borel measure on $\mathcal{D}$. As in Kovachki et al. (2021), here we take μ to be the Lebesgue measure in $\mathbb{R}^d$, but other choices can be made, for example to incorporate prior information. We assume that $\mathcal{K}$ is stationary in $\mathbf{x}$ so that eq. (6.2) can be rewritten in terms of the spatial convolution $*$

$$\boldsymbol{u}(t) = \mathcal{K}(t) * \boldsymbol{u}_0 + \int_0^t \mathcal{K}(t-s) * H_\xi(\boldsymbol{u}(s))ds \tag{6.4}$$

We propose two different ways of parameterizing the kernel $\mathcal{K}$ and evaluating Equation (6.4); we will outline the details of both methods in Sections 6.3.2 and 6.3.3 below. For now, we will denote either parameterization by $\mathcal{K}_\theta$.

[2]Although, extending the result in (Hairer and Pardoux, 2015) to a generic differential operators would require a similar rigorous proof, which is out of the scope of this thesis.

For a large class of SPDEs of the form of eq. (6.1), both F and G are local operators acting on a function $\boldsymbol{h} \in \mathcal{H}_u$. In other words, the evaluations $F(\boldsymbol{h})(\mathbf{x})$ and $G(\boldsymbol{h})(\mathbf{x})$ at any point $\mathbf{x} \in \mathcal{D}$ only depend on the evaluation $\boldsymbol{h}(\mathbf{x})$ at $\mathbf{x}$, and not on the evaluation $\boldsymbol{h}(\mathbf{y})$ at some other point $\mathbf{y} \in \mathcal{D}$ in the neighbourhood of $\mathbf{x}$. This motivates the following model.

6.3.1 The model

Let $d_h > d_u$ be an integer, $\mathcal{H}_h := L^2(\mathcal{D}, \mathbb{R}^{d_h})$, and let

$$\begin{aligned} L_\theta &: \mathbb{R}^{d_u} \to \mathbb{R}^{d_h}, & \Pi_\theta &: \mathbb{R}^{d_h} \to \mathbb{R}^{d_u}, \\ F_\theta &: \mathbb{R}^{d_h} \to \mathbb{R}^{d_h}, & G_\theta &: \mathbb{R}^{d_h} \to \mathbb{R}^{d_h \times d_\xi}, \end{aligned}$$

four feedforward neural networks. For any differentiable control $\boldsymbol{\xi} : [0, T] \to \mathcal{H}_\xi$, define the map $H_{\theta,\xi} : \mathcal{H}_u \to \mathcal{H}_u$

$$H_{\theta,\xi}(\boldsymbol{h})(\mathbf{x}) = F_\theta(\boldsymbol{h}(\mathbf{x})) + G_\theta(\boldsymbol{h}(\mathbf{x}))\boldsymbol{\xi}(t, \mathbf{x}),$$

for any $\boldsymbol{h} \in \mathcal{H}_u$, any $\mathbf{x} \in \mathcal{D}$ and $t \in [0, T]$.

The *neural stochastic partial differential equation* (Neural SPDE) model is defined as follows: $\boldsymbol{z}_0(\mathbf{x}) = L_\theta(\boldsymbol{u}_0(\mathbf{x}))$,

$$\boldsymbol{z}(t) = \mathcal{K}_\theta(t) * \boldsymbol{z}_0 + \int_0^t \mathcal{K}_\theta(t - s) * H_{\theta,\xi}(\boldsymbol{z}(s))ds \tag{6.5}$$

and $\boldsymbol{u}(t, \mathbf{x}) = \Pi_\theta(\boldsymbol{z}(t, \mathbf{x}))$, for any $\mathbf{x} \in \mathcal{D}$ and any $t \in [0, T]$. A schematic view of how Neural SPDEs can process spatio-temporal data is given in Fig. 6.1. We note that globally Lipschitz conditions can be imposed using, for example, ReLU or tanh activation functions in the neural networks F_θ and G_θ.

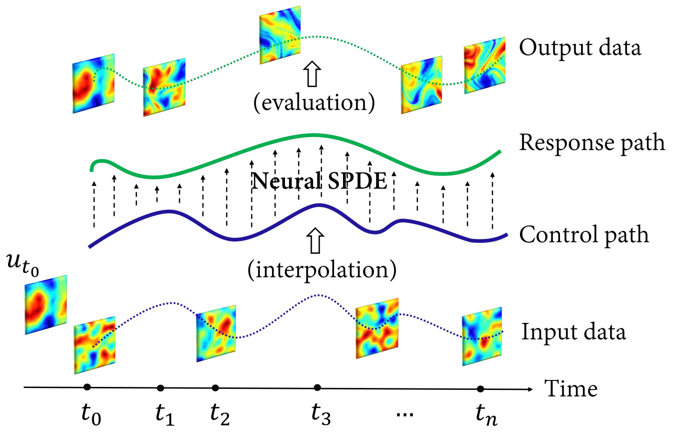


Figure 6.1: Schematic view of a Neural SPDE. The model is formulated directly at the level of function spaces, which makes it possible to deploy the model on different discretization meshes.

In the next two subsections, we propose two numerical methods to evaluate the Neural SPDE model in eq. (6.5) which produce two different parameterisations for the kernel $\mathcal{K}$ in eq. (6.3).

6.3.2 Kernel parameterization 1: ODE solver approach

Using the *convolution theorem* we can rewrite the integral in eq. (6.5) as follows

$$\boldsymbol{z}(t) = \mathcal{F}^{-1}\Big(\mathcal{F}(\mathcal{K}(t))\,\mathcal{F}(\boldsymbol{z}_0) + \int_0^t \mathcal{F}(\mathcal{K}(t-s))\,\mathcal{F}(H_{\theta,\xi}(\boldsymbol{z}(s)))\,ds\Big), \qquad (6.6)$$

where $\mathcal{F}, \mathcal{F}^{-1}$ are the *d-dimensional Fourier transform* (FT) and its inverse (see Def. 20). Assuming that $\mathcal{L}$ is a polynomial differential operator[3] of degree N of the form

$$\mathcal{L} = \sum_{n=0}^{N} \sum_{\substack{n_1,\ldots,n_d \\ n_1+\ldots+n_d=n}} \mathbf{C}_{n_1,\ldots,n_d} \frac{\partial^n}{\partial x_1^{n_1}\ldots\partial x_d^{n_d}},$$

where $\mathbf{C}_{n_1,\ldots,n_d} \in \mathbb{C}^{d_h\times d_h}$ are complex matrices, then the FT of the kernel associated to $\mathcal{L}$ satisfies

$$\mathcal{F}(\mathcal{K}(t))\,(\mathbf{y}) = e^{tP(i\mathbf{y})} \in \mathbb{C}^{d_h\times d_h},$$

for any $\mathbf{y} \in \mathbb{C}^d$, where e is the matrix exponential and P is the following matrix-valued polynomial

$$P(\mathbf{y}) = \sum_{n=0}^{N} \sum_{\substack{n_1,\ldots,n_d \\ n_1+\ldots+n_d=n}} (2\pi)^n y_1^{k_1}\ldots y_d^{k_d}\mathbf{C}_{n_1,\ldots,n_d}$$

This means that there exists a map $A : \mathbb{C}^d \to \mathbb{C}^{d_h\times d_h}$ such that $\mathcal{F}(\mathcal{K}(t))\,(\mathbf{y}) = e^{tA(\mathbf{y})}$. Thus, eq. (6.6) becomes

$$\boldsymbol{z}(t) = \mathcal{F}^{-1}\Big(\underbrace{e^{tA}\mathcal{F}(\boldsymbol{z}_0) + \int_0^t e^{(t-s)A}\mathcal{F}(H_{\theta,\xi}(\boldsymbol{z}(s)))\,ds}_{:=\boldsymbol{v}(t)}\Big),$$

where $\boldsymbol{v}(t) : \mathbb{C}^d \to \mathbb{C}^{d_h}$ is the solution of the following ODE

$$\begin{aligned}\frac{d\boldsymbol{v}}{dt}(t) &= A\boldsymbol{v}(t) + \mathcal{F}(H_{\theta,\xi}(\boldsymbol{z}(t)))\\ &= A\boldsymbol{v}(t) + \mathcal{F}\big(H_{\theta,\xi}(\mathcal{F}^{-1}(\boldsymbol{v}(t)))\big).\end{aligned}$$

[3] This already contains a big class of SPDEs studied in physics. One could consider $C_{n_1,\ldots,n_d}$ to be a function of space for a more general setting.

Hence, $\boldsymbol{z}(t)$ can be obtained by applying the inverse FT to the output of an ODE solver on $[0,t]$ with initial condition $\mathcal{F}(\,\boldsymbol{z}_0\,)$, vector field $\Psi_{\theta,\xi} := A+\mathcal{F}\circ H_{\theta,\xi}\circ\mathcal{F}^{-1}$,

$$\boldsymbol{z}(t) \approx \mathcal{F}^{-1}(\,\mathrm{ODESolve}(\mathcal{F}(\,\boldsymbol{z}_0\,)\,, \Psi_{\theta,\xi}, [0,t])\,)\,.$$

This approach can naturally be seen as a "neural version" of the classical *spectral Galerkin solver* for SPDEs described in Appendix D.1.3, with nonlinearities F, G and differential operator $\mathcal{L}$ parameterized as outlined in this section.

We note that this numerical evaluation of the Neural SPDE model in eq. (6.5) allows us to inherit memory efficient adjoint-based backpropagation capabilities as in the evaluation of a Neural CDE. For further details on adjoint-based backpropagation, we refer the reader to Chen et al. (2018).

6.3.3 Kernel parameterization 2: fixed point approach

We now propose a second option for parametrizing the kernel. To this aim, we will make use of three different versions of the FT: the time-only FT $\mathcal{F}_1$ and its inverse $\mathcal{F}_1^{-1}$, the space-only FT $\mathcal{F}_d$ and its inverse $\mathcal{F}_d^{-1}$, and the space-time FT $\mathcal{F}_{d+1}$ and its inverse $\mathcal{F}_{d+1}^{-1}$ (see Def. 20). Denoting by $\star$ the space-time convolution, the integral in eq. (6.5) can be rewritten as

$$\boldsymbol{z}(t) = \mathcal{K}(t) * \boldsymbol{z}_0 + \left(\mathcal{K} \star \mathbb{1}_{s\geq 0} H_{\theta,\xi}(\boldsymbol{z})\right)(t),$$

where $\mathbb{1}_{s\geq 0}$ is the indicator function restricting the temporal domain to the positive real line. Using again the convolution theorem, we obtain the following

$$\boldsymbol{z}(t) = \mathcal{F}_d^{-1}(\,\mathcal{F}_d(\,\mathcal{K}(t)\,)\,\mathcal{F}_d(\,\boldsymbol{z}_0\,)\,) + \mathcal{F}_{d+1}^{-1}(\,\mathcal{F}_{d+1}\,(\mathcal{K})\,\mathcal{F}_{d+1}(\,\mathbb{1}_{s\geq 0}H_{\theta,\xi}(\boldsymbol{z})\,)\,)\,(t),$$

where all multiplications are matrix-vector multiplications.

Parameterizing $\mathcal{F}_{d+1}(\,\mathcal{K}\,)\,(\mathbf{y})$ directly in Fourier space as a complex tensor B, eq. (6.5) becomes

$$\boldsymbol{z}(t) = \mathcal{F}_d^{-1}\big(\,\mathcal{F}_1^{-1}(\,B\,)(t)\,\mathcal{F}_d(\,\boldsymbol{z}_0\,)\big) + \mathcal{F}_{d+1}^{-1}(\,B\,\mathcal{F}_{d+1}(\,\mathbb{1}_{s\geq 0}H_{\theta,\xi}(\boldsymbol{z})\,)\,)\,(t),$$

where we used the fact that $\mathcal{F}_d(\,\mathcal{K}(t)\,) = \mathcal{F}_1^{-1}(\,\mathcal{F}_{d+1}(\,\mathcal{K}\,)\,)\,(t)$. Hence, the solution $\boldsymbol{z}$ can be obtained by solving the fixed point problem $\boldsymbol{z} = \Phi_{\theta,\xi}(\boldsymbol{z})$, where

$$\Phi_{\theta,\xi}(\boldsymbol{z})(t) := \mathcal{F}_d^{-1}\big(\,\mathcal{F}_1^{-1}(\,B\,)(t)\,\mathcal{F}_d(\,\boldsymbol{z}_0\,)\big) + \mathcal{F}_{d+1}^{-1}(\,B\,\mathcal{F}_{d+1}(\,\mathbb{1}_{s\geq 0}H_{\theta,\xi}(\boldsymbol{z})\,)\,)\,(t).$$

This can be solved numerically by Picard's iteration:

$$\boldsymbol{z} \approx \mathrm{FixedPointSolve}(\boldsymbol{z}_0, \Phi_{\theta,\xi}).$$

Analogously to the adjoint-based backpropagation for the ODE solver approach mentioned in Sec. 6.3.2, there is a mechanism that leverages the *implicit function theorem* to backpropagate through the operations of a fixed point solver in a memory-efficient way. We refer the reader to Bai et al. (2019) for further details.

6.3.4 Space-time resolution-invariance

Let $D = \{\mathbf{x}_1, ..., \mathbf{x}_m\} \subset \mathcal{D}$ be an m-points discretization of the spatial domain $\mathcal{D}$, and let $\mathcal{T} = \{t_0, ..., t_n\} \subset [0, T]$ be an $(n+1)$-points discretization of the time interval $[0, T]$ with $t_0 < ... < t_n$. As depicted in Fig. 6.1, the input data corresponds to an irregularly sampled sequence of partially-observed spatial observations on the space-time grid $D \times \mathcal{T}$. The data are then interpolated to a continuous spatio-temporal signal $\boldsymbol{\xi}$ (and initial condition $\boldsymbol{u}_0$). By construction, the Neural SPDE model operates on the tuple of functions $(\boldsymbol{u}_0, \boldsymbol{\xi})$ to produce a spatio-temporal response $\boldsymbol{u}$, also continuous in space and time; the function $\boldsymbol{u}$ can then be evaluated at an arbitrary space-time resolution (possibly different from $D \times \mathcal{T}$) to produce the output data. Therefore, even if trained on a coarser resolution, a Neural SPDE can be directly evaluated on a finer resolution. In Sec. 6.5 we will empirically demonstrate this *zero-shot super-resolution* property of the Neural SPDE model. We note that the FTs are numerically approximated using the *discrete Fourier transform* (DFT) and selecting a maximum number of frequency modes [4] (see Appendix D.1.1 for details).

Next, we provide a short overview of Neural CDEs and variants, and Neural Operators, both used as main benchmarks in the experimental section.

6.4 Related work

6.4.1 Neural CDEs, SDEs, RDEs

A *neural controlled differential equation* (Neural CDE), as popularized by (Kidger et al., 2020), is a parametric model that takes as input a time-series interpolated into a continuous path $\boldsymbol{x} : [0, T] \to \mathbb{R}^{d_\xi}$ of bounded variation. The architecture of the model consists of a matrix-valued feedforward neural network $f_\theta : \mathbb{R}^{d_h} \to \mathbb{R}^{d_h \times d_\xi}$ (satisfying some minimal Lipschitz regularity) parameterizing the vector field of the following dynamics

$$\boldsymbol{z}_0 = \ell_\theta(\boldsymbol{u}_0), \quad \boldsymbol{z}(t) = \boldsymbol{z}_0 + \int_0^t f_\theta(\boldsymbol{z}(s)) d\boldsymbol{x}(s), \quad \boldsymbol{u}(t) = \pi_\theta(\boldsymbol{z}(t)),$$

[4]The DFT approximates the Fourier series expansion truncated at a maximum number of modes. This allows to specify the shape of the two complex tensors A ($k^1_{\max} \times \ldots \times k^d_{\max} \times d_h \times d_h$) in Sec. 6.3.2 and B ($k^1_{\max} \times \ldots \times k^{d+1}_{\max} \times d_h \times d_h$) in Sec. 6.3.3. The $k^i_{\max}$ are treated as hyperparameters of the model.

where $\ell_\theta : \mathbb{R}^{d_u} \to \mathbb{R}^{d_h}$ and $\pi_\theta : \mathbb{R}^{d_h} \to \mathbb{R}^{d_u}$ are feedforward neural networks. The output $\boldsymbol{u}(t)$ is then fed to a loss function (mean squared, cross-entropy, etc.) and trained via stochastic gradient descent in the usual way. Depending on the regularity assumptions of the control $\boldsymbol{x}(t) = \int_0^t \boldsymbol{\xi}(s)ds$, the above integral can be interpreted as a Riemann–Stieltjes, a stochastic, or even a rough integral.

In the original version of Neural CDEs (Kidger et al., 2020), $\boldsymbol{x}$ is assumed differentiable and obtained via natural cubic splines interpolation of the original time-series; therefore the term "$d\boldsymbol{x}(s)$" can be interpreted as "$\dot{\boldsymbol{x}}(s)ds$" and the integral can be evaluated numerically via a call to an ODE solver of choice (Euler, Runge-Kutta, adaptive schemes...).

Neural stochastic differential equations (Neural SDEs) are Neural CDEs where the control $\boldsymbol{\xi}$ is a sample path from a $\mathbb{R}^{d_\xi}$-dimensional Brownian motion; Neural SDEs can be used as generative models (trained as VAEs or GANs) for time-series Kidger et al. (2021). In principle, Neural SPDEs could be used in an analogous way as the generator of a generative model for spatio-temporal data, provided one can construct a sensible notion of discrepancy on the relevant space of distributions for the discriminator.

Neural rough differential equations (Neural RDEs) (Morrill et al., 2021) generalize Neural CDEs in that they allow us to relax the regularity assumption on $\boldsymbol{x}$ and consider a larger class of controls. In practice, Neural RDEs are particularly well suited for long time-series. This is due to the fact that model can be evaluated via a numerical scheme from stochastic analysis (called the *log-ODE method*) over intervals much larger than what would be expected given the sampling rate or length of the time-series. However, the space complexity of the numerical solver increases exponentially in the number of channels, thus model complexity becomes intractable for high-dimensional time-series.

6.4.2 Neural Operators

Neural Operators (Kovachki et al., 2021; Li et al., 2020b,c) are generalizations of neural networks that offer a way to model mappings between function spaces. Contrary to Neural CDEs, Neural Operators can be used to model mappings of the form $\boldsymbol{u}_0 \mapsto \boldsymbol{u}$, where $\boldsymbol{u}_0 : \mathcal{D} \to \mathbb{R}^{d_u}$ is the initial condition and $\boldsymbol{u} : \mathcal{D} \to \mathbb{R}^{d_u}$ is the solution of an underlying PDE. Contrary to prior work attempting to merge PDEs and deep learning techniques, Neural Operators do not require knowledge of the exact form of the underlying PDE, they are resolution-invariant and they achieve superior performance compared to previous physics-inspired deep learning models.

Given an initial condition $\boldsymbol{u}_0$, a Neural Operator with M layers is defined as follows

$$\boldsymbol{z}^0 = L_\theta(\boldsymbol{u}_0), \quad \boldsymbol{z}^k = N_\theta^k(\boldsymbol{z}^{k-1}), \quad \boldsymbol{u} = \Pi_\theta(\boldsymbol{z}^M),$$

for $k = 1, ..., M$, and with feedforward neural networks $L_\theta : \mathbb{R}^{d_u} \to \mathbb{R}^{d_h}$ and $\Pi_\theta : \mathbb{R}^{d_h} \to \mathbb{R}^{d_u}$. Each layer N_θ^k is a parametric operator defined for any $\boldsymbol{z} : \mathcal{D} \to \mathbb{R}^{d_h}$ as

$$N_\theta^k(\boldsymbol{z}) := \sigma\Big(A_\theta^k \boldsymbol{z} + b_\theta^k + \int_{\mathcal{D}} \mathcal{K}_\theta^k(\cdot, \mathbf{y})\boldsymbol{z}(\mathbf{y})d\mathbf{y}\Big) \tag{6.7}$$

where $A_\theta^k \in \mathbb{R}^{d_h \times d_h}$, $b_\theta^k \in \mathbb{R}^{d_h}$, $\mathcal{K}_\theta^k : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^{d_h \times d_h}$ is a matrix-valued kernel and σ is an activation function.

In the case of evolution-type PDEs, the solution $\boldsymbol{u} : [0, T] \times \mathcal{D} \to \mathbb{R}^{d_u}$ also depends on time. There are two ways of dealing with such time-dependent problems using Neural Operators. One can either consider $\widetilde{\mathcal{D}} = [0, T] \times \mathcal{D}$ as the new input domain or use an auto-regressive structure in time to model the mapping $\boldsymbol{z}(t_{i-1}, \cdot) \to \boldsymbol{z}(t_i, \cdot)$ according to eq. (6.7). This allows the use of Neural Operators for learning mappings of the form $\boldsymbol{\xi} \mapsto \boldsymbol{u}$, where $\boldsymbol{\xi} : [0, T] \times \mathcal{D} \to \mathbb{R}^{d_\xi}$ is a forcing term of the PDE. Among all kinds of Neural Operators, *Fourier Neural Operators* (FNOs) Li et al. (2020d) stand out because of their easier parametrization while maintaining similar learning performance. The *Deep Operator Network* (DeepONet) (Lu et al., 2021) is another popular model for learning operators on function spaces that we use as an additional benchmark. For details on this model see Appendix D.2.1.

Despite their ability to model operators mapping either the initial condition $\boldsymbol{u}_0$ or a time-dependent forcing $\boldsymbol{\xi}$ to the solution $\boldsymbol{u}$ of a PDE, Neural Operators offer no natural mechanism to learn solution operators mapping the pair $(\boldsymbol{u}_0, \boldsymbol{\xi})$ to the solution $\boldsymbol{u}$, which is crucial for solving SPDEs. Furthermore, as highlighted in the conclusion of Li et al. (2020d), in order to learn complex PDEs, Neural Operators require a large amount of training samples, which is a hard constraint for real-world applications as data generation is generally a very expensive procedure.

Neural CDE, Neural RDE, FNO, and DeepONet will be the main benchmark models in our experimental section on semilinear SPDEs. Moreover, we propose an additional hybrid baseline consisting of a Neural CDE where the drift is modelled by an FNO and the diffusion by a feedforward neural network. The motivation for using a FNO to represent the drift comes from the approximation properties of FNOs as detailed in Kovachki et al. (2021, Theorem 4). We call the resulting model Neural CDE-FNO.

6.5 Experiments

In this section, we conduct experiments on three semilinear SPDEs: the stochastic Ginzburg-Landau equation in Sec. 6.5.1, the stochastic Korteweg-De Vries equation in Sec. 6.5.2, and the stochastic Navier-Stokes equations in Sec. 6.5.3. In particular, we consider three supervised operator learning settings:

- $u_0 \mapsto u$, assuming the noise ξ is not observed;
- $\xi \mapsto u$, assuming the noise ξ is observed, but the initial condition u_0 is fixed across samples;
- $(u_0, \xi) \mapsto u$, assuming the noise ξ is observed and u_0 changes across samples.

We note that learning the operator $u_0 \mapsto u$ of an SPDE without observing the driving noise ξ unavoidably yields poor results as only partial information about the system is provided as input to the models. However, we find informative to include the performances obtained in this setting, as this provides a sanity check that emphasizes the importance of the noise in all the experiments. Moreover, the ability to process the initial condition u_0 on its own (in absence of noise) means that Neural SPDEs can be also used to learn deterministic PDEs.

For all the experiments, the loss function is the un-normalized pathwise L^2 loss. The hyperparameters for all the models were selected by grid search (see Appendix D.2.1 for further experimental details). The code for the experiments is provided as a supplementary material. We note that although the assumption of globally Lipschitz vector fields might be violated for the following SPDEs, well-posedness (i.e. existence of global solutions) can be shown using equation-specific arguments. Additional experiments may be found in Appendix D.2.

6.5.1 Stochastic Ginzburg-Landau equation

We start with a reaction diffusion equation in 1D called stochastic Ginzburg-Landau equation and given by

$$\begin{aligned}
\partial_t u - \Delta u &= 3u - u^3 + \xi, \\
u(t,0) &= u(t,1), \\
u(0,x) &= u_0(x), \quad (t,x) \in [0,T] \times [0,1]
\end{aligned}$$

This equation is also known as the Allen-Cahn equation in 1-dimension and is used to model various physical phenomena such as superconductivity (Temam, 2012). Here, ξ denotes space-time white noise with sample paths generated using classical sampling schemes for Wiener processes described in Appendix D.1.2.

We consider two data-regimes: a *low data regime* where the total number of training observations is $N = 1\,000$, and a *large data regime* where $N = 10\,000$. In both cases, the response paths are generated by solving the SPDE along each sample path of the noise ξ using a finite difference scheme described in

Table 6.1: **Ginzburg-Landau equation**. We report the relative L2 error on the test set. The symbol x indicates that the model is not applicable. *In the first task ($u_0 \mapsto u$) only partial information (u_0) is provided as input; the underlying noise ξ is used to generate the dynamics, it varies across samples, but is not provided as an input to the models.* This explains why the performance of the applicable models (DeepONet, FNO and NSPDE) is poorer than for all other settings. In the second setting, ($\xi \mapsto u$) the initial condition u_0 is kept fixed. In the third setting, the initial condition u_0 and the noise ξ change and are both provided as input.

Model	$N = 1\,000$			$N = 10\,000$		
	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
NCDE	x	0.112	0.127	x	0.056	0.072
NRDE	x	0.129	0.150	x	0.070	0.083
NCDE-FNO	x	0.071	0.066	x	0.066	0.069
DeepONet	0.130	0.126	x	0.126	0.061	x
FNO	0.128	0.032	x	0.126	0.027	x
NSPDE (Ours)	0.128	**0.009**	**0.012**	0.126	**0.006**	**0.006**

Appendix D.1.3 using 128 evenly distanced points in space and time and step size $\Delta t = 10^{-3}$. Following the same setup as in Chevyrev et al. (2021, equation (3.6)), we solve the SPDE until $T = 0.05$ resulting in 50 time points and choose as the initial condition $u_0(x) = x(1-x) + \kappa\eta(x)$, where

$$\eta(x) = a_0 + \sum_{k=-10}^{k=10} \frac{a_k}{1+|k|^2} \sin(k\pi x), \qquad \text{with } \ a_k \sim \mathcal{N}(0,1). \tag{6.8}$$

We either take $\kappa = 0$ or $\kappa = 0.1$ to generate a dataset in which the initial data is either fixed or varies across samples. We provide extra experiments on this SPDE for larger time horizons T and multiplicative forcing in Appendix D.2.2.

We report the results in Table 6.1. The Neural SPDE model (NSPDE) yields the lowest relative error for all tasks, reaching one order of magnitude improvement on the main task $(u_0, \xi) \mapsto u$ in the large data regime compared to all the applicable benchmark models (NCDE, NRDE, NCDE-FNO). In all settings, even with a limited amount of training samples ($N = 1\,000$), the NSPDE model is applicable and achieves $\sim 1\%$ error rate, and marginally improves to $< 1\%$ error when $N = 10\,000$.

6.5.2 Stochastic Korteweg–De Vries equation

Next, we consider a higher order SPDEs, namely the stochastic Korteweg–De Vries (KdV) equation given by

$$\begin{aligned} \partial_t u + \gamma \partial_x^3 u &= 6u\partial_x u + \xi, \\ u(t,0) &= u(t,1), \\ u(0,x) &= u_0(x), \quad (t,x) \in [0,T] \times [0,1] . \end{aligned}$$

This equation is used to describe the propagation of nonlinear waves at the surface of a fluid subject to random perturbations (another wave equation is studied in Appendix D.2.3). We refer the reader to Wazwaz (2009) for an overview of the KdV equation and its relation to solitary waves.

The stochastic forcing is given by $\xi = \dot{W}$ for W being a partial sum approximation of a Q-Wiener process as per Example 10.8 in Lord et al. (2014) with $\lambda_j \sim j^{-5+\varepsilon}$ and $\phi_j(x) = \sin(j\pi x)$ (see Equation (D.1) in Appendix D.1.2). Taking small $\varepsilon > 0$ guarantees that W_t is twice differentiable in space for every $t \geq 0$. To generate the datasets, we solve the SPDE with $\gamma = 0.1$ until $T = 0.5$. The stochastic forcing is simulated using 128 evenly distanced points in space and a time step $\Delta t_{\text{ref}} = 10^{-3}$. We then approximate realizations of the solution of the KdV equation using a time step $\Delta t = 10^{-2}$ to $T = 0.5$. Here, the initial condition is given by $u_0(x) = \sin(2\pi x) + \kappa\eta(x)$, where η is defined in Equation (6.8). Similarly to Sec. 6.5.1 we either take $\kappa = 0$ or $\kappa = 1$ to generate datasets where the initial condition is either fixed or varies across samples. Each dataset consists of $N = 1\,000$ training observations.

As reported in Table 6.2, Neural SPDEs outperforms the second best model FNO by a full order of magnitude in the task $\xi \mapsto u$ and the second best model NCDE-FNO by almost two orders of magnitude in the task $(u_0, \xi) \mapsto u$. We also perform the same tasks for a larger time horizons $T = 1$ and report the results of a comparison against FNO in Table 6.3.

6.5.3 Stochastic Navier-Stokes equations

Finally, we consider the vorticity form of the Navier-Stokes equations in 2-dimensions for an incompressible flow,

$$\begin{aligned} \partial_t w - \nu \Delta w &= -\boldsymbol{v} \cdot \nabla w + f + \sigma\xi \\ w(0,\mathbf{x}) &= w_0(\mathbf{x}), \quad (t,\mathbf{x}) \in [0,T] \times [0,1]^2 \end{aligned} \tag{6.9}$$

where $\boldsymbol{v}$ is the unique divergence-free ($\nabla \cdot \boldsymbol{v} = 0$) velocity field such that $w = \nabla \times \boldsymbol{v}$. The Navier-Stokes equations describe the motion of incompressible fluid with viscosity ν subject to the external forces (Temam, 2012). The

Table 6.2: **Stochastic KdV**. We report the relative L2 error on the test set. The symbol x indicates that the model is not applicable. N is fixed to 1 000 and $T = 0.5$.

Model	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
NCDE	x	0.464	0.466
NRDE	x	0.497	0.503
NCDE-FNO	x	0.126	0.259
DeepONet	0.874	0.235	x
FNO	0.835	0.079	x
NSPDE (Ours)	0.832	**0.004**	**0.008**

Table 6.3: **Stochastic KdV** with longer time horizon $T = 1$.

Model	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
FNO	0.913	0.112	x
NSPDE (Ours)	0.904	**0.009**	0.012

deterministic forcing f is a function of space only and is defined as in Li et al. (2020d). The stochastic forcing ξ is given by $\xi = \dot{W}$ for W being a Q-Wiener process which is colored in space and rescaled by $\sigma = 0.05$ (see Appendix D.1.2). The initial condition w_0 is generated according to $w_0 \sim \mathcal{N}(0, 3^{3/2}(-\Delta + 49I)^{-3})$ with periodic boundary conditions. The viscosity parameter is set to $\nu = 10^{-4}$.

For each realization of the Q-Wiener process (sampled according to the scheme in Appendix D.1.2) we solve Equation (6.9) with a pseudospectral solver described in Appendix D.1.3, where the time is advanced with a Crank–Nicolson update. We solve the SPDE on a 64×64 mesh in space and use a time step of size 10^{-3}. For the tasks $u_0 \mapsto u$ and $\xi \mapsto u$, we generate the datasets by solving the SPDE up to time $T = 1$ and downsample the trajectories by a factor of 10 in time (resulting in 100 time steps) and 4 in space (resulting in a 16×16 spatial resolution). The number of training samples is $N = 1\,000$. To generate the training set for the task $(u_0, \xi) \mapsto u$, we generate 10 long trajectories of 15 000 steps each up to time $T = 15$. We partition each of these 10 trajectories into consecutive subtrajectories of 500 time steps using a rolling window. This procedure yields a total of 2 000 input-output pairs. We chose to split the data into shorter sequences of 500 time steps so that one training batch could fit into memory on a Tesla P100 NVIDIA GPU.

We report the results in Table 6.4. Neural SPDE marginally outperforms FNO in the simpler task $\xi \mapsto u$, but with a significantly larger performance gap

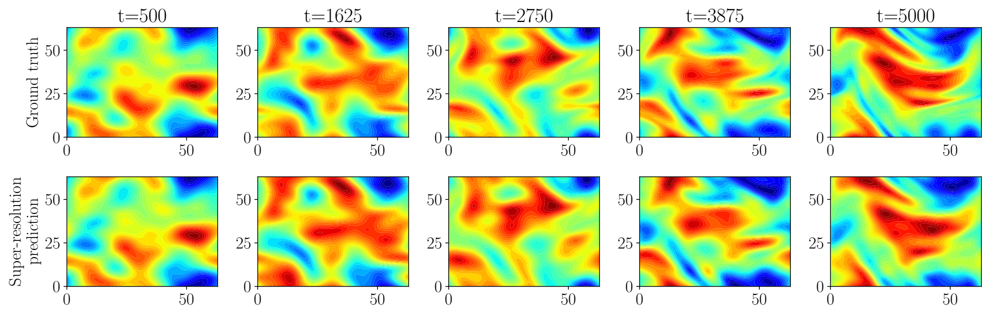


Figure 6.2: *Top panel*: Solution of the vorticity equation for one realization of the stochastic forcing between the 500th and the 5 000th time steps. *Bottom panel*: Predictions with the Neural SPDE model given the initial condition at the 500th time step and the forcing between the 500th and the 5 000th time steps. The model is trained on a 16×16 mesh and evaluated on a 64×64 mesh.

Table 6.4: **Stochastic Navier-Stokes equations in two dimensions**. We report the relative L2 error on the test set. The symbol x indicates that the model is not applicable. The symbol - indicates that the model does not fit in memory.

Model	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
NCDE	x	0.366	0.843
NRDE	x	-	-
NCDE-FNO	x	0.326	0.178
DeepONet	0.432	0.348	x
FNO	0.188	0.039	x
NSPDE (Ours)	0.155	**0.034**	**0.049**

in the harder task $(u_0, \xi) \mapsto u$ with the second-best model NCDE-FNO. Fig. 6.2 indicates that the Neural SPDE model is capable of zero-shot super-resolution in space and in time, achieving good performance even when evaluated on a larger time horizon of 5 000 time steps and on an upsampled 64×64 spatial grid. Finally, we report in Table 6.5 the ratios of inference time of the trained Neural SPDEs over the runtime of the solvers, indicating that NSPDEs can be up to 3 orders of magnitude faster than traditional numerical solvers.

6.6 Conclusion

We introduced the Neural SPDE model to learn solution operators of PDEs with (possibly stochastic) forcing from partially observed data. The proposed approach provides an extension to two popular classes of physics-inspired models. The model extends Neural CDEs, SDEs, RDEs in that it is resolution-invariant both in space and in time. It also extends Neural Operators as it can learn

Table 6.5: Ratio of the inference time of NSPDE (once trained) over the runtime of the numerical solver for different SPDEs. In each case we consider a single solution. We use the same space and time discretization for both NSPDE and the numerical solver.

Dataset	Speedup
Stochastic Ginzburg-Landau Equation	$59\times$
Stochastic Korteweg-De Vries Equation	$80\times$
Stochastic Navier-Stokes	$300\times$

solution operators of SPDEs depending simultaneously on the initial condition and a realization of the driving noise. We performed extensive experiments on various semilinear SPDEs, including the stochastic Ginzburg-Landau, Korteweg-De Vries and Navier-Stokes equations. The outcomes illustrate that the model achieves superior performance while requiring a lower amount of training data compared to alternative models, and its evaluation is up to 3 orders of magnitude faster than traditional numerical solvers.

Chapter 7

Discussion

In this thesis, we leveraged mathematical tools from rough paths analysis, the theory of reproducing kernels and neural differential equations (reviewed in Chapter 2), to develop new methodologies addressing three learning problems on sequential data (in Chapters 3 to 6): large scale Bayesian inference for GP models indexed on sequences, learning from sets of multivariate time-series, and modelling the relationships between systems evolving in space and time.

7.1 Contributions of the thesis

We began in Chapter 3 by addressing the prohibitive computational cost of inference algorithms for GPs on sequential data. More precisely, we considered models in which the signature kernel defines the covariance function of the GP prior. Subsequently, we leveraged two properties of the signature kernel to develop a fast approximate inference scheme based on variational inference. First, we used the explicit feature expansion of the signature kernel to introduce an evidence lower bound which can be computed without costly matrix inversions. Second, we derived a PDE to efficiently compute the gradients of the signature kernel for gradient-based optimization of the evidence lower bound. Altogether, this allowed to fit Gaussian process models and perform model selection on large datasets of multivariate time-series which were previously intractable.

In Chapter 4 we showed how the signature features (respectively, the signature kernel) can be used to construct a set of basis functions (respectively, kernel functions) for regression analysis on probability measures corresponding to the laws of stochastic processes (DR). The mathematical properties of the expected signature allowed us to prove the universal approximation property of the newly introduced features and the universality of the DR kernel. We introduced two learning algorithms to conduct DR on discrete (or empirical) probability measures, representing sets of multivariate time-series. In Chapter 5 we built upon the results of Chapter 4 and the notion of conditional kernel mean embeddings from the theory of RKHSs, to construct a more expressive kernel on stochastic processes, which captures information that goes beyond their law. The kernel trick for evaluating the signature kernel and the inner products of conditional KME was instrumental in developing practical algorithms for applications ranging from DR to kernel two-sample tests.

In Chapter 6 we introduced the Neural SPDE model, which extends NCDEs to model the functional relationship between discretely observed spatio-temporal signals. To this aim, we blended SPDEs, an important class of models in physics, with the high expressivity and flexibility of deep learning models. Leveraging the notion of mild solution of an SPDE, we formulated the Neural SPDE model directly at the level of functional input and output spaces. We empirically demonstrated the capability of the model to be deployed on a finer mesh resolution than the data it has been trained with. We also showed that a learned Neural SPDE model can be used as a fast surrogate for traditional numerical solvers.

7.2 Future work

Scalability As highlighted in Chapter 4, a key challenge is to develop techniques which scale well with both the number of training examples and the dimensionality of the inputs. While the signature kernel trick allows to avoid the cost of the explicit computation of the coordinates of the signature, thereby enabling to handle high-dimensional time-series, methods that operate on the kernel matrix of the data scale poorly with the number of training examples. Various approaches proposed in the context of static data (Pham and Pagh, 2013; Rahimi and Recht, 2007) could be adapted to the signature kernel. However, this would also require understanding theoretically the approximation quality of these methods when extended to the sequential setting. While the approximate inference scheme developed in Chapter 3 improves the scalability with respect to the size of the training dataset, the number of inducing variables increases exponentially with the state space dimension of the input time-series. Future work should focus on improving the scaling in the state space dimension.

Hyperparameter selection Most methodologies developed in this thesis make use of the signature kernel, whose parameters need to be adjusted for best performance. In Chapter 3, kernel selection was performed by ML-II, a Bayesian model selection method. In the setting of two-sample tests (Chapter 5) one can leverage various hyperparameter selection methods which have been proposed in the kernel literature, including approaches aiming at maximizing the test power using the signal-to-noise-ratio as an objective Gretton et al. (2012b); Liu et al. (2020); Sutherland et al. (2017). However, in the context of

DR, future work is needed for tuning the hyperparemeters of the higher order KMEs, without resorting to expensive cross-validation procedures. Flaxman et al. (2016); Hsu et al. (2018) are certainly a good starting point for such an investigation.

Generative modelling The higher order KMEs introduced in Chapter 5 have the potential to be used beyond two-sample tests and distribution regression. For example, higher order MMDs could be used as discriminators in generative models for time-series. The Neural SPDE model developed in Chapter 6 also has the potential to be used in the context of generative models, as it may be used to parameterize a generator network. However, this direction requires constructing a discrepancy between probability measures on spatio-temporal signals, generalizing for example the signature kernel MMD in Salvi et al. (2021). Neural SPDEs paired with such discrepancy would allow the design of new generative models for spatio-temporal signals generalizing the results in Kidger et al. (2021).

Appendix A

Appendix to Chapter 3

A.1 Additional proof

In this section we prove Thm. 3.4.1 which underpins Alg. 4, an algorithm for computing the gradients of the signature kernel with respect to its input paths, without backprogating through the operations of the PDE solver. For the reader's convenience the theorem is recalled below.

Theorem 4.1. *For any $\boldsymbol{\gamma} \in \mathcal{X}(\mathbb{R}^d)$ the directional derivative $K_{\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y})$ of the signature kernel along the path $\boldsymbol{\gamma}$ satisfies the following relation*

$$K_{\boldsymbol{\gamma}}(\boldsymbol{x}, \boldsymbol{y}) = \int_0^T \int_0^T u(s,t)\widetilde{u}(T-s, T-t)\langle \dot{\boldsymbol{\gamma}}(s), \dot{\boldsymbol{y}}(t)\rangle dsdt,$$

where $\widetilde{u}(s,t) = K(\overleftarrow{\boldsymbol{x}}|_{[0,s]}, \overleftarrow{\boldsymbol{y}}|_{[0,t]})$ and where $\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}}$ are respectively the paths $\boldsymbol{x}, \boldsymbol{y}$ reversed in time.

Before proving the theorem we need the following important lemma.

Lemma A.1.1. *For any two paths continuous paths of bounded variation $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}(\mathbb{R}^d)$ the signature kernel satisfies the following relation*

$$K(\boldsymbol{x}, \boldsymbol{y}) = K(\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}})$$

where $\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}}$ are the respectively $\boldsymbol{x}, \boldsymbol{y}$ reversed in time.

Proof. The subset $\{\mathcal{A} \in \mathscr{T}(\mathbb{R}^d) \mid A_0 = 1\}$ of the tensor algebra $\mathscr{T}(\mathbb{R}^d)$ is a group, and it is a standard result in rough path theory (see for example Lyons et al. (2007)) that $S(\overleftarrow{\boldsymbol{x}}) = S(\boldsymbol{x})^{-1}$. The operator $g : S(\boldsymbol{x}) \mapsto S(\boldsymbol{x})^{-1}$ reverses the order of the indices in the multi-index $(i_1, \ldots, i_k)$ and multiplies the result $S^{i_k \cdots i_1}(x)$ by -1 if the length k of the multi-index is odd. Expanding out $K(\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}})$ coordinate-wise it is easy to see that the two -1's for multi-indices of odd length cancel as multiplied together, therefore the expansion of $K(\boldsymbol{x}, \boldsymbol{y})$ matches the one of $K(\overleftarrow{\boldsymbol{x}}, \overleftarrow{\boldsymbol{y}})$. $\square$

We also recall the notation for the signature kernel and its directional derivative used in the statement of Thm. 3.4.1 in Chapter 3:

$$u(s,t) := K\left(\boldsymbol{x}|_{[0,s]}, \boldsymbol{y}|_{[0,t]}\right)$$
$$u_{\boldsymbol{\gamma}}(s,t) := K_{\boldsymbol{\gamma}}\left(\boldsymbol{x}|_{[0,s]}, \boldsymbol{y}|_{[0,t]}\right).$$

Proof of Thm. 3.4.1. Let $\boldsymbol{\gamma} : [0,T] \to \mathbb{R}^d$ 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

$$u_{\boldsymbol{\gamma}}(s,t) = \int_0^s \int_0^t A_{s,t}(p,q) u(p,q) \langle \dot{\boldsymbol{\gamma}}(p), \dot{\boldsymbol{y}}(q) \rangle dp dq. \tag{A.1}$$

Differentiating $u_{\boldsymbol{\gamma}}$ with respect to s and t we get

$$\begin{aligned}\frac{\partial^2 u_{\boldsymbol{\gamma}}}{\partial s \partial t} = &\int_0^s \int_0^t \frac{\partial^2 A_{s,t}(p,q)}{\partial s \partial t} u(p,q) \langle \dot{\boldsymbol{\gamma}}(p), \dot{\boldsymbol{y}}(q) \rangle dp dq \\ &+ A_{s,t}(s,t) u(s,t) \langle \dot{\boldsymbol{\gamma}}(s), \dot{\boldsymbol{y}}(t) \rangle.\end{aligned} \tag{A.2}$$

By eq. (3.19) in Chapter 3 we know that the directional derivative of the signature kernel along the path $\boldsymbol{\gamma}$ solves the following PDE,

$$\frac{\partial^2 u_{\boldsymbol{\gamma}}}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle u_{\boldsymbol{\gamma}} + \langle \dot{\boldsymbol{\gamma}}(s), \dot{\boldsymbol{y}}(t) \rangle u. \tag{A.3}$$

Equating eq. (A.2) and eq. (A.3) we deduce that $A_{s,t}(s,t) = 1$ and

$$\begin{aligned}&\int_0^s \int_0^t \frac{\partial^2 A_{s,t}(p,q)}{\partial s \partial t} u(p,q) \langle \dot{\boldsymbol{\gamma}}(p), \dot{\boldsymbol{y}}(q) \rangle dp dq \\ &= \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle u_{\boldsymbol{\gamma}}(s,t) \\ &= \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle \int_0^s \int_0^t A_{s,t}(p,q) u(p,q) \langle \dot{\boldsymbol{\gamma}}(p), \dot{\boldsymbol{y}}(q) \rangle dp dq.\end{aligned}$$

Which implies that,

$$\frac{\partial^2 A_{s,t}(p,q)}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle A_{s,t}(p,q) \tag{A.4}$$

or equivalently, by integrating with respect to s and t,

$$A_{s,t}(p,q) = 1 + \int_p^s \int_q^t A_{s',t'}(p,q) \langle \dot{\boldsymbol{x}}(s'), \dot{\boldsymbol{y}}(t') \rangle ds' dt'.$$

Hence, we have:

$$\begin{aligned} A_{T,T}(p,q) &= \left\langle S(\boldsymbol{x})_{[p,T]}, S(\boldsymbol{y})_{[q,T]} \right\rangle \\ &= K\left(\overleftarrow{\boldsymbol{x}}|_{[0,T-p]}, \overleftarrow{\boldsymbol{y}}|_{[0,T-q]} \right), \end{aligned}$$

where the last equality is a consequence of Lemma A.1.1. Pluging back this result into eq. (A.1) concludes the proof. □

A.2 Additional experimental details

In this section we describe the experimental setup for Sec. 3.6. All experiments were conducted on NVIDIA Tesla P100 GPUs.

A.2.1 Data collection process

The classification tasks of Sections 3.6.1 and 3.6.2 were performed on two datasets (PenDigits, RightWhaleCalls) from the UCR & UEA time-series classification repository.[1] For the large scale classification experiment of Sec. 3.6.3 we used a dataset of 1M satellite time-series (STS).[2] Lastly, the climatic data (WeatherForecast) for rainfall prediction task in Sec. 3.6.4 were downloaded from the Max Planck Institute for Biogeochemistry website.[3]

Data pre-processing included the following two steps. As explained in Sec. 3.3.1, we first add a monotonically increasing coordinate to all multivariate time-series that we call "time", which effectively augments by one the number of channels. Then, we standard scale the time-series using the tslearn library (Tavenard et al., 2020). This is particularly important for the WeatherForecast dataset where channels have different scales. Additional processing steps have been performed for two datasets (RightWhaleCalls, WeatherForecast) which we treat separately next.

A standard data transformation to tackle classification tasks on audio signals consists in computing their *spectrograms*. We follow this procedure for the RightWhaleCalls dataset which contains univariate highly-oscillatory time-series of length 2 000. We used the scipy Python library to do so. The spectrogram is commonly represented as a graph with one axis representing time, the other axis representing the frequency, and the color intensity representing the amplitude of a particular frequency at a particular time. We consider the spectrogram as a multivariate time-series, where each channel represents the change in amplitude of a particular frequency over time. Furthermore, exploiting the fact that frequencies in whale call signals are typically between

[1] `https://timeseriesclassification.com`
[2] `https://cloudstor.aarnet.edu.au/plus/index.php/s/pRLVtQyNhxDdCoM`
[3] `https://www.bgc-jena.mpg.de/wetter/weather_data.html`

50 and 300Hz, we only consider frequencies which fall within this range. As a result we obtain 28-dimensional time-series each of length 30. We then apply the pre-processing steps described above.

To create the WeatherForecast dataset we used the recordings of various climatic variables in two weather stations located in Germany from 2004 to 2020. The outliers were filtered out, and we used the recordings of 7 variables (depicted on Fig. 3.2) over 6 hours in order to predict whether it would rain by more than 1mm over the next hour. There is one recording every 10min resulting in input time-series of length $\ell = 36$. Since there were much fewer positive cases (raining) than negative cases (not raining), we dropped at random a fraction of the data, such that the ratio of positive/negative examples is brought down to 3.

A.2.2 Training procedure

The datasets for classification of sequential digits (PenDigits), audio signals (RightWhaleCalls), and satellite time-series (STS) come with a predefined test-train split. In order to report standard deviations on our results we subsampled 20% (PenDigits,RightWhaleCalls) or 2% (STS) of the training set to form a validation set.

The training was equally split into 3 different phases. During the first phase, only the variational parameters are trained. For the second phase, both the variational parameters and the hyperparameters of the kernel are trained. During the last phase the variational parameters are trained on the full training set (the validation data being merged back). Overall, the hyperparameters are fixed for two-third of the iterations. SigGPDE and the GPSig-IT/IS baselines have the same set of hyperparameters, which correspond to the scaling factors for each channel for the ARD parametrization of the signature kernel (see Sec. 3.3.3). Those were initialized with the same value for all models. The inducing tensors for GPSig-IT and inducing sequences for GPSig-IS were initialized following the procedure outlined in Toth and Oberhauser (2020). We recall that for SigGPDE there is no such parameters to initialize. As recommended in Toth and Oberhauser (2020), we use a truncation level of $n = 4$ for their signature kernel algorithm (GPSig-IT/IS).

The minibatch size is either 50 (PenDigits, RightWhaleCalls) or 200 (STS). We used the Nadam optimizer (Dozat, 2016) with learning rate 10^{-3}. In Chapter 3 we report the time per iteration which corresponds to one minibatch.

A.3 Additional algorithmic details

In this section we start by outlining the space and time complexities of the algorithms underlying SigGPDE. Then, we explain how we have developed a dedicated CUDA TensorFlow operator for GPU acceleration to speed-up the computation of the signature kernel and its gradients.

A.3.1 Complexity analysis

The main algorithms underpinning SigGPDE consist in computing three different covariance matrices to evaluate the ELBO. These are the covariance matrix between the inducing variables $\mathbf{u}$ (denoted by $\mathbf{C_{uu}}$), between the marginal $\mathbf{f}$ and the inducing variables (denoted by $\mathbf{C_{fu}}$), and finally the covariance matrix of $\mathbf{f}$ (its diagonal is denoted by $\mathrm{diag}(\mathbf{C_{ff}})$). In Tables A.1 and A.2 we compare the time and space complexities for the corresponding SigGPDE algorithms to those of GPSig-IT/IS.

In the SigGPDE sparse variational inference framework, $\mathbf{C_{uu}}$ is diagonal which lowers both the memory and computational costs (first line in Tables A.1 and A.2). Besides there is no need to compute the Cholesky decomposition of $\mathbf{C_{uu}}$ to invert it (see last line Table A.1). Lastly, in SigGPDE the inducing variables do not depend on any variational parameter (see last line Table A.2).

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$\mathbf{C_{uu}}$	$\mathcal{O}(1)$	$\mathcal{O}(n^2M^2d)$	$\mathcal{O}((n+d)M^2\tilde{\ell}^2)$
$\mathbf{C_{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell d)$	$\mathcal{O}((n+d)\tilde{N}M\ell\tilde{\ell})$
$\mathrm{diag}(\mathbf{C_{ff}})$	$\mathcal{O}(d\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$
Lin. Alg.	$\mathcal{O}(\tilde{N}M^2)$	$\mathcal{O}(\tilde{N}M^2+M^3)$	$\mathcal{O}(\tilde{N}M^2+M^3)$

Table A.1: Comparison of time complexities. M is the number of inducing variables, $\tilde{N}$ the batch size, d the number of channels in the time-series, ℓ the length of the sequences, n the truncation level (for GPSig-IT and GPSig-IS) and $\tilde{\ell}$ the length of the inducing sequences.

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$\mathbf{C_{uu}}$	N/A	$\mathcal{O}(n^2M^2)$	$\mathcal{O}(M^2\tilde{\ell}^2)$
$\mathbf{C_{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell)$	$\mathcal{O}(\tilde{N}M\ell\tilde{\ell})$
$\mathrm{diag}(\mathbf{C_{ff}})$	$\mathcal{O}(\tilde{N}\ell^2)$	$\mathcal{O}(\tilde{N}\ell^2)$	$\mathcal{O}(\tilde{N}\ell^2)$
$\mathbf{z}$	N/A	$\mathcal{O}(n^2Md)$	$\mathcal{O}(M\tilde{\ell}d)$

Table A.2: Comparison of space complexities, separated by algorithm to compute each covariance matrix. The last line accounts for the storage of the inducing tensors and inducing sequences in GPSig-IT and GPSig-IS.

A.3.2 Computing the signature kernel and its gradients

Recall that the signature kernel solves the following PDE

$$\frac{\partial^2 u}{\partial s \partial t} = \langle \dot{\boldsymbol{x}}(s), \dot{\boldsymbol{y}}(t) \rangle u \qquad u(0,\cdot) = 1,\ u(\cdot,0) = 1, \tag{A.5}$$

therefore each kernel evaluation amounts to a call to a PDE solver. Using a straightforward implementation of a finite difference PDE solver which consists in applying an update of the form (as explained in more details in Sec. 2.2.1)

$$u(s_i, t_j) = f_{\text{upd}}(u(s_{i-1}, t_{j-1}), u(s_i, t_{j-1}), u(s_{i-1}, t_j)),$$

in row or column order, the time complexity for N kernel evaluations for time-series with d channels of length ℓ is $\mathcal{O}(dN\ell^2)$. Indeed, there is no data dependencies between each of the N kernel evaluations, hence we can solve each PDE in parallel. But, this does not reduce the quadratic complexity with respect to the length ℓ. 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 with in parallel as there is no data dependency between them. Therefore, we propose a CUDA implementation where N collections of $2\ell - 1$ threads (the number of cells on the biggest antidiagonal) running in parallel can simultaneously update an antidiagonal of the solution grids.

To compute the gradients, we use the result from Thm. 3.4.1. During the forward pass we solve the PDEs defined by the input time-series using the CUDA operator described above. For the backward pass, we first solve the PDEs with the input time-series reversed in time, by calling the same CUDA operator. Second, we compute the gradients using simple vectorized TensorFlow operations.

Appendix B

Appendix to Chapter 4

B.1 Proofs

In this section, we prove that the expected signature $\bar{S}$ is weakly continuous (Appendix B.1.1), and that the pathwise expected signature Φ is injective and weakly continuous (Appendix B.1.2).

Recall that in Chapter 4 we considered a compact subset of paths $U \subset \mathcal{X}(E)$, where E is a Banach space of finite dimension $d > 1$, and $\mathcal{X}(E)$ as defined in eq. (2.4) such that $(\mathcal{X}(E), \|\cdot\|)$ is a Banach space with $\|\boldsymbol{x}\| = \mathcal{V}(\boldsymbol{x}) + \sup_{t\in[0,T]} \|\boldsymbol{x}(t)\|_E$ for all $\boldsymbol{x} \in \mathcal{X}(E)$. We will denote by $\mathcal{P}(U)$ the set of Borel probability measures on U and by $S(U) \subset H(E)$ the image of U by the signature $S : \mathcal{X}(E) \to H(E)$.

As shown in (Chevyrev and Oberhauser, 2018, Section 3), if E is a Hilbert space with inner product $\langle\cdot,\cdot\rangle_E$, then for any $k \geq 1$ the following bilinear form defines an inner product on $E^{\otimes k}$,

$$\left\langle \mathbf{e}_{i_1} \otimes \ldots \otimes \mathbf{e}_{i_k}, \mathbf{e}_{j_1} \otimes \ldots \otimes \mathbf{e}_{j_k} \right\rangle_{E^{\otimes k}} = \prod_{r=1}^{k} \delta_{i_r,j_r}, \qquad \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{B.1}$$

which extends by linearity to an inner product $\langle \mathcal{A}, \mathcal{B} \rangle_{H(E)} = \sum_{k=0}^{\infty} \langle A_k, B_k \rangle_{E^{\otimes k}}$ on $H(E)$ that thus becomes also a Hilbert space.

B.1.1 Weak continuity of the expected signature

Definition 17. *A sequence of probability measures $\mu_n \in \mathcal{P}(U)$ converges weakly to μ if for every $f \in C_b(U, \mathbb{R})$ we have $\int_U f d\mu_n \to \int_U f d\mu$ as $n \to \infty$, where $C_b(U, \mathbb{R})$ is the space of real-valued continuous bounded functions on U.*

Remark. *Since U is a compact metric space, we can drop the word "bounded" in Def. 17.*

Definition 18. *Given two probability measures $\mu, \nu \in \mathcal{P}(U)$, the Wasserstein-1* *distance is defined as follows*

$$W_1(\mu,\nu) = \inf_{\gamma\in\Pi(\mu,\nu)} \int_{\boldsymbol{x},\boldsymbol{y}\in U} \|\boldsymbol{x}-\boldsymbol{y}\|\,\gamma(d(\boldsymbol{x},\boldsymbol{y})) \tag{B.2}$$

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

Lemma B.1.1. *(Chevyrev and Oberhauser, 2018, Theorem 5.3) The signature $S : \mathcal{X}(E) \to H(E)$ is injective.*[1]

Lemma B.1.2. *(Chevyrev et al., 2016, Corollary 5.5) The signature is continuous w.r.t. $\|\cdot\|$.*

Lemma B.1.3. *(Chevyrev and Oberhauser, 2018, Theorem 5.6) The expected signature $\bar{S} : \mathcal{P}(U) \to H(E)$ is injective.*

This result was firstly proved in Fawcett (2002) for probability measures supported on compact subsets U of $\mathcal{X}(E)$, which is enough for this thesis. It was also proved in a more abstract setting in Chevyrev et al. (2016). The authors of Chevyrev and Oberhauser (2018) introduce a normalization that is not needed in case of compact supports, as they mention in (Chevyrev and Oberhauser, 2018, (I) - page 2).

Theorem B.1.1. *The expected signature $\bar{S} : \mathcal{P}(U) \to H(E)$ is weakly continuous.*

Proof. Consider a sequence $\{\mu_n\}_{n\in\mathbb{N}}$ of probability measures on $\mathcal{P}(U)$ converging weakly to a measure $\mu \in \mathcal{P}(U)$. By Lemma B.1.2, the signature map $S : \boldsymbol{x} \mapsto S(\boldsymbol{x})$ is continuous w.r.t. $\|\cdot\|$. Hence, by definition of weak-convergence (and because U is compact), for any $k > 0$ and any multi-index $(i_1, \dots, i_k) \in \{1, \dots, d\}^k$ it follows that

$$\int_{\boldsymbol{x}\in U} S^{i_1\dots i_k}(\boldsymbol{x})\mu_n(d\boldsymbol{x}) \to \int_{\boldsymbol{x}\in U} S^{i_1\dots i_k}(\boldsymbol{x})\mu(d\boldsymbol{x}).$$

The factorial decay given by Lyons et al. (2007, Proposition 2.2) yields

$$\int_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x})\mu_n(d\boldsymbol{x}) \to \int_{\boldsymbol{x}\in\mathcal{X}} S(\boldsymbol{x})\mu(d\boldsymbol{x}),$$

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

B.1.2 Injectivity and weak continuity of the pathwise expected signature

Theorem B.1.2. *(Lyons et al., 2007, Theorem 3.7) Let $\boldsymbol{x} \in \mathcal{X}(E)$ and recall the definition of the projection $\Pi_t : \boldsymbol{x} \mapsto \boldsymbol{x}|_{[0,t]}$. Then, the $H(E)$-valued path*

[1] Up to tree-like equivalence (see (Chevyrev and Oberhauser, 2018, appendix B) for a definition and detailed discussion).

defined by

$$S_{path}(\boldsymbol{x}) : t \mapsto S \circ \Pi_t(\boldsymbol{x}), \tag{B.3}$$

is continuous and of bounded variation. Furthermore the map $\boldsymbol{x} \mapsto S_{path}(\boldsymbol{x})$ is continuous w.r.t. $\|\cdot\|$.

For any $\mu \in \mathcal{P}(U)$ the path $\Phi(\mu) \in \mathcal{X}(H(E))$. Indeed $\Phi(\mu)$ is a continuous path because $\boldsymbol{x}$, Π_t, S and Φ are all continuous and the composition of continuous functions is continuous. The fact that Φ is of bounded variation comes from the fact that $\|\Phi(\mu)\| \leq \mu(U) \sup_{\boldsymbol{x} \in U} \|S_{path}(\boldsymbol{x})\| < +\infty$ by Thm. B.1.2.

Theorem B.1.3. *The pathwise expected signature $\Phi : \mathcal{P}(U) \to \mathcal{X}(H(E))$ is injective.*

Proof. Let $\mu, \nu \in \mathcal{P}(\mathcal{X})$ be two probability measures. If $\Phi(\mu) = \Phi(\nu)$, then for any $t \in [0, T]$, $\mathbb{E}_{X \sim \mu}[S \circ \Pi_t(X)] = \mathbb{E}_{Y \sim \nu}[S \circ \Pi_t(Y)]$. In particular, for $t = T$

$$\bar{S}(\mu) = \mathbb{E}_{X \sim \mu}[S(X)] = \mathbb{E}_{Y \sim \nu}[S(Y)] = \bar{S}(\nu).$$

The result follows from the injectivity of the expected signature $\bar{S}$ (Lemma B.1.3). □

Theorem B.1.4. *The pathwise expected signature $\Phi : \mathcal{P}(U) \to \mathcal{X}(H(E))$ is weakly continuous.*

Proof. Let $\{\mu_n\}_{n \in \mathbb{N}}$ be a sequence in $\mathcal{P}(U)$ converging weakly to $\mu \in \mathcal{P}(U)$. As S_{path} is continuous (Thm. B.1.2), 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 U is compact, it follows that the image $S_{path}(U)$ is a compact subset of the Banach space $\mathcal{X}(H(E))$. By (Villani, 2008, Theorem 6.8) weak convergence of probability measures on compact supports is equivalent to convergence in *Wasserstein-1 distance*. By *Jensen's inequality*

$$\|\mathbb{E}[S_{path}\#\mu_n] - \mathbb{E}[S_{path}\#\mu]\| \leq \mathbb{E}[\|S_{path}\#\mu_n - S_{path}\#\mu\|].$$

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,

$$\|\mathbb{E}[S_{path}\#\mu_n] - \mathbb{E}[S_{path}\#\mu]\| \leq W_1(S_{path}\#\mu_n, S_{path}\#\mu) \to 0$$

which yields the convergence,

$$\mathbb{E}[S_{path}\#\mu_n] \to \mathbb{E}[S_{path}\#\mu]$$

in $\|\cdot\|$ over $\mathcal{X}(H(E))$. Noting that $\mathbb{E}[S_{path}\#\mu] = \Phi(\mu)$ concludes the proof. □

B.2 Experimental details

In our experiments we benchmark KES and SES against DeepSets and DR-k where $k \in \{\mathrm{RBF}, \mathrm{Matern32}, \mathrm{GA}\}$. Apart from DR-GA, all other baselines are designed to operate on vectorial data. Therefore, in order to deploy them in the setting of DR on sequential data, manual pre-processing (such as padding) is required. In the next section we describe how we turn discrete time-series into continuous paths on which the signature operates.

B.2.1 Transforming discrete time-series into continuous paths

Consider a d-dimensional time-series of the form $\underline{\mathbf{x}} = \{(t_1, \mathbf{x}_1), \ldots, (t_\ell, \mathbf{x}_\ell)\}$ with time-stamps $0 = t_1 < \ldots < t_\ell = T$ and values $\mathbf{x}_k \in \mathbb{R}^d$, and the continuous path $\boldsymbol{x}$ obtained by linearly interpolating between the points $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$. The signature (truncated at level n) of $\boldsymbol{x}$ can be computed explicitly with existing Python packages (Kidger and Lyons, 2020; Lyons, 2010; Reizenstein and Graham, 2018), does not depend on the time-stamps $(t_1, \ldots, t_\ell)$, and produces $(d^{n+1} - 1)/(d-1)$ terms when $d > 1$. When $d = 1$ the signature is trivial since

$$S(\boldsymbol{x}) = \left(1,\ (\mathbf{x}_\ell - \mathbf{x}_1),\ \frac{1}{2}(\mathbf{x}_\ell - \mathbf{x}_1)^2,\ \ldots,\ \frac{1}{n!}(\mathbf{x}_\ell - \mathbf{x}_1)^n,\ \ldots\right).$$

As mentioned in Sec. 4.2.5 we can simply augment the paths with a monotonous coordinate, such that $\hat{\boldsymbol{x}} : t \mapsto (t, \boldsymbol{x}(t))$, where $t \in [0, T]$, effectively reintroducing a time parametrization. Another way to augment the state space of the data and obtain additional signature terms is the *lead-lag* transformation (see Def. 19) which turns a 1-d data stream into a 2-d path. For example if the data stream is $\{1, 5, 3\}$ one obtains the 2-d continuous path $\hat{\boldsymbol{x}} : t \mapsto (\boldsymbol{x}^{\mathrm{lead}}(t), \boldsymbol{x}^{\mathrm{lag}}(t))$ where $\boldsymbol{x}^{\mathrm{lead}}$ and $\boldsymbol{x}^{\mathrm{lag}}$ are the linear interpolations of $\{1, 5, 5, 3, 3\}$ and $\{1, 1, 5, 5, 3\}$ respectively. A key property of the lead-lag transform is that the difference between $S^{12}(\hat{\boldsymbol{x}})$ and $S^{21}(\hat{\boldsymbol{x}})$ is the quadratic variation $QV(\boldsymbol{x}) = \sum_{k=1}^{\ell-1}(\boldsymbol{x}(t_{k+1}) - \boldsymbol{x}(t_k))^2$ (Chevyrev and Kormilitzin, 2016). Hence, even when $d > 1$, it may be of interest to lead-lag transform the coordinates of the paths for which the quadratic variation is important for the task at hand.

Definition 19 (Lead-lag). *Given a sequence of points $\underline{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_\ell)$ in $\mathbb{R}^d$ the lead-lag transform yields two new sequences $\underline{\mathbf{x}}^{lead}$ and $\underline{\mathbf{x}}^{lag}$ of length $2\ell - 1$ of the following form*

$$\mathbf{x}_p^{lead} = \begin{cases} \mathbf{x}_k & \text{if } p = 2k-1 \\ \mathbf{x}_k & \text{if } p = 2k-2. \end{cases} \qquad \mathbf{x}_p^{lag} = \begin{cases} \mathbf{x}_k & \text{if } p = 2k-1 \\ \mathbf{x}_k & \text{if } p = 2k. \end{cases}$$

In our experiments we add time and lead-lag all coordinates except for the first task which consists in inferring the phase of an electronic circuit (see

Sec. 4.5.1).

B.2.2 Implementation details

The distribution regression methods (including DR-k, KES and SES) are implemented on top of the Scikit-learn library (Pedregosa et al., 2011), whilst we use the existing codebase `https://github.com/manzilzaheer/DeepSets` for DeepSets.

KES

The KES algorithm relies on the signature kernel trick which is referred to as PDESolve in this thesis. We used Alg. 2 where the discretization level of the PDE solver is fixed to $\kappa = 0$ such that the time complexity to approximate the solution of the PDE is $\mathcal{O}(d\ell^2)$ where ℓ is the length of the longest data stream.

SES

The SES algorithm from Chapter 4 relies on an algebraic property for fast computation of signatures, known as Chen's relation. Given a piecewise linear path $\boldsymbol{x} = \Delta\boldsymbol{x}(t_2) \star \ldots \star \Delta\boldsymbol{x}(t_\ell)$ given by the concatenation $\star$ of individual increments $\mathbb{R}^d \ni \Delta\boldsymbol{x}(t_k) = \boldsymbol{x}(t_k) - \boldsymbol{x}(t_{k-1}),\ k = 2, \ldots, \ell$, one has $S(\boldsymbol{x}) = \exp(\Delta\boldsymbol{x}(t_2)) \otimes \ldots \otimes \exp(\Delta\boldsymbol{x}(t_\ell))$, where exp denotes the tensor exponential and $\otimes$ the tensor product. Using Chen's relation, computing the signature (truncated at level n) of a sequence of length ℓ has complexity $\mathcal{O}(\ell d^n)$.

Baselines

Table B.1: Kernels k for the kernel-based baselines. See Cuturi and Blondel (2017) for the definition of $\text{dtw}_{1/\gamma}$ in the GA kernel.

RBF	$\exp(-\gamma^2 \lVert\mathbf{x} - \mathbf{x}'\rVert^2)$
Matern32	$(1 + \sqrt{3}\gamma^2 \lVert\mathbf{x} - \mathbf{x}'\rVert) \exp(-\sqrt{3}\gamma^2 \lVert\mathbf{x} - \mathbf{x}'\rVert)$
GA	$\exp(-\gamma\, \text{dtw}_{1/\gamma}(\boldsymbol{x}, \boldsymbol{x}'))$

For the kernel-based baselines DR-k, we perform Kernel Ridge regression with the kernel defined by $k_{\text{dr}}(\mathbb{P}_i, \mathbb{P}_j) = \exp\left(-\sigma^2 \lVert\rho(\mathbb{P}_i) - \rho(\mathbb{P}_j)\rVert^2_{\mathcal{H}_k}\right)$, where $\rho(\mathbb{P}_i) = N_i^{-1} \sum_{p=1}^{N_i} k(\cdot, \boldsymbol{x}_{i,p})$. For $k \in \{\text{RBF}, \text{Matern32}\}$, if the time-series are multi-dimensional, the dimensions are stacked to form one large vector $\mathbf{x} \in \mathbb{R}^{d\ell}$. See Table B.1 for the expressions of the kernels k used as baselines.

For DeepSets, the two neural networks are feedforward neural networks with ELU activations. We train DeepSets by minimizing the mean squared error.

B.2.3 Hyperparameter selection

All models are run 5 times. The hyperparameters of KES, SES and DR-k are selected by cross-validation via a grid search on the training set (80% of the data selected at random) of each run. The range of values for each parameter is specified in Table B.2.

Table B.2: Range of values for each parameter of DR-k, KES and SES. We denote by α the regularization parameter in kernel Ridge regression and Lasso regression. The kernels parameters γ and σ are expressed in terms of lengthscales ℓ_1 and ℓ_2 such that $\gamma^2 = 1/(2\ell_1^2)$ and $\sigma^2 = 1/(2\ell_2^2)$.

Model	ℓ_1	ℓ_2	α	n	m
DR-RBF	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	N/A	N/A
DR-Matern32	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	N/A	N/A
DR-GA	$\{7 \cdot 10^{1}, 7 \cdot 10^{2}\}$	$\{10^{-3}, 10^{-2}, \cdots, 10^{2}, 10^{3}\}$	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	N/A	N/A
KES	N/A	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	$\{10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}\}$	N/A	N/A
SES	N/A	N/A	$\{10^{-5}, 10^{-4}, \ldots, 10^{4}, 10^{5}\}$	$\{2, 3\}$	$\{2\}$

B.3 Interpretability

When dealing with complex data streams, interactions 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 $0 < u_1 < \ldots < u_k < T$. This ordering of the domain of integration, naturally captures the influence that the coordinate-paths $x^{i_1}, \ldots, x^{i_k}$ may have on each other. For example, considering only two variables x^i and x^j, if an increase (decrease) in x^i is systematically followed by an increase (decrease) in x^j, the term $S^{i,j}(\boldsymbol{x})$ will be positive. Conversely, if an increase (decrease) in x^i is systematically followed by a decrease (increase) in x^j, the term $S^{i,j}(\boldsymbol{x})$ will be negative.

Taking this property into account, we revisit the crop yield prediction example (see Sec. 4.5.4 and Fig. B.2) to show how the iterated integrals from the signature (of the 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 Huete et al. (2002). 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) (Huete et al., 2002). To carry out this experiment, we use a publicly available dataset (Hubert-Moy et al., 2019) 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 available at `http://ec.europa.eu/eurostat/data/database`). Instead of relying on a predefined vegetation index signal, such as the aforementionned NDVI : $t \mapsto (x^{nR}(t) - x^{R}(t))/(x^{nR}(t) + x^{R}(t))$, we use the raw signals in the form of 2-dimensional paths $\boldsymbol{x} : t \mapsto (x^{nR}(t), x^{R}(t))$ to perform a Lasso DR with SES.

Interpretation Chlorophyll strongly absorbs light at wavelengths around $0.67\mu m$ (red) and reflects strongly 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 (Rahman et al., 2004). 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 Fig. B.1 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.

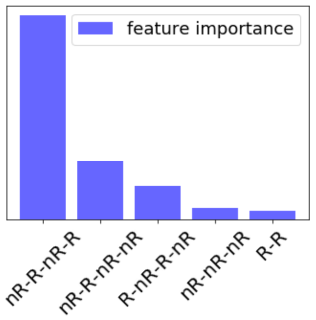


Figure B.1: The 5 most predictive features provided by (Lasso) SES for the task of crop yield prediction.

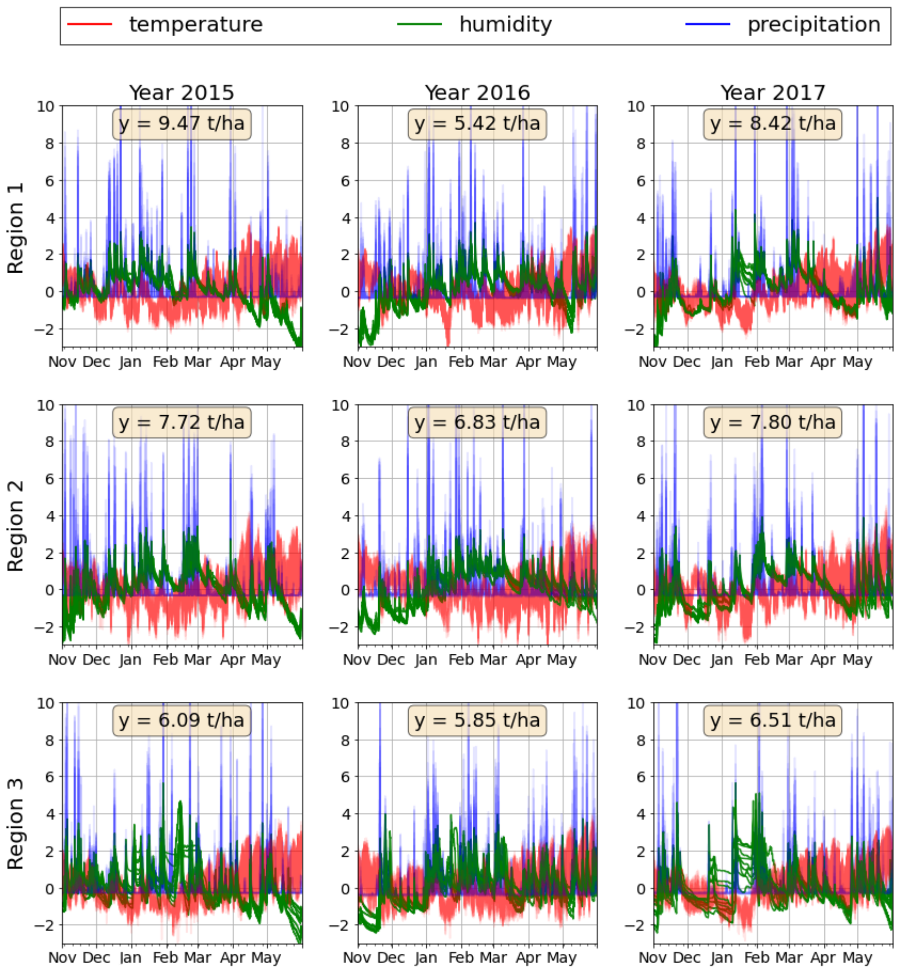


Figure B.2: GLDAS/Eurostat dataset. Each panel shows the normalized time-series of temperature, humidity and precipitation, measured over 10 different locations across a region within a year.

Appendix C

Appendix to Chapter 5

This Appendix is organised as follows: C.1) using the formalism of cross-covariance operators we define an Hilbert-Schmidt conditional independence criterion for stochastic processes, and provide further details on the construction of the estimator for the 2nd order MMD; C.2) we outline algorithms and their complexities for computing higher order MMDs; C.3) we provide further experimental details; C.4) we prove the theorems from Chapter 5.

C.1 Cross-covariance operators

Covariance and cross-covariance operators on RKHSs are important concepts for modern applications of conditional KMEs (Fukumizu et al., 2004; Song et al., 2009). In this section we will use this formalism (adapted to the case of path-valued random variables) to firstly derive a criterion for conditional independence of stochastic processes and secondly provide more details on the derivation of the estimator of the 2nd order MMD from Chapter 5.

Let $X, Y \in \mathcal{P}(\mathcal{X}(V))$ be two stochastic processes and their joint process $(X, Y) \in \mathcal{P}(\mathcal{X}(V \oplus V))$. Define the *cross-covariance operator* $\mathcal{C}_{Y,X}$ as the following point in the tensor product of RKHSs $\mathcal{H}(V) \otimes \mathcal{H}(V)$

$$\mathcal{C}_{Y,X} = \mathbb{E}_{(X,Y)}[K(\cdot, Y) \otimes K(\cdot, X)],$$

or equivalently as the *Hilbert-Schmidt operator* $\mathcal{C}_{Y,X} : \mathcal{H}(V) \to \mathcal{H}(V)$ defined for any function $f \in \mathcal{H}(V)$ as follows

$$\mathcal{C}_{Y,X}(f)(\cdot) = \int_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{X}(V \oplus V)} K(\cdot, \boldsymbol{y}) f(\boldsymbol{x}) \mathbb{P}_{(X,Y)}(d(\boldsymbol{x}, \boldsymbol{y})).$$

Let $\mathrm{HS}(\mathcal{H}(V), \mathcal{H}(V))$ be the space of Hilbert-Schmidt operators from $\mathcal{H}(V)$ to itself. The equivalence between tensor product of RKHSs and Hilbert-Schmidt operators is given by the isomorphism $\Phi : \mathcal{H}(V) \otimes \mathcal{H}(V) \to \mathrm{HS}(\mathcal{H}(V), \mathcal{H}(V))$

defined as follows

$$\Phi\left(\sum_{\substack{i_1\dots i_n\\ j_1\dots j_m}} \alpha^{i_1\dots i_n}_{j_1\dots j_m} S^{i_1\dots i_n}\otimes S^{j_1\dots j_m}\right) = \sum_{\substack{i_1\dots i_n\\ j_1\dots j_m}} \alpha^{i_1\dots i_n}_{j_1\dots j_m} \left\langle \cdot, S^{i_1\dots i_n}\right\rangle S^{j_1\dots j_m},$$

where $S^{i_1\dots i_n}$ denotes an element of an orthogonal basis of $\mathcal{H}(V)$ and the inner product is taken in $\mathcal{H}(V)$. An example of such basis is given by the *signature basis* defined for any path $\boldsymbol{x} \in \mathcal{X}(V)$ and any coordinate $(i_1, ..., i_n)$ as

$$S^{i_1\dots i_n}(\boldsymbol{x}) = \int_{0<t_1<\dots<t_n<T} \dots \int dx^{i_1}(t_1)\dots dx^{i_n}(t_n).$$

The centered version $\widetilde{\mathcal{C}}_{Y,X} \in \mathcal{H}(V)\otimes\mathcal{H}(V)$ of the operator $\mathcal{C}_{Y,X}$ is defined as

$$\widetilde{\mathcal{C}}_{Y,X} = \mathcal{C}_{Y,X} - \mu^{(1)}_X \otimes \mu^{(1)}_Y.$$

Similarly let $\mathcal{C}_{X,X} \in \mathcal{H}(V)\otimes\mathcal{H}(V)$ be the following covariance operator

$$\mathcal{C}_{X,X} = \mathbb{E}_X[K(\cdot, X)\otimes K(\cdot, X)],$$

or equivalently $\mathcal{C}_{X,X} \in \mathrm{HS}\left(\mathcal{H}(V), \mathcal{H}(V)\right)$

$$\mathcal{C}_{X,X}(f)(\cdot) = \int_{\boldsymbol{x}\in\mathcal{X}(V)} K(\cdot, \boldsymbol{x}) f(\boldsymbol{x}) \mathbb{P}_X(d\boldsymbol{x}).$$

Under the assumption that for any $f \in \mathcal{H}(V)$ the function $\boldsymbol{x} \mapsto \mu^{(1)}_{f(Y)|X=\boldsymbol{x}}$ from $\mathcal{X}(V)$ to $\mathbb{R}$ is in $\mathcal{H}(V)$, the authors in Fukumizu et al. (2004); Song et al. (2009) showed that

$$\mu^{(1)}_{Y|X} = \mathcal{C}_{Y,X}\mathcal{C}^{-1}_{X,X}. \tag{C.1}$$

However, this assumption might not hold in general (Fukumizu et al., 2004; Song et al., 2009). This technical issue can be circumvented by resorting to a regularized version of eq. (C.1): this yields to

$$\mu^{(1)}_{Y|X} \approx \mathcal{C}_{Y,X}(\mathcal{C}_{X,X} + \lambda I_{\mathcal{H}(V)})^{-1}, \quad \lambda > 0 \tag{C.2}$$

where $I_{\mathcal{H}(V)}$ is the identity map from $\mathcal{H}(V)$ to itself. Under some mild conditions the empirical estimator of eq. (C.2) is equal to the empirical estimator of eq. (C.1) (Fukumizu et al., 2013, Thm. 8). In particular, one has

$$\hat{\mu}^{(1)}_{Y|X=\boldsymbol{x}} = \mathbf{k}_X(\boldsymbol{x})'(\mathbf{K}_X + m\lambda\mathbf{I}_m)^{-1}\mathbf{k}_Y(\cdot)$$

based on sample paths $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$ from (X, Y) where $\mathbf{k}_X(\boldsymbol{x})$, $\mathbf{K}_X$ and $\mathbf{k}_Y(\cdot)$

are such that

$$[\mathbf{k}_X(\boldsymbol{x})]_i = K(\boldsymbol{x}_i, \boldsymbol{x}) \quad [\mathbf{K}_X]_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \mathbf{k}_Y(\cdot) = [K(\cdot, \boldsymbol{y}_1), \dots, K(\cdot, \boldsymbol{y}_m)]'.$$

Cross-covariance operators have been used to define kernel-based measures of conditional dependence, as we shall discuss in the next section.

C.1.1 Hilbert-Schmidt conditional independence criterion for stochastic processes

Multiple measures of conditional dependence have been proposed in the literature (Fukumizu et al., 2007; Sun et al., 2007; Tillman et al., 2009). In this section, we follow Tillman et al. (2009) to define a nonparametric conditional dependence measure for stochastic processes, based on the *conditional cross-covariance operator* $\widetilde{\mathcal{C}}_{Y,X|Z} : \mathcal{H}(V) \to \mathcal{H}(V)$

$$\widetilde{\mathcal{C}}_{Y,X|Z} = \widetilde{\mathcal{C}}_{Y,X} - \widetilde{\mathcal{C}}_{Y,Z}\widetilde{\mathcal{C}}_{Z,Z}^{-1}\widetilde{\mathcal{C}}_{Z,X}. \tag{C.3}$$

The squared Hilbert-Schmidt norm $H_{YX|Z} := \|\widetilde{\mathcal{C}}_{(Y,Z),X|Z}\|_{\text{HS}}^2$ can be used as measure of conditional dependence of stochastic processes. Since the signature kernel K is characteristic, it follows that $X \perp\!\!\!\perp Y \mid Z \iff H_{YX|Z} = 0$ (Tillman et al., 2009).

Given m sample paths $\{(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z}_i)\}_{i=1}^m$ from the joint distribution of (X, Y, Z), let $\mathbf{K}_X, \mathbf{K}_Y$ and $\mathbf{K}_Z$ be the Gram matrices with entries

$$[\mathbf{K}_X]_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad [\mathbf{K}_Y]_{i,j} = K(\boldsymbol{y}_i, \boldsymbol{y}_j) \quad [\mathbf{K}_Z]_{i,j} = K(\boldsymbol{z}_i, \boldsymbol{z}_j).$$

An empirical estimator of the kernel conditional dependence measure $H_{YX|Z}$ is then given by

$$\begin{aligned}\widehat{H}_{YX|Z} = \frac{1}{m^2}\Big\{&\text{tr}(\widetilde{\mathbf{K}}_X\widetilde{\mathbf{K}}_Y) - 2\text{tr}(\widetilde{\mathbf{K}}_X\widetilde{\mathbf{K}}_Z(\widetilde{\mathbf{K}}_Z^{\epsilon})^{-2}\widetilde{\mathbf{K}}_Z\widetilde{\mathbf{K}}_Y)\\ &+ \text{tr}(\widetilde{\mathbf{K}}_X\widetilde{\mathbf{K}}_Z(\widetilde{\mathbf{K}}_Z^{\epsilon})^{-2}\widetilde{\mathbf{K}}^Z\widetilde{\mathbf{K}}_Y\widetilde{\mathbf{K}}_Z(\widetilde{\mathbf{K}}_Z^{\epsilon})^{-2}\widetilde{\mathbf{K}}_Z)\Big\},\end{aligned}$$

where $\widetilde{\mathbf{K}}_Z^{\epsilon} = \widetilde{\mathbf{K}}_z + \epsilon\mathbf{I}_m$ and $\widetilde{\mathbf{K}}_X, \widetilde{\mathbf{K}}_Y, \widetilde{\mathbf{K}}_Z$ are the centered versions of the matrices $\mathbf{K}_X, \mathbf{K}_Y$ and $\mathbf{K}_Z$

$$\widetilde{\mathbf{K}}_X = \mathbf{H}\mathbf{K}_X\mathbf{H} \quad \widetilde{\mathbf{K}}_Y = \mathbf{H}\mathbf{K}_Y\mathbf{H} \quad \widetilde{\mathbf{K}}_Z = \mathbf{H}\mathbf{K}_Z\mathbf{H},$$

with $\mathbf{H} = \mathbf{I}_m - m^{-1}\mathbf{1}_m$ and $\mathbf{1}_m$ the $m \times m$ matrix with all entries set to 1. This estimator can be used as a test statistic for testing whether X and Y are independent given Z. However, it is not known how to analytically compute the null distribution of the test statistic, and permutation tests are typically used. In Sec. 5.4.3 we use this measure of conditional dependence as part of the kPC algorithm to infer causal relationships between multidimensional stochastic processes. We provide more details in Appendix C.3.

C.1.2 Construction of the estimator for the second order MMD

As discussed in Chapter 5, the estimation of the second order MMD, requires the ability to compute inner products of the form $\langle \widetilde{\boldsymbol{x}}(s), \widetilde{\boldsymbol{y}}(t) \rangle$ in $\mathcal{H}(V)$. Here, we provide more details on the approximation that we have used in eq. (5.12), also restated below for the reader's convenience

$$\big\langle \widetilde{\boldsymbol{x}}(s), \widetilde{\boldsymbol{y}}(t) \big\rangle_{\mathcal{H}(V)} \approx \mathbf{k}_X^s(\boldsymbol{x})'(\mathbf{K}_X^s + m\lambda \mathbf{I}_m)^{-1}\mathbf{K}_{X,Y}(\mathbf{K}_Y^t + n\lambda \mathbf{I}_n)^{-1}\mathbf{k}_Y^t(\boldsymbol{y}),$$

where $\widetilde{\boldsymbol{x}}$ and $\widetilde{\boldsymbol{y}}$ are sample paths from the processes $\mu^{(1)}_{X|\mathcal{F}_X}$ and $\mu^{(1)}_{Y|\mathcal{F}_Y}$ respectively. In particular, we have

$$\widetilde{\boldsymbol{x}}(s) = \mu^{(1)}_{X|\boldsymbol{x}|_{[0,s]}} \quad \text{and} \quad \widetilde{\boldsymbol{y}}(t) = \mu^{(1)}_{Y|\boldsymbol{y}|_{[0,t]}}.$$

As discussed at the beginning of this section, their empirical estimates are constructed from m samples $\{\boldsymbol{x}_i\}_{i=1}^m$ from X and n samples $\{\boldsymbol{y}_j\}_{j=1}^n$ from Y respectively

$$\widetilde{\boldsymbol{x}}(s) \approx \mathbf{k}_X^s(\boldsymbol{x})'(\mathbf{K}_X^s + m\lambda \mathbf{I}_m)^{-1}\mathbf{k}_X(\cdot) \quad \text{and} \quad \widetilde{\boldsymbol{y}}(t) \approx \mathbf{k}_Y^t(\boldsymbol{y})'(\mathbf{K}_Y^t + n\lambda \mathbf{I}_n)^{-1}\mathbf{k}_Y(\cdot) \tag{C.4}$$

where $\mathbf{k}_X^s(\boldsymbol{x})$, $\mathbf{K}_X^s$, $\mathbf{k}_X(\cdot)$ are defined by

$$[\mathbf{k}_X^s(\boldsymbol{x})]_i = K(\boldsymbol{x}_i|_{[0,s]}, \boldsymbol{x}|_{[0,s]}) \qquad [\mathbf{K}_X^s]_{i,j} = K(\boldsymbol{x}_i|_{[0,s]}, \boldsymbol{x}_j|_{[0,s]})$$

and

$$\mathbf{k}_X(\cdot) = [K(\cdot, \boldsymbol{x}_1), \ldots, K(\cdot, \boldsymbol{x}_m)]'.$$

Alternatively, we can write

$$\widetilde{\boldsymbol{x}}(s) \approx \sum_{i=1}^m \alpha_i K(\cdot, \boldsymbol{x}_i) \quad \boldsymbol{\alpha} = (\mathbf{K}_X^s + m\lambda \boldsymbol{I}_m)^{-1}\mathbf{k}_X^s(\boldsymbol{x})$$
$$\widetilde{\boldsymbol{y}}(t) \approx \sum_{j=1}^n \beta_j K(\cdot, \boldsymbol{y}_j) \quad \boldsymbol{\beta} = (\mathbf{K}_Y^t + n\lambda \mathbf{I}_n)^{-1}\mathbf{k}_Y^t(\boldsymbol{y}).$$

Therefore, the inner product between $\widetilde{\boldsymbol{x}}(s)$ and $\widetilde{\boldsymbol{y}}(t)$ can be approximated as follows

$$\begin{aligned}\langle \widetilde{\boldsymbol{x}}(s), \widetilde{\boldsymbol{y}}(t)\rangle_{\mathcal{H}(V)} &\approx \sum_{i=1}^{m}\sum_{j=1}^{n}\alpha_i\beta_j K(\boldsymbol{x}_i, \boldsymbol{y}_j)\\ &= \boldsymbol{\alpha}'\mathbf{K}_{XY}\boldsymbol{\beta}\\ &= \mathbf{k}_X^s(\boldsymbol{x})'(\mathbf{K}_X^s + m\lambda \boldsymbol{I}_m)^{-1}\mathbf{K}_{XY}(\mathbf{K}_Y^t + n\lambda \boldsymbol{I}_n)^{-1}\mathbf{k}_Y^t(\boldsymbol{y}),\end{aligned}$$

where $\mathbf{K}_{XY} \in \mathbb{R}^{m\times n}$ with $[\mathbf{K}_{XY}]_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{y}_j)$. Next we outline the algorithm to compute an estimate of the second order MMD.

C.2 Algorithms

In this section, we provide algorithms to compute the empirical estimate $\hat{\mathscr{D}}_K^{(k)}$ for the k^{th} order MMD, which rely on the ability to evaluate the signature kernel $K(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ are two paths taking their values in the Hilbert space $\mathcal{H}^{(k-1)}(V)$ (with the convention that $\mathcal{H}^{(0)}(V) = V$). Following Cass et al. (2020, Sec. 3.1.) we use an explicit finite difference scheme to approximate the PDE solution u on a grid $\mathcal{D}$ of size $P \times Q$

$$\mathcal{D} = \{0 = s_1 < s_2 < \ldots < s_P = T\} \times \{0 = t_1 < t_2 < \ldots < t_Q = T\}.$$

Writing $u(s_i, t_j) = u_{i,j}$ to make the notation more concise, we use an update rule of the form

$$u_{i+1,j+1} = f_{\text{upd}}(u_{i,j+1}, u_{i+1,j}, u_{i,j}, M_{i,j}), \quad M_{i,j} = \langle \boldsymbol{x}_{s_{i+1}} - \boldsymbol{x}_{s_i}, \boldsymbol{y}_{t_{j+1}} - \boldsymbol{y}_{t_j}\rangle.$$

Hence, computing $K(\boldsymbol{x}, \boldsymbol{y})$ consists in forming the $(P-1)\times(Q-1)$ matrix $\mathbf{M}$ such that

$$[\mathbf{M}]_{i,j} = \langle \boldsymbol{x}(s_{i+1}) - \boldsymbol{x}(s_i), \boldsymbol{y}(t_{j+1}) - \boldsymbol{y}(t_j)\rangle_{\mathcal{H}^{(k-1)}(V)},$$

and iteratively applying the update rule as outlined in Alg. 8. Besides, in Alg. 8 we distinguish the case where the solution u on the entire grid is returned, and the case where only the solution at the final times $(s_P, t_Q) = (T, T)$ is returned, which corresponds to the value of the kernel $K(\boldsymbol{x}, \boldsymbol{y})$. The runtime complexity to solve one PDE is $\mathcal{O}(PQ)$. We make use of the parallelization strategy described in Appendix A.3.2 to drastically speed-up the PDE solver on CUDA-enabled GPUs.

C.2.1 Algorithm for the first order MMD

In this section we provide the algorithm to compute an empirical estimator of the 1st order MMD. This way, we also introduce subroutines (Alg. 8 and Alg. 9) for the estimator of the 2nd order MMD.

Let $\{\boldsymbol{x}_i\}_{i=1}^m \sim X$ and $\{\boldsymbol{y}_j\}_{j=1}^n \sim Y$. An estimate for the 1st order MMD Gretton et al. (2012a) reads as

$$\left(\widehat{\mathscr{D}}_K^{(1)}(X,Y)\right)^2 = \frac{1}{m(m-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{m} K(\boldsymbol{x}_i,\boldsymbol{x}_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} K(\boldsymbol{x}_i,\boldsymbol{y}_j) + \frac{1}{n(n-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{n} K(\boldsymbol{y}_i,\boldsymbol{y}_j).$$

Hence, in order to compute this estimate, we need to form the following three Gram matrices $\mathbf{G}^1_{X,X} \in \mathbb{R}^{m\times m}$, $\mathbf{G}^1_{X,Y} \in \mathbb{R}^{m\times n}$ and $\mathbf{G}^1_{Y,Y} \in \mathbb{R}^{n\times n}$ such that,

$$[\mathbf{G}^1_{X,X}]_{i,j} = K(\boldsymbol{x}_i,\boldsymbol{x}_j) \quad [\mathbf{G}^1_{X,Y}]_{i,j} = K(\boldsymbol{x}_i,\boldsymbol{y}_j) \quad [\mathbf{G}^1_{Y,Y}]_{i,j} = K(\boldsymbol{y}_i,\boldsymbol{y}_j)$$

As explained at the begining of this section (and outlined in Alg. 9), this consists in two steps. Taking $\mathbf{G}^1_{X,Y}$ for example, first one forms $m\times n$ matrices of size $(P-1)\times(Q-1)$ each of the form,

$$[\mathbf{M}]_{p,q} = \langle \boldsymbol{x}_i(s_{p+1}) - \boldsymbol{x}_i(s_p),\ \boldsymbol{y}_j(t_{q+1}) - \boldsymbol{y}_j(t_q)\rangle_V$$

and then one solves $m\times n$ PDEs with Alg. 8. The full procedure is summarized in Alg. 10. Assuming that $m=n$ and $P=Q$, Alg. 10 has time complexity $\mathcal{O}(dm^2P^2)$ where d is the number of coordinates of the paths x and y.

Algorithm 8 PDESolve $\mathcal{O}(P^2)$

```
1: Input: matrix $M \in \mathbb{R}^{P\times Q}$, full $\in$ {True, False}
2: Output: solution $u \in \mathbb{R}^{P\times Q}$ with $u[p,q] = K(\boldsymbol{x}|_{[0,s_p]}, \boldsymbol{y}|_{[0,t_q]})$ if full,
   $u[P,Q] = K(\boldsymbol{x},\boldsymbol{y})$ otherwise
3: $u[1,:] \leftarrow 1$
4: $u[:,1] \leftarrow 1$
5: for $p$ from 1 to $P-1$ do
6:     for $q$ from 1 to $Q-1$ do
7:         $u[p+1,q+1] \leftarrow f_{\mathrm{upd}}(u[p,q+1], u[p+1,q], u[p,q], M[p,q])$
8:     end for
9: end for
10: if full then return $u$ else return $u[P,Q]$
```

Algorithm 9 FirstOrderGram $\mathcal{O}(dm^2P^2)$

```
1: Input: Sample paths $\{\boldsymbol{x}_i\}_{i=1}^m \sim X$ and $\{\boldsymbol{y}_j\}_{j=1}^n \sim Y$, full $\in \{\text{True}, \text{False}\}$
2: Output: $G \in \mathbb{R}^{m\times n\times P\times Q}$ where $G[i,j,p,q] = K(\boldsymbol{x}_i|_{[0,s_p]}, \boldsymbol{y}_j|_{[0,t_q]})$ if full,
   $G[:,:,P,Q]$ otherwise
3: $M[i,j,p,q] \leftarrow \langle \boldsymbol{x}_i(s_p), \boldsymbol{y}_j(t_q)\rangle \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,n\},\ \forall p \in \{1,\dots,P\},\ \forall q \in \{1,\dots,Q\}$
4: $M \leftarrow M[:,:,1:,1:] + M[:,:,:-1,:-1] - M[:,:,1:,:-1] - M[:,:,:-1,1:]$
5: $G[i,j] \leftarrow \mathsf{PDESolve}(M[i,j]), \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,n\}$
6: if full then return $G$ else return $G[:,:,P,Q]$
```

Algorithm 10 FirstOrderMMD $\mathcal{O}(dm^2P^2)$

```
1: Input: sample paths $\{\boldsymbol{x}_i\}_{i=1}^m \sim X$ and $\{\boldsymbol{y}_j\}_{j=1}^n \sim Y$
2: Output: an empirical estimator of the 1st order MMD between $X$ and $Y$
3: $G^1_{XX} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{x}_i\}_{i=1}^m, \{\boldsymbol{x}_i\}_{i=1}^m, \mathsf{full} = \text{False})$
4: $G^1_{XY} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{x}_i\}_{i=1}^m, \{\boldsymbol{y}_j\}_{j=1}^n, \mathsf{full} = \text{False})$
5: $G^1_{YY} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{y}_j\}_{j=1}^n, \{\boldsymbol{y}_j\}_{j=1}^n, , \mathsf{full} = \text{False})$
6: return $\text{avg}(G^1_{XX}) - 2 * \text{avg}(G^1_{XY}) + \text{avg}(G^1_{YY})$
```

C.2.2 Algorithm for the second order MMD

In Chapter 5, we used the following estimate of the 2nd order MMD,

$$\left(\widehat{\mathscr{D}}^{(2)}_K(X,Y)\right)^2 = \frac{1}{m(m-1)} \sum_{\substack{i,j=1\\ i\neq j}}^{m} K(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} K(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_j) + \frac{1}{n(n-1)} \sum_{\substack{i,j=1\\ i\neq j}}^{n} K(\widetilde{\boldsymbol{y}}_i, \widetilde{\boldsymbol{y}}_j)$$

Compared to the 1st order MMD, in order to compute this estimate, as outlined in Alg. 12, we need to form the following three Gram matrices $\mathbf{G}^2_{X,X} \in \mathbb{R}^{m\times m}$, $\mathbf{G}^2_{X,Y} \in \mathbb{R}^{m\times n}$ and $\mathbf{G}^2_{Y,Y} \in \mathbb{R}^{n\times n}$,

$$[\mathbf{G}^2_{X,X}]_{i,j} = K(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{x}}_j) \quad [\mathbf{G}^2_{X,Y}]_{i,j} = K(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_j) \quad [\mathbf{G}^2_{Y,Y}]_{i,j} = K(\widetilde{\boldsymbol{y}}_i, \widetilde{\boldsymbol{y}}_j)$$

As outlined in Alg. 11, this consists in two steps. Taking $\mathbf{G}^2_{X,Y}$ for example, first one forms $m \times n$ matrices of size $(P-1)\times(Q-1)$ each of the form

$$[\mathbf{M}]_{p,q} = \langle \widetilde{\boldsymbol{x}}(s_{p+1}) - \widetilde{\boldsymbol{x}}(s_p), \widetilde{\boldsymbol{y}}(t_{q+1}) - \widetilde{\boldsymbol{y}}(t_q)\rangle_{\mathcal{H}(V)},$$

(see Alg. 13) and then one solves $m \times n$ PDEs with Alg. 8. This is summarized in Alg. 12, which has time complexity $\mathcal{O}((d+m)m^2P^2)$ where d is the number of coordinates of the paths $\boldsymbol{x}$ and $\boldsymbol{y}$.

Algorithm 11 SecondOrderGram $\mathcal{O}(P^2m^3)$

```
1: Input: $G_{XX}, G_{XY}, G_{YY}$ with $G_{XY}[i,j,p,q] = K(\boldsymbol{x}_i|_{[0,s_p]}, \boldsymbol{y}_j|_{[0,t_q]})$ and hyperparameter $\lambda$.
2: Output: an empirical estimate of $G^2_{X,Y} \in \mathbb{R}^{m\times n}$, where $G^2_{X,Y}[i,j] = K(\widetilde{\boldsymbol{x}}_i, \widetilde{\boldsymbol{y}}_j)$
3: $M \leftarrow \mathsf{InnerProdPredCondKME}(G_{XX}, G_{XY}, G_{YY}, \lambda)$
4: $M \leftarrow M[:,:,1:,1:] + M[:,:,:-1,:-1] - M[:,:,1:,:-1] - M[:,:,:-1,1:]$
5: $G^2_{XY}[i,j] \leftarrow \mathsf{PDESolve}(M[i,j]), \quad \forall i \in \{1,\dots,m\},\ \forall j \in \{1,\dots,n\}$
6: return $G^2_{XY}$
```

Algorithm 12 SecondOrderMMD $\mathcal{O}(dm^2P^2 + P^2m^3)$

```
1: Input: sample paths $\{\boldsymbol{x}_i\}_{i=1}^m \sim X$ and $\{\boldsymbol{y}_j\}_{j=1}^n \sim Y$, hyperparameter $\lambda$.
2: Output: an empirical estimate of the 2nd order MMD between $X$ and $Y$
3: $G^1_{XX} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{x}_i\}_{i=1}^m, \{\boldsymbol{x}_i\}_{i=1}^m, \mathsf{full} = \text{True})$
4: $G^1_{XY} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{x}_i\}_{i=1}^m, \{\boldsymbol{y}_j\}_{j=1}^n, \mathsf{full} = \text{True})$
5: $G^1_{YY} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{y}_j\}_{j=1}^n, \{\boldsymbol{y}_j\}_{j=1}^n, \mathsf{full} = \text{True})$
6: $G^2_{XX} \leftarrow \mathsf{SecondOrderGram}(G^1_{XX}, G^1_{XX}, G^1_{XX}, \lambda)$
7: $G^2_{XY} \leftarrow \mathsf{SecondOrderGram}(G^1_{XX}, G^1_{XY}, G^1_{YY}, \lambda)$
8: $G^2_{YY} \leftarrow \mathsf{SecondOrderGram}(G^1_{YY}, G^1_{YY}, G^1_{YY}, \lambda)$
9: return $\text{avg}(G^2_{XX}) - 2 * \text{avg}(G^2_{XY}) + \text{avg}(G^2_{YY})$
```

Algorithm 13 InnerProdPredCondKME $\mathcal{O}(P^2m^3)$

```
1: Input: three Gram matrices $G_{XX}, G_{XY}, G_{YY}$ and hyperparameter $\lambda$
2: Output: returns an empirical estimate of $M \in \mathbb{R}^{m\times n\times P\times Q}$ where
   $M[i,j,p,q] = \langle \widetilde{\boldsymbol{x}}_i(s_p), \widetilde{\boldsymbol{y}}_j(t_q)\rangle$
3: $W_X[:,:,p] \leftarrow (G_{XX}[:,:,p,p] + m\lambda I)^{-1}, \quad \forall p \in \{1,\dots,P\}$
4: $W_Y[:,:,q] \leftarrow (G_{YY}[:,:,q,q] + n\lambda I)^{-1}, \quad \forall q \in \{1,\dots,Q\}$
5: for $p$ from 1 to $P$ do
6:     for $q$ from 1 to $Q$ do
7:         $GW_X \leftarrow G_{XX}[:,:,p,p]W_X[:,:,p]$
8:         $WG_Y \leftarrow W_Y[:,:,q]G_{YY}[:,:,q,q]$
9:         $M[:,:,p,q] \leftarrow GW_X G_{XY}[:,:,P,Q]WG_Y$
10:    end for
11: end for
```

C.2.3 Algorithm for higher order MMDs

Now, we generalize the procedure in Appendix C.2.2 for computing an estimate of $\mathscr{D}_K^{(k+1)}$ when $k>1$,

$$\left(\widehat{\mathscr{D}}_K^{(k+1)}(X,Y)\right)^2 = \frac{1}{m(m-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{m} K(\widetilde{\boldsymbol{x}}_i^{(k)}, \widetilde{\boldsymbol{x}}_j^{(k)}) - \frac{2}{mn}\sum_{i,j=1}^{m,n} K(\widetilde{\boldsymbol{x}}_i^{(k)}, \widetilde{\boldsymbol{y}}_j^{(k)}) + \frac{1}{n(n-1)}\sum_{\substack{i,j=1\\ i\neq j}}^{n} K(\widetilde{\boldsymbol{y}}_i^{(k)}, \widetilde{\boldsymbol{y}}_j^{(k)}),$$

where $\widetilde{\boldsymbol{x}}_i^{(k)}$ and $\widetilde{\boldsymbol{y}}_j^{(k)}$ denote sample paths from the processes $\mu^{(k)}_{X|\mathcal{F}_X}$ and $\mu^{(k)}_{Y|\mathcal{F}_Y}$ respectively. In order to compute this estimate, as outlined in Alg. 12, we need to form the following three Gram matrices $\mathbf{G}^{k+1}_{X,X} \in \mathbb{R}^{m\times m}$, $\mathbf{G}^{k+1}_{X,Y} \in \mathbb{R}^{m\times n}$ and $\mathbf{G}^{k+1}_{Y,Y} \in \mathbb{R}^{n\times n}$,

$$[\mathbf{G}^{k+1}_{X,X}]_{i,j} = K(\widetilde{\boldsymbol{x}}_i^{(k)}, \widetilde{\boldsymbol{x}}_j^{(k)}) \quad [\mathbf{G}^{k+1}_{X,Y}]_{i,j} = K(\widetilde{\boldsymbol{x}}_i^{(k)}, \widetilde{\boldsymbol{y}}_j^{(k)}) \quad [\mathbf{G}^{k+1}_{Y,Y}]_{i,j} = K(\widetilde{\boldsymbol{y}}_i^{(k)}, \widetilde{\boldsymbol{y}}_j^{(k)})$$

As outlined in Alg. 14, this consists in two steps. Taking $\mathbf{G}^{k+1}_{X,Y}$ for example, first one forms $m\times n$ matrices of size $(P-1)\times(Q-1)$ each of the form,

$$[\mathbf{M}]_{p,q} = \left\langle \widetilde{\boldsymbol{x}}^{(k)}(s_{p+1}) - \widetilde{\boldsymbol{x}}^{(k)}(s_p), \widetilde{\boldsymbol{y}}^{(k)}(t_{q+1}) - \widetilde{\boldsymbol{y}}^{(k)}(t_q)\right\rangle_{\mathcal{H}^{(k)}(V)}$$

(see Alg. 13) and then one solves $m\times n$ PDEs with Alg. 8. This is summarized in Alg. 15, which has time complexity $\mathcal{O}((d+km)m^2P^2)$ where d is the number of coordinates of the paths $\boldsymbol{x}$ and $\boldsymbol{y}$.

Algorithm 14 HigherOrderGram $\mathcal{O}(P^2 m^3)$

```
1: Input: $G^k_{XX}, G^k_{XY}, G^k_{YY}$ with $G^k_{XY}[i,j,p,q] = K(\widetilde{\boldsymbol{x}}_i^{(k-1)}|_{[0,s_p]}, \widetilde{\boldsymbol{y}}_j^{(k-1)}|_{[0,t_q]})$ and hyperparameter $\lambda$.
2: Output: an estimate of $G^{k+1}_{XY} \in \mathbb{R}^{m\times n\times P\times Q}$, where $G^{k+1}_{XY}[i,j,p,q] = K(\widetilde{\boldsymbol{x}}_i^{(k)}|_{[0,s_p]}, \widetilde{\boldsymbol{y}}_j^{(k)}|_{[0,t_q]})$
3: $M \leftarrow \mathsf{InnerProdPredCondKME}(G^k_{XX}, G^k_{XY}, G^k_{YY}, \lambda)$
4: $M \leftarrow M[:,:,1:,1:] + M[:,:,:-1,:-1] - M[:,:,1:,:-1] - M[:,:,:-1,1:]$
5: $G^{k+1}_{XY}[i,j] \leftarrow \mathsf{PDESolve}(M[i,j], \mathsf{full} = \text{True}), \quad \forall i \in \{1,\dots,m\}, \forall j \in \{1,\dots,n\}$
6: return $G^{k+1}_{XY}$
```

Algorithm 15 HigherOrderMMD $\mathcal{O}(dm^2P^2 + (k-1)P^2m^3)$

```
1: Input: sample paths $\{\boldsymbol{x}_i\}_{i=1}^m \sim X$ and $\{\boldsymbol{y}_j\}_{j=1}^n \sim Y$, hyperparameter $\lambda$, order $k$.
2: Output: an empirical estimate of the $k^{\text{th}}$ order MMD between $X$ and $Y$
3: $G_{XX} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{x}_i\}_{i=1}^m, \{\boldsymbol{x}_i\}_{i=1}^m, \mathsf{full} = \text{True})$
4: $G_{XY} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{x}_i\}_{i=1}^m, \{\boldsymbol{y}_j\}_{j=1}^n, \mathsf{full} = \text{True})$
5: $G_{YY} \leftarrow \mathsf{FirstOrderGram}(\{\boldsymbol{y}_j\}_{j=1}^n, \{\boldsymbol{y}_j\}_{j=1}^n, \mathsf{full} = \text{True})$
6: for order from 2 to $k$ do
7:     $G^{\mathsf{new}}_{XX} \leftarrow \mathsf{HigherOrderGram}(G_{XX}, G_{XX}, G_{XX}, \lambda)$
8:     $G^{\mathsf{new}}_{XY} \leftarrow \mathsf{HigherOrderGram}(G_{XX}, G_{XY}, G_{YY}, \lambda)$
9:     $G^{\mathsf{new}}_{YY} \leftarrow \mathsf{HigherOrderGram}(G_{YY}, G_{YY}, G_{YY}, \lambda)$
10:    $G_{XX}, G_{XY}, G_{YY} \leftarrow G^{\mathsf{new}}_{XX}, G^{\mathsf{new}}_{XY}, G^{\mathsf{new}}_{YY}$
11: end for
12: return $\text{avg}(G_{XX}[:,:,P,P]) - 2 * \text{avg}(G_{XY}[:,:,P,Q]) + \text{avg}(G_{YY}[:,:,Q,Q])$
```

C.3 Experimental details

We start with further experimental details for the applications of higher order distribution regression to quantitative finance (Sec. 5.4.2), where we consider the problem of optimally stopping fractional Brownian motions with different hurst exponents.

C.3.1 Rough volatility

Rough volatility models constitute a class of models that are empirically well-tailored to fit observed implied market volatilities in the context of option pricing for short maturity assets. The basic model for option pricingis called the Black-Scholes model in which the volatility is assumed to be constant. Stochastic volatility models are extensions of the Black-Scholes model to the case where the volatility is itself stochastic. The main shortcoming of such stochastic volatility models is that they are able to capture the true steepness of the implied volatility smile close to maturity (see Bayer et al. (2016) for extra details). This is where rough volatility models become useful. Among them, the rough Bergomi model introduced by Bayer et al. (2016), stood out for its ability to explain implied volatility and other phenomena related to European options.

C.3.2 Higher order distribution regression

Data We use the data generator from the publicly available repository `https://github.com/HeKrRuTe/OptStopRandNN` to simulate sample paths from X a fractional Brownian motion (fBm) and obtain the solution of the optimal stopping time problem $\sup_\tau \mathbb{E}[g(X_\tau)\,|\,X_0]$. We note that although fBm is not typically used as a stock price model in quantitative finance, it is nevertheless considered a respected challenging example for optimal stopping algorithms (Becker et al., 2019; Herrera et al., 2021).

Models We use a kernel Ridge regressor with different distribution regression kernels. Each is of the form $K_{\mathrm{dr}}(X,Y) = \exp(-\mathscr{D}_k(X,Y)^2/\sigma^2)$ where $\mathscr{D}_k(X,Y)$ is a maximum mean discrepancy associated to the kernel k. The models $K_{\mathrm{dr}}^{(1)}$ and $K_{\mathrm{dr}}^{(2)}$ correspond to the 1st and 2nd order maximum mean discrepancies $\mathscr{D}_K^{(1)}$ and $\mathscr{D}_K^{(2)}$. We consider two other baselines (Matérn and RBF) for which the MMD is computed using the Matern 3/2 covariance function k_{mat32}, and the RBF covariance function k_{rbf},

$$\begin{aligned}
k_{\mathrm{mat32}}(\mathbf{x}, \mathbf{y}) &= \left(1 + \frac{\sqrt{3}}{\gamma^2} \|\mathbf{x} - \mathbf{y}\|\right) \exp\left(-\frac{\sqrt{3}}{\gamma^2} \|\mathbf{x} - \mathbf{y}\|\right), \\
k_{\mathrm{rbf}}(\mathbf{x}, \mathbf{y}) &= \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\gamma^2}\right)
\end{aligned}$$

All models are run 3 times. The hyperparameters of all models are selected by cross-validation via a grid search on the training set (70% of the data selected at random) of each run.

Inferring causal graph for interacting bodies

We provide further details for the last application (Sec. 5.4.3) where the task is to infer whether any two bodies are connected by a spring from multiple observations of their 2D trajectories.

Data We adapt the Pymunk simulator from Li et al. (2020a) publicly available at `https://github.com/pairlab/v-cdn`. For each pair of balls, there is a one-half probability that they are connected by nothing, or a spring. For each graph we run multiple episodes each of 20 time steps. At the beginning of each episode, we randomly assign the balls in different positions. The stiffness of the spring relation is set to 20, and we randomly sample the rest length between $[20, 120]$.

Causal discovery algorithm The PC algorithm (Spirtes et al., 2000) uses conditional independence tests to generate a causal graph from a dataset. The PC algorithm consists in two stages. The first stage, referred to as the *skeleton phase*, consists in finding the structure of the causal graph. In the second stage, the edges are oriented by repetitively applying orientation rules. In the multi-body interaction example, we only need to perform the skeleton phase, which is sketched hereafter,

1. Start with a complete graph.
2. For each X and Y which are still connected. If there is a third variable Z_1 connected to X or Y, such that $X \perp\!\!\!\perp Y \mid Z_1$, remove the edge between X and Y.
3. For each X and Y which are still connected, if there is a third and a fourth variable Z_1 and Z_2 connected to X or Y such that $X \perp\!\!\!\perp Y \mid Z_1, Z_2$, remove the edge between X and Y.
4. Iteratively increase the cardinality of the set of variables on which to condition.

To test for conditional independence we use the Hilbert-Schmidt conditional independence criterion $H_{XY|Z}$ for stochastic processes (Appendix C.1.1). The combination of the PC algorithm with a kernel-based dependence measure has been used in Sun et al. (2007) and Tillman et al. (2009) where it is termed kPC. However, to our knowledge it has never been used in conjunction with a kernel-based measure of dependence for multidimensional stochastic processes.

Since the null distribution of the test statistics $H_{YX|Z}$ is not known, one possibility would be to use a permutation approach as in Tillman et al. (2009, Sec 2). However the latter is not computationally efficient. We leave the development of a faster approach for future work, and adopt the approach from Sun et al. (2007) for this experiment. That is we use a threshold α and remove an edge if there is a Z such that $H_{XY|Z} < \alpha$. We repeat 15 times the causal discovery procedure and use 30% of the runs to fix α.

All experiments in Sec. 5.4 have been run on a P100 GPU to leverage an efficient dedicated CUDA implementation of the signature kernel.

C.4 Proofs

Theorem C.4.1. *Given two stochastic processes X, Y*

$$\mathscr{D}_K^{(2)}(X,Y) = 0 \iff \mathbb{P}_{X|\mathcal{F}_X} = \mathbb{P}_{Y|\mathcal{F}_Y}.$$

Furthermore

$$\mathscr{D}_K^{(2)}(X,Y) = 0 \implies \mathscr{D}_K^{(1)}(X,Y) = 0,$$

but the converse is not generally true.

Proof. First we note that by a standard result in signature kernel learning theory (Chevyrev and Oberhauser, 2018), for every t, the mapping

$$\mathbb{P}_{X|\mathcal{F}_{X_t}} \mapsto \mu^{(1)}_{X|\mathcal{F}_{X_t}} = \int K(\cdot, \boldsymbol{x}) \mathbb{P}_{X|\mathcal{F}_{X_t}}(d\boldsymbol{x})$$

is a homeomorphism (with respect to the weak topology and the Hilbert space topology); in particular, we have

$$\mathbb{P}_{X|\mathcal{F}_X} = \mathbb{P}_{Y|\mathcal{F}_Y} \iff \mathbb{P}_{\mu^{(1)}_{X|\mathcal{F}_X}} = \mathbb{P}_{\mu^{(1)}_{Y|\mathcal{F}_Y}}.$$

Then, using the same argument for $\mathbb{P}_{\mu^{(1)}_{X|\mathcal{F}_X}}$ and $\mathbb{P}_{\mu^{(1)}_{Y|\mathcal{F}_Y}}$, we can deduce that

$$\begin{aligned}\mathbb{P}_{\mu^{(1)}_{X|\mathcal{F}_X}} = \mathbb{P}_{\mu^{(1)}_{Y|\mathcal{F}_Y}} &\iff \int K(\cdot, \boldsymbol{x}) \mathbb{P}_{\mu^{(1)}_{X|\mathcal{F}_X}}(d\boldsymbol{x}) = \int K(\cdot, \boldsymbol{y}) \mathbb{P}_{\mu^{(1)}_{Y|\mathcal{F}_Y}}(d\boldsymbol{y}) \\ &\iff \mu^{(2)}_X = \mu^{(2)}_Y.\end{aligned}$$

Since by definition it holds that $\mathscr{D}_K^{(2)}(X,Y) = \|\mu_X^{(2)} - \mu_Y^{(2)}\|_{\mathcal{H}^{(2)}(V)}$, we complete the proof of the first claim in this theorem.

For the second claim, it is easy to see that by definition

$$\mathbb{P}_{X|\mathcal{F}_X} = \mathbb{P}_{Y|\mathcal{F}_Y} \Rightarrow \mathbb{P}_X = \mathbb{P}_Y.$$

Besides, we know that

$$\mathbb{P}_X = \mathbb{P}_Y \iff \mathscr{D}_K^{(1)}(X,Y) = 0,$$

and therefore

$$\mathscr{D}_K^{(2)}(X,Y) = 0 \implies \mathscr{D}_K^{(1)}(X,Y) = 0.$$

Moreover, we refer readers to Hoover and Keisler (1984, Example 3.1) for a simple example which shows that there exist processes X and Y with $\mathscr{D}_K^{(1)}(X,Y) = 0$ but $\mathscr{D}_K^{(2)}(X,Y) > 0$. □

Theorem C.4.2. *$\widehat{\mathscr{D}}_K^{(2)}(X,Y)$ is a consistent estimator for the 2nd order MMD,*

$$\left|\widehat{\mathscr{D}}_K^{(2)}(X,Y) - \mathscr{D}_K^{(2)}(X,Y)\right| \xrightarrow{p} 0 \quad \text{as} \quad m,n \to \infty \tag{C.5}$$

with $\{X_i\}_{i=1}^m \sim X$, $\{Y_i\}_{i=1}^n \sim Y$ and where convergence is in probability.

Proof. Recall that given m independent sample paths $\{\boldsymbol{x}_i\}_{i=1}^m \sim X$, we can use the estimator in eq. (C.4) to approximate sample paths $\{\widetilde{\boldsymbol{x}}_i\}_{i=1}^m$ from the 1st order predictive KME $\mu_{X|\mathcal{F}_X}^{(1)}$. Hence, it suffices to prove the following claim.

Claim: consider m independent sample paths $\{\widetilde{\boldsymbol{x}}_i\}_{i=1}^m \sim \widehat{\mu}_{X|\mathcal{F}_X}^{(1)}$. Then, the estimator given by $\widehat{\mu}_X^{(2)} = \frac{1}{m}\sum_{i=1}^m K(\cdot, \widetilde{\boldsymbol{x}}_i)$ is consistent for the 2nd order predictive KME $\mu_X^{(2)}$, that is

$$\left\|\widehat{\mu}_X^{(2)} - \mu_X^{(2)}\right\|_{\mathcal{H}^{(2)}(V)}^2 \xrightarrow{p} 0, \text{ as } m \to \infty.s \tag{C.6}$$

By the triangular inequality

$$\begin{aligned}
\left\|\widehat{\mu}_X^{(2)} - \mu_X^{(2)}\right\|_{\mathcal{H}^{(2)}(V)}^2 &= \left\|\frac{1}{m}\sum_{i=1}^m K(\cdot, \widetilde{\boldsymbol{x}}_i) - \mathbb{E}\Big[K(\cdot, \mu_{X|\mathcal{F}_X}^{(1)})\Big]\right\|_{\mathcal{H}^{(2)}(V)}^2 && \text{(C.7)}\\
&\leq \left\|\frac{1}{m}\sum_{i=1}^m K(\cdot, \widetilde{\boldsymbol{x}}_i) - \mathbb{E}\Big[K(\cdot, \widehat{\mu}_{X|\mathcal{F}_X}^{(1)})\Big]\right\|_{\mathcal{H}^{(2)}(V)}^2 && \text{(C.8)}\\
&+ \left\|\mathbb{E}\Big[K(\cdot, \widehat{\mu}_{X|\mathcal{F}_X}^{(1)})\Big] - \mathbb{E}\Big[K(\cdot, \mu_{X|\mathcal{F}_X}^{(1)})\Big]\right\|_{\mathcal{H}^{(2)}(V)}^2 && \text{(C.9)}
\end{aligned}$$

The term in eq. (C.8) converges to 0 as $m \to \infty$ by the weak law of large numbers. Therefore, it remains to show that as $m \to \infty$

$$A := \left\| \mathbb{E}_X \left[K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) \right] - \mathbb{E}_X \left[K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \right] \right\|_{\mathcal{H}^{(2)}(V)} \xrightarrow{p} 0 \tag{C.10}$$

First note the following upper bound,

$$A \leq \mathbb{E}_X \left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) - K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \right\|_{\mathcal{H}^{(2)}(V)}.$$

We will show convergence of the right-hand-side. By Park and Muandet (2021, Theorem 3.4), for every $t = 1, \ldots, T$

$$\mathbb{E}_{X|\mathcal{F}_{X_t}} \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} - \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} \xrightarrow{p} 0 \quad \text{as } m \to \infty. \tag{C.11}$$

Now let us assume that the above convergences also hold almost surely for every $t = 1, \ldots, T$. Then by the Egorov's theorem, for any $\delta > 0$, there is a subset Ω_δ with $\mathbb{P}(\Omega_\delta) > 1 - \delta$ and the above convergence eq. (C.11) holds uniformly on Ω_δ. This implies that on Ω_δ for every $\varepsilon > 0$ there is an $N(\varepsilon)$ such that for all $m \geq N(\varepsilon)$ and all $t = 1, \ldots, T$, it holds that

$$\mathbb{E}_{X|\mathcal{F}_{X_t}} \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} - \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} \leq \varepsilon^2. \tag{C.12}$$

From this estimate we immediately obtain by the triangle inequality on Ω_δ

$$\mathbb{E}_{X|\mathcal{F}_{X_t}} \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} \leq 2\mathbb{E}_{X|\mathcal{F}_{X_t}} \left\| \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} + 2\varepsilon^2, \tag{C.13}$$

and, by the Chebyshev's inequality, on Ω_δ, $\forall t = 1, \ldots, T$, $\forall m \geq N(\varepsilon)$

$$\begin{aligned} \mathbb{P}_{X|\mathcal{F}_{X_t}} \left[\left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} - \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|_{\mathcal{H}(V)} > \sqrt{\varepsilon} \right] &\leq \frac{1}{\varepsilon} \mathbb{E}_{X|\mathcal{F}_{X_t}} \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} - \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} \\ &\leq \frac{1}{\varepsilon} \varepsilon^2 = \varepsilon, \end{aligned}$$

which implies that on Ω_δ, $\forall t = 1, \ldots, T$, the sequence $\widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}}$ converges to $\mu^{(1)}_{X|\mathcal{F}_{X_t}}$ in probability with respect to $\mathbb{P}_X$. By a standard result in rough path theory (Lyons, 1998) there exists a universal constant $\beta \in \mathbb{R}$ such that

$$\left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) \right\|_{\mathcal{H}^{(2)}(V)} \leq \beta \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_X} \right\|^{1\text{–var}}_{\mathcal{H}(V)}, \tag{C.14}$$

where $\|\cdot\|^{1\text{–var}}_{\mathcal{H}(V)}$ denotes the total variation norm of paths taking values in $\mathcal{H}(V)$. Since we are in a finite discrete time setup, we have

$$\left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_X} \right\|^{1\text{–var}}_{\mathcal{H}(V)} \leq C(T) \sum_{t=1}^{T} \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} \right\|_{\mathcal{H}(V)}. \tag{C.15}$$

Hence, combining all the above arguments, we can conclude that on Ω_δ and for all $m \geq N(\varepsilon)$,

$$\begin{aligned}
\mathbb{E}_X \left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) \right\|^2_{\mathcal{H}^{(2)}(V)} &\leq \beta^2 \mathbb{E}_X \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_X} \right\|^2_{1\text{-var};\mathcal{H}(V)} \\
&\leq \beta^2 C(T)^2 \sum_{t=1}^{T} \mathbb{E}_X \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} \\
&\leq \beta^2 C(T)^2 \Big(2 \sum_{t=1}^{T} \mathbb{E}_X \left\| \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} + 2T\varepsilon^2 \Big) \\
&\leq C < \infty
\end{aligned}$$

where in the penultimate line we used eq. (C.13). As a result, we obtain that on Ω_δ,

$$\sup_{m \geq N(\varepsilon)} \mathbb{E}_X \left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) - K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \right\|^2_{\mathcal{H}^{(2)}(V)} < \infty \tag{C.16}$$

which in turn implies, by the de la Vallée–Poussin theorem, that on Ω_δ the sequence,

$$\left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) - K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \right\|^2_{\mathcal{H}^{(2)}(V)}$$

$m \geq N(\varepsilon)$ is uniformly integrable for $\mathbb{P}_X$. Thanks to the uniform integrability of the sequence and the continuity of the kernel K,

$$\left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) - K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \right\|_{\mathcal{H}^{(2)}(V)} \to 0$$

in probability for $\mathbb{P}_X$. Then recalling that we have shown that $\widehat{\mu}^{(1)}_{X|\mathcal{F}_X}$ converges to $\mu^{(1)}_{X|\mathcal{F}_X}$ in probability with respect to $\mathbb{P}_X$ as $m \to \infty$, a standard result in probability theory ensures that on Ω_δ,

$$\mathbb{E}_X \left\| K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) - K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \right\|_{\mathcal{H}^{(2)}(V)} \to 0, \text{ as } m \to \infty$$

which implies that,

$$\left\| \mathbb{E}_X \Big[K(\cdot, \widehat{\mu}^{(1)}_{X|\mathcal{F}_X}) \Big] - \mathbb{E}_X \Big[K(\cdot, \mu^{(1)}_{X|\mathcal{F}_X}) \Big] \right\|_{\mathcal{H}^{(2)}(V)} \to 0, \text{ as } m \to \infty$$

on Ω_δ. Clearly, as δ was arbitrary, we have

$$\mathbb{E}_{X|\mathcal{F}_{X_t}} \left\| \widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} - \mu^{(1)}_{X|\mathcal{F}_{X_t}} \right\|^2_{\mathcal{H}(V)} \to 0 \quad \text{as } m \to \infty$$

almost surely. Finally, note that the above result holds true for any subsequence of $\left(\widehat{\mu}^{(1)}_{X|\mathcal{F}_{X_t}} \right)_{m \geq 1}$ (because every sequence converging in probability has a subsequence converging almost surely), which proves the desired result eq. (C.6). □

Theorem C.4.3. *Given two stochastic processes* X, Y

$$\mathscr{D}_K^{(n)}(X,Y) = 0 \implies \mathscr{D}_K^{(k)}(X,Y) = 0 \quad \textit{for any} \;\; 1 < k < n \tag{C.17}$$

but the converse is not generally true.

Proof. Let $X \in \mathcal{X}(V)$, then we denote $\mathbb{P}_{X|\mathcal{F}_{X_t}} =: X_t^{(1)}$. Then we continue this procedure and define $X_t^{(n)} := \mathbb{P}_{X^{(n-1)}|\mathcal{F}_{X_t}}$ (it is called the rank n prediction process in Bonnier et al. (2020)). Now we can apply the same argument as in the proof of Theorem Thm. C.4.1 together with an induction procedure easily prove that

$$\mathscr{D}_K^{(n)}(X,Y) = 0 \iff \mathbb{P}_{X^{(n-1)}} = \mathbb{P}_{Y^{(n-1)}} \tag{C.18}$$

for all $n > 1$. From the definition of these processes $X^{(n)}$ and $Y^{(n)}$ we can immediately see that $\mathbb{P}_{X^{(n)}} = \mathbb{P}_{Y^{(n)}}$ ensures that $\mathbb{P}_{X^{(k)}} = \mathbb{P}_{Y^{(k)}}$ for all $k < n$, which yields the desired result. We refer readers to Hoover and Keisler (1984, Example 3.2) for examples which illustrate that for each n there exist processes X and Y with $\mathscr{D}_K^{(n)}(X,Y) = 0$ (equivalently, $\mathbb{P}_{X^{(n-1)}} = \mathbb{P}_{Y^{(n-1)}}$) but $\mathscr{D}_K^{(n+1)}(X,Y) > 0$ (equivalently, $\mathbb{P}_{X^{(n)}} \neq \mathbb{P}_{Y^{(n)}}$). $\square$

Remark. Using terminologies from Hoover and Keisler (1984) and Bonnier et al. (2020), $\mathbb{P}_{X^{(n)}} = \mathbb{P}_{Y^{(n)}}$ means that processes X and Y have the same adapted distribution up to rank n. Therefore Thm. C.4.1 and Thm. C.4.3 tell us that $\mathscr{D}_K^{(n)}(X,Y) = 0$ if and only if they ave the same adapted distribution up to rank n. Moreover, using the partial isometry between the RKHS generated by K and the tensor algebra, see e.g. Chevyrev and Oberhauser (2018, Theorem E.2), one can use an induction argument to verify that $\mathscr{D}_K^{(n)}$ coincides with the metric d_{n-1} defined in Bonnier et al. (2020, Definition 14), and therefore by Bonnier et al. (2020, Theorem 4) we can obtain a stronger result that $\mathscr{D}_K^{(n)}$ actually metrizes the so–called rank $n-1$ adapted topology (see Bonnier et al. (2020, Definition 5), Hoover and Keisler (1984, Definition 2.25)). For more details regarding adapted topologies we refer to Bonnier et al. (2020).

Theorem C.4.4. *Let* $f : \mathbb{R} \to \mathbb{R}$ *be a globally analytic function with non-negative coefficients. Define the family of kernels* $K_{dr}^{(n)} : \mathcal{P}(\mathcal{X}(V)) \times \mathcal{P}(\mathcal{X}(V)) \to \mathbb{R}$ *as follows*

$$K_{dr}^{(n)}(X,Y) = f(\mathscr{D}_K^{(n)}(X,Y)), \quad n \in \mathbb{N}_*. \tag{C.19}$$

Then the RKHS associated to $K_{dr}^{(n)}$ *is dense in the space of functions from* $\mathcal{P}(\mathcal{X}(V))$ *to* $\mathbb{R}$ *which are continuous with respect to the* k^{th} *order MMD for any* $1 \leq k \leq n$.

Proof. By Christmann and Steinwart (2010, Thm. 2.2) if K is a compact metric space and H is a separable Hilbert space such that there exists a continuous (w.r.t. a topology τ on K) and injective map $\rho : K \to H$, then for any globally analytic function with non-negative coefficients $f : \mathbb{R} \to \mathbb{R}$ the kernel $k : K \times K \to \mathbb{R}$ given by

$$k(z, z') = f\left(\left\|\rho(z) - \rho(z')\right\|_H\right) \tag{C.20}$$

is universal in the sense that its RKHS is τ-dense in the space of τ-continuous functions from K to $\mathbb{R}$. By assumption, $\mathcal{X}(V)$ is a $\mathscr{D}_K^{(1)}$-compact metric space, therefore by Thm. C.4.3 it is also $\mathscr{D}_K^{(n)}$-compact for every $n \geq 1$. Hence, by Walkden (2014, Thm. 10.2) the set of stochastic processes $\mathcal{P}(\mathcal{X}(V))$ is also $\mathscr{D}_K^{(n)}$-compact. To show that $\rho : X \mapsto \mu_X^{(n)}$ is injective and continous with respect to $\mathscr{D}_K^{(n)}$ we refer readers to Bonnier et al. (2020, Proposition 4). Furthermore, $\mathcal{H}^{(n)}(V)$ can be shown by induction to be a Hilbert space with a countable basis, hence it is separable. Setting $K = \mathcal{P}(\mathcal{X}(V)),\ H = \mathcal{H}^{(n)}(V)$ and $\rho : X \mapsto \mu_X^{(n)}$ concludes the proof. $\square$

Appendix D

Appendix to Chapter 6

This appendix is organized as follows. In Appendix D.1 we provide a summary of the computational aspects of SPDEs used for data simulation and model definition, emphasizing the important role of the Fourier Transform (Appendix D.1.1) for simulating noise realizations of Wiener processes (Appendix D.1.2) and building numerical solvers for SPDEs (Appendix D.1.3). In Appendix D.2 we provide further experimental details (Appendix D.2.1) and additional experiments on the stochastic Ginzburg-Landau (Appendix D.2.2) and wave (Appendix D.2.3) equations.

D.1 Computational aspects of SPDEs

We start this section with the definition of the *Fourier Transform* (FT). We then define the *Discrete Fourier Transform* (DFT) as an approximation to the FT of a function observed at finitely many locations. Next, we discuss the role played by the FT to sample realizations of Wiener processes, necessary to build spectral solvers for SPDEs. The interested reader is referred to Briggs and Henson (1995) and Lord et al. (2014) for further details.

D.1.1 The Fourier Transform

Let V be a vector space over the complex numbers (e.g. $\mathbb{C}^{d_h}$ or $\mathbb{C}^{d_h \times d_h}$). Let $r \in \mathbb{N}_*$ and let $\mathcal{C} \subset \mathbb{R}^r$ be a compact subset of $\mathbb{R}^r$. In Chapter 6 we used either $r = d$ and $\mathcal{C} = \mathcal{D}$ or $r = d + 1$ and $\mathcal{C} = [0, T] \times \mathcal{D}$.

Definition 20 (*r*-**dimensional Fourier Transform**). *The r-dimensional FT $\mathcal{F}_r : L^2(\mathbb{R}^r, V) \to L^2(\mathbb{R}^r, V)$ and its inverse $\mathcal{F}_r^{-1} : L^2(\mathbb{R}^r, V) \to L^2(\mathbb{R}^r, V)$ are defined as follows*

$$\mathcal{F}_r(f)(\mathbf{y}) = \int_{\mathbb{R}^r} e^{-2\pi i \langle \mathbf{x}, \mathbf{y} \rangle} f(\mathbf{x}) d\mathbf{x}, \quad \mathcal{F}_r^{-1}(g)(\mathbf{x}) = \int_{\mathbb{R}^r} e^{2\pi i \langle \mathbf{x}, \mathbf{y} \rangle} g(\mathbf{y}) d\mathbf{y},$$

for any $f, g \in L^2(\mathbb{R}^r, V)$, where $i = \sqrt{-1}$ is the imaginary unit and $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product on $\mathbb{R}^r$.

In practice, we do not observe a function on $\mathbb{R}^r$ but on a subset $\mathcal{C} \subset \mathbb{R}^r$. Furthermore, functions are observed at finitely many locations in $\mathcal{C}$, and another transform—the discrete Fourier transform (DFT)—is used for numerical computations.

In the sequel we denote by Π_N the set of periodic sequences indexed on $\mathbb{Z}_r$ with period vector $(N_1, \dots, N_r)$.

Definition 21 (**r-dimensional Discrete Fourier Transform**). *The r-dimensional DFT $\mathcal{D}_r : \Pi_N \to \Pi_N$ and its inverse $\mathcal{D}_r^{-1} : \Pi_N \to \Pi_N$ are defined as follows,*

$$\mathcal{D}_r(u)_{\mathbf{n}} = \sum_{k \in \mathbb{Z}^r \cap \mathcal{R}_N} u_{\mathbf{k}} e^{-2\pi i \langle \mathbf{n}, \mathbf{N}^{-1}\mathbf{k} \rangle}, \quad \mathcal{D}_r^{-1}(v)_{\mathbf{k}} = \frac{1}{|\det \mathbf{N}|} \sum_{\mathbf{n} \in \mathbb{Z}^r \cap \mathcal{R}_N} v_{\mathbf{n}} e^{2\pi i \langle \mathbf{n}, \mathbf{N}^{-1}\mathbf{k} \rangle},$$

with $\mathbf{N} = \operatorname{diag}(N_1, \dots, N_r) \in \mathbb{N}^{r \times r}$, and $\mathcal{R}_N$ the rectangular domain $\mathcal{R}_N = \{\mathbf{x} \in \mathbb{R}^r \mid 0 \le x_i < N_i,\ i = 1, \dots, r\}$.

The DFT of a sequence can be computed exactly and efficiently using the *fast Fourier transform* (FFT) algorithm (Cooley and Tukey, 1965) which reduces the complexity from $\mathcal{O}(M^2)$ to $\mathcal{O}(M \log M)$ where $M = N_1 N_2 \dots N_r$. Most importantly, the FFT algorithm is implemented in machine learning libraries such as PyTorch, which provide support for GPU acceleration and automatic differentiation capabilities.

Note that if we have a finite sequence, we may still define its DFT by implicitly extending the sequence periodically. In particular, when a compactly supported function is sampled on its interval of support, and the samples are used as input for a DFT, it is as if the periodic extension of the function had been sampled. More precisely, consider an input sequence which corresponds to the evaluation of a function f on a regular grid of $\mathcal{C} = \mathcal{R}_N$. For simplicity, suppose that $N_i = N_1$ for all $i = 1, \dots, r$ and consider the grid points $\mathbf{x}_{\mathbf{n}} = \mathbf{n}L/N_1$ for $\mathbf{n} \in \mathbb{Z}^r \cap \mathcal{R}_N$. Taking the DFT of the sequence of general term $u_{\mathbf{n}} = f(\mathbf{x}_{\mathbf{n}})$ we obtain for all $\mathbf{n} \in \mathbb{Z}^r$

$$\mathcal{D}_r(u)_{\mathbf{n}} = \sum_{\mathbf{k} \in \mathbb{Z}^r \cap \mathcal{R}_N} u_{\mathbf{k}} e^{-2\pi i \langle \mathbf{n}, \mathbf{k}/N_1 \rangle} = \sum_{\mathbf{k} \in \mathbb{Z}^r \cap \mathcal{R}_N} f(\mathbf{x}_{\mathbf{k}}) e^{-2\pi i \langle \mathbf{y}_{\mathbf{n}}, \mathbf{x}_{\mathbf{k}} \rangle},$$

where $\mathbf{y}_{\mathbf{n}}$ are the reciprocal frequency points given by $\mathbf{y}_{\mathbf{n}} = \mathbf{n}/L$ for $\mathbf{n} \in \mathbb{Z}^r \cap \mathcal{R}_N$. The DFT of a compactly supported (or approximately compactly supported) function f sampled on the regular grid of points $\mathbf{x}_{\mathbf{k}}$ approximates the FT of f at the frequency points $\mathbf{y}_{\mathbf{n}}$ (up to a constant multiplicative factor).

The FT is closely related to the notions of *Fourier coefficients* and *Fourier Series* defined hereafter.

Definition 22 (*r*-**dimensional Fourier series**). *Let f be a piecewise smooth function $f : \mathbb{R}^r \to V$ which is periodic in x_i with period $L_i \in \mathbb{R}_+$ for all $i = 1, \ldots, r$. The r-dimensional Fourier series of f is a representation of the form,*

$$f(\mathbf{x}) \sim \sum_{\mathbf{n}\in\mathbb{Z}^r} c_{\mathbf{n}}(f) e^{2\pi i\langle \mathbf{L}^{-1}\mathbf{n}, \mathbf{x}\rangle},$$

where $\mathbf{L} = \operatorname{diag}(L_1, \ldots, L_r) \in \mathbb{R}^{r\times r}$ and $c_{\mathbf{n}}(f)$ are complex coefficients, called Fourier coefficients, given by

$$c_{\mathbf{n}}(f) = \frac{1}{|\det\mathbf{L}|}\int_{\mathcal{R}_L} e^{-2\pi i\langle \mathbf{L}^{-1}\mathbf{n}, \mathbf{x}\rangle} f(\mathbf{x})d\mathbf{x}, \quad n \in \mathbb{Z}^r$$

where $\mathcal{R}_L \subset \mathbb{R}^r$ denotes the rectangular domain of sides $L_1, \ldots, L_r$.

We note that in the definition above, the sign $\sim$ means that the series is a formal series and no statement is made about the convergence of the series (the forms of convergence are studied in Alimov et al. (1992)). If f is compactly supported on $\mathcal{R}_L$, we may still define its Fourier coefficients, and in this case $\mathcal{F}_r(f)(\mathbf{y_n}) = |\det\mathbf{L}|c_{\mathbf{n}}(f)$ at the frequency points $y_{\mathbf{n}} = \mathbf{L}^{-1}\mathbf{n}$.

Numerical consideration

Consider a function f which has compact support (or is periodic) which is observed at M locations in its support (or its unitary cell $\mathcal{R}_L$). When using the DFT to approximate M points of the spectrum $\mathcal{F}_r(f)(\mathbf{y_k})$ (or M coefficients $c_{\mathbf{k}}(f)$), a so-called *aliasing* error usually occurs: due to the periodicity of the DFT, the $\mathbf{k}^{\text{th}}$ coefficient of the DFT includes the contributions not only of the $\mathbf{k}^{\text{th}}$ frequency mode, but also from higher modes of the underlying function f. In general the accuracy of the highest frequency modes is more impacted by this error, and aliasing occurs specifically when we compute nonlinear terms in the physical space. For example, in Chapter 6 we approximate the evaluation on the grid $D \times \mathcal{T}$ of $\mathcal{F}^{-1}_{d+1}(\mathcal{F}_{d+1}(\mathcal{K})\mathcal{F}_{d+1}(f))$ by $\mathcal{D}^{-1}_{d+1}(B\mathcal{D}_{d+1}(f|_{D\times\mathcal{T}}))$ where $f = \mathbb{1}_{s\geq 0}H_{\theta,\xi}(z)$ and $H_{\theta,\xi}$ is nonlinear. One possibility to mitigate aliasing is to set to zero the DFT terms (arising in nonlinearities) corresponding to the highest frequency modes before we apply the inverse DFT to go back to the physical space. This is precisely what we do when we parametrize only $k^1_{\max} \times \ldots \times k^{d+1}_{\max} \times d_h \times d_h$ entries of the complex tensor B, and set the others to zero, hence resolving potential aliasing errors. We note that specific rules have been proposed (notably in the literature on pseudo-spectral solvers) to deal with specific nonlinearities. However, in the context of Neural SPDE we learn the nonlinearities, hence the number of frequency modes that we retain is treated as a hyperparameter.

D.1.2 Stochastic simulation of Wiener processes

After defining Wiener processes we outline the sampling procedure that we used to simulate the datasets in Chapter 6. For more details on computational aspects of SPDEs the reader is referred to Lord et al. (2014).

Throughout this section, H will denote a separable Hilbert space (e.g. $H = L^2(\mathcal{D})$) with a complete orthonormal basis $\{\phi_k\}_{k\in\mathbb{N}}$. Let $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$ be a filtered probability space.

Q-Wiener process

Consider an operator $\mathcal{Q} : H \to H$ such that there exists a bounded sequence of nonnegative real numbers $\{\lambda_k\}_{k\in\mathbb{N}}$ such that $Q\phi_k = \lambda_k\phi_k$ for all $k \in \mathbb{N}$ (this is implied by Q being a trace class, non-negative, symmetric operator, for example).

Definition 23 (**Q-Wiener process**). *Let Q be a trace class non negative, symmetric operator on H. A H-valued stochastic process $\{W(t) : t \geq 0\}$ is called a Q-Wiener process if*

1. *$W(0) = 0$ almost surely;*
2. *$W(t;\omega)$ is a continuous sample trajectory $\mathbb{R}^+ \mapsto H$, for each $\omega \in \Omega$;*
3. *$W(t)$ is $\mathcal{F}_t$-adapted and has independent increments $W(t) - W(s)$ for $s < t$;*
4. *$W(t) - W(s) \sim \mathcal{N}(0, (t-s)Q)$ for all $0 \leq s \leq t$.*

In analogy to the Karhunen Loéve expansion, it can be shown that $W(t)$ is a Q-Wiener process if and only if for all $t \geq 0$

$$W(t) = \sum_{j=1}^{\infty} \sqrt{\lambda_j}\phi_j\beta_j(t), \tag{D.1}$$

where $\beta_j(t)$ are i.i.d. Brownian motions, and the series converges in $L^2(\Omega, H)$. Moreover the series is $\mathbb{P}$-a.s. uniformly convergent on $[0, T]$ for arbitrary $T > 0$. (i.e. converges in $L^2(\Omega, \mathcal{C}([0, T], H))$).

In the Navier-Stokes example, we drive the SPDE by samples ξ from a Q-Wiener process in two dimensions. Here we follow Lord et al. (2014, Example 10.12) and explain how the sampling procedure works in this case. Let $D = (0, L_1) \times (0, L_2)$ and consider an $L^2(D)$-valued Q-Wiener process $W(t)$. If the eigenfunctions of Q are given by

$$\phi_k(x) = \frac{1}{\sqrt{L_1 L_2}} e^{2i\pi(k_1 x_1/L_1 + k_2 x_2/L_2)},$$

numerical approximation of sample paths from $W(t)$ are easy to obtain through a DFT. Denote by λ_k the eigenvalues of Q (e.g. $\lambda_k = e^{-\alpha|k|^2}$ for some parameter $\alpha > 0$) and let $\mathcal{J}$ be the index set defined by

$$\mathcal{J} := \{(j_1, j_2) \in \mathbb{Z}^2 \ : -J_1/2 + 1 \leq j_1 \leq J_1/2, \ -J_2/2 + 1 \leq j_2 \leq J_2/2\}.$$

The goal is to sample from the truncated expansion of $W(t)$

$$W_J(t) = \sum_{j \in \mathcal{J}} \sqrt{\lambda_j} \phi_j \beta_j(t),$$

at the collection of sample points

$$x_k = (L_1 k_1/J_1, L_2 k_2/J_2)', \qquad 0 \leq k_1 \leq J_1 - 1, \ 0 \leq k_2 \leq J_2 - 1.$$

Consider the random variable $Z(t_n, x)$ defined by

$$Z(t_n, x) = \sqrt{\Delta t} \sum_{j \in \mathcal{J}} \sqrt{\lambda_j} \phi_j(x) \xi_j^n, \qquad \xi_j^n \sim \mathbb{C}\mathcal{N}(0, 2),$$

meaning that $\xi_j^n = a + ib$ with $a, b \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, 1)$ such that $Z(t_n, x_k)$ is a complex random variable with independent real and imaginary part with the same distribution as two independent copies of the increment $W_J(t_n + \Delta t, x_k) - W_J(t_n, x_k)$. Furthermore, $Z(t_n, x_k)$ can be expressed in the form

$$Z(t_n, x_k) = \frac{1}{J_1 J_2} \sum_{j_1 = -J_1/2+1}^{J_1/2} \sum_{j_2 = -J_2/2+1}^{J_2/2} \widetilde{Z}_{j_1, j_2} e^{2i\pi\left(j_1 \frac{k_1}{J_1} + j_2 \frac{k_2}{J_2}\right)}, \tag{D.2}$$

where $\widetilde{Z}_{j_1,j_2} = \sqrt{\Delta t \lambda_{j_1,j_2}} J_1 J_2 \xi^n_{j_1,j_2}$ We recognize that the matrix with entries given by eq. (D.2) is the 2D inverse DFT of the $J_1 \times J_2$ matrix with entries $\widetilde{Z}_{j_1,j_2}$. Therefore, we can sample two independent copies of

$$W_J(t_n + \Delta t, x_k) - W_J(t_n, x_k), \ \ 0 \leq k_1 \leq J_1 - 1, \ 0 \leq k_2 \leq J_2 - 1,$$

by computing a single 2D inverse DFT.

Cylindrical Wiener process

If the operator $Q = I$ is the identity, then Q is not of trace class on H so that the series in eq. (D.1) does not converge in $L^2(\Omega, H)$. This motivates the definition of cylindrical Wiener processes.

Definition 24 (**Cylindrical Wiener process**). *Let H be a separable Hilbert space. A cylindrical Wiener process (a.k.a space-time white noise) is a H-valued* *stochastic process* $\{W(t) : t \geq 0\}$ *defined by*

$$W(t) = \sum_{j=1}^{\infty} \phi_j \beta_j(t), \tag{D.3}$$

where $\{\phi_j\}$ *is any orthonormal basis of* H *and* $\beta_j(t)$ *are i.i.d. Brownian motions.*

In all examples except Navier-Stokes, we drive the SPDE by samples ξ from a cylindrical Wiener process in one dimension. Let $D = (0, L)$ and consider an $L^2(D)$-valued cylindrical Wiener process $W(t)$. As explained in Lord et al. (2014, Example 10.31), if we take the basis

$$\phi_k(x) = \sqrt{2/L} \sin(k\pi x/L),$$

numerical approximation of sample paths from $W(t)$ are easy to obtain. The goal is to sample from the truncated expansion

$$W_J(t) = \sum_{j=1}^{J} \phi_j \beta_j(t), \tag{D.4}$$

at the collection of sample points $x_k = kL/J$ for $k = 1, \dots, J$. Observing that a trigonometric identity yields

$$\operatorname{Cov}(W_J(t, x_i), W_J(t, x_k)) = (tL/J)\delta_{ik}, \qquad i, k = 1, \dots, J,$$

the increments $W_J(t_n + \Delta t, x_k) - W_J(t_n, x_k) \sim \mathcal{N}(0, \Delta t L/J)$ for all $k = 1, \dots, J$.

D.1.3 Numerical solvers

In this section we present an overview of the numerical solvers for SPDEs we used to generate the data for all the experiments. The stochastic Ginzburg-Landau (Sec. 6.5.1 and appendix D.2.2), stochastic wave (Appendix D.2.3) equations have been solved using the finite difference method, while the stochastic Korteweg–De Vries (Sec. 6.5.2) and Navier Stokes (Sec. 6.5.3) equations have been solved using the spectral Galerkin method. We use the same setup as in Sec. 4.2. In particular, we focus on stochastic semilinear evolution equations of the form

$$du(t) = (\mathcal{L}u(t) + F(u(t)))\,dt + G(u(t))dW(t), \tag{D.5}$$

where $W(t)$ is either a Q-Wiener process or a cylindrical Wiener process and $\mathcal{L}$ is a linear differential operator generating a semigroup $e^{t\mathcal{L}}$. We consider nonlinearities F, G regular enough (see Lord et al. (2014, Assumption 10.23)) to guarantee existence and uniqueness of mild solutions of eq. (D.5) (Lord et al., 2014, Theorem 10.26).

Finite difference method

We illustrate this numerical method for the reaction-diffusion equation

$$du(t) = \left(\epsilon\partial_{xx}^2 u(t) + F(u(t))\right) dt + \sigma dW(t), \quad u(0,x) = u_0(x),$$

with homogeneous Dirichlet boundary conditions and where $\epsilon, \sigma > 0$ are constants. We assume for simplicity that $u_0, u(t), W(t)$ are real-valued and $\mathcal{D} = (0, a)$. The generalization to higher dimensions is straightforward.

Consider the grid points $x_j = jh$, where $h = \frac{a}{J}$ and $j = 0, ..., J$, for some spatial resolution $J \in \mathbb{N}$. Let $u_J(t)$ be the *finite difference approximation* of $[u(t, x_1), ..., u(t, x_{J-1})]$ (similarly for $W_J(t)$) resulting from the solution of the following SDE

$$du_J(t) = [-\epsilon M u_J(t) + \hat{f}(u_J(t))]dt + \sigma dW_J(t),$$

where $\hat{f}(u_J) = [f(u_1), ..., f(u_{J-1})]'$ and M is the $(J-1) \times (J-1)$ matrix approximating Laplacian (with free boundary conditions) which is given by

$$M = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}.$$

One could modify M for specific boundary conditions. For instance in the case of periodic boundary one should modify $M_{1,J-1} = M_{J-1,1} = -1$ (see Lord et al. (2014, Chapter 3.4) for Dirichlet and Neuman boundary condition modifications of M). To discretize in time, we may apply numerical methods for SDEs (see for example Lord et al. (2014, Chapter 8)). Choosing the standard Euler-Marayama scheme with time step Δt yields an approximation $u_{J,n}$ to $u_J(t_n)$ at $t_n = n\Delta t$ defined by

$$u_{J,n+1} = (I + \Delta t \epsilon M)^{-1}\left(u_{J,n} + \hat{f}(u_{J,n})\Delta t + \sigma(W_J(t_{n+1}) - W_J(t_n))\right)$$

The increments $(W_J(t_{n+1}) - W_J(t_n))$ are generated using techniques discussed in Appendix D.1.2.

Spectral Galerkin method

Consider again a separable Hilbert space H. Assume that the differential operator $\mathcal{L}$ in eq. (D.5) has a complete set of orthonormal eigenfunctions $\{\phi_j\}_{j\in\mathbb{N}}$ and eigenvalues $\lambda_j < 0$, ordered so that $\lambda_{j+1} < \lambda_j$. Then, we can define the semigroup $e^{t\mathcal{L}}$ as follows

$$e^{t\mathcal{L}}h = \sum_{j=1}^{\infty} e^{\lambda_j t}\langle h, \phi_j\rangle \phi_j, \quad h \in H.$$

Define the *Galerkin subspace* $V_J = \mathrm{Span}\{\phi_1, ..., \phi_J\}$ and the orthonormal projections $P_J : H \to V_J$ as follows

$$P_J h = \sum_{i=1}^{J} \langle u, \phi_j\rangle \phi_j, \quad h \in H.$$

Then, the following defines *spectral Galerkin approximation* of eq. (D.5)

$$du_J(t) = (\mathcal{L}_J u_J(t) + P_J F(u_J(t)))dt + P_J G(u_J(t))dW_J(t), \quad u_J(0) = P_J u_0$$

where $u_J := P_J u$ and $\mathcal{L}_J := P_J \mathcal{L}$ and $W_J = P_J W$ is as in eq. (D.4). Using a Euluer-Marayama discretization as above, we obtain the following discretization

$$u_{J,n+1} = (I + \Delta t \mathcal{L}_J)^{-1}(u_{J,n} + \Delta t P_J F(u_{J,n}) + P_J G(u_{J,n})\Delta W_{J,n}).$$

This approach is particularly convenient for problems with additive noise where the eigenfunctions of $\mathcal{L}$ and Q (the covariance of the Q-Wiener process W) are equal, which is the case for all the experiments in Chapter 6 generated with this method. The eigenfunctions of the Laplacian with periodic boundary conditions correspond to the Fourier basis exponentials; therefore, one can define the projection P_J in terms of the DFT.

D.2 Further experiments

We start this section with additional details on the training of NSPDE and the baseline models, including how the relevant hyperparameters have been selected for each model.

D.2.1 Additional experimental details

For all experiments the dataset is split into a training, validation and test sets with relative sizes 70%/15%/15%. For all models, a grid search on the hyperparameters is performed using the training and validation sets. We use the Adam optimizer and a scheduler which reads the validation loss and reduces the learning rate if no improvement is seen for a *patience* number of epochs. Additionally, an early stopping method is used to halt the training of the model if no improvement is seen after a *patience* number of epochs. The hyperparameters included in the grid search are stated below.

NSPDE The hyperparameters included in the grid search are the number of frequency modes used to parametrize the kernel in Fourier space $B = \mathcal{F}_{d+1}(\mathcal{K})$ and the number of forward iterations used to solve the fixed point problem.

FNO The hyperparameters included in the grid search are the number of frequency modes used to parametrize the kernel and the number of layers M. Note that the numbers of frequency modes in the grid search differ from the ones used for the NSPDE model by a factor 2 to ensure that the effective number of retained modes is the same. For both the NSPDE model and FNO, we kept the number of hidden channels fixed to $d_h = 32$ as this systematically yielded better performances than previously included values and enabled to perform the grid search in a reasonable time.

DeepONet The *Deep Operator Network* (DeepONet) (Lu et al., 2021) is another popular class of neural network models for learning operators on function spaces. The DeepONet architecture is based on the universal approximation theorem of Chen and Chen (1995). It consists of two sub-networks referred to as the *branch* and the *trunk* networks. The trunk acts on the coordinates $(t, x) \in [0, T] \times \mathcal{D}$, while the branch acts on the evaluation of the initial condition u_0 on a discretized grid D. Therefore, the DeepONet is not a space resolution-invariant architecture. The output of the network is expressed as

$$\text{DeepONet}(u_0)(t, x) = \sum_{k=1}^{p} b_k(u_0)\tau_k(t, x) + b_0,$$

where the b_k and the τ_k are the outputs of the branch and trunk network respectively. The trunk network is usually a feedforward neural network, and one can chose the architecture of the branch network depending on the structure of the input domain. We follow Lu et al. (2021) and use feedforward neural networks for both the trunk and the branch networks. We perform a grid search on the depth and width of the trunk and branch feedforward neural networks.

NRDE/NCDE/NCDE-FNO The hyperparameters included in the grid search are the number of hidden channels and the type of solver as implemented by torchdiffeq (Chen et al., 2018). We note that we used a depth-2 NRDE model (depth-2 already results in $d_\xi = 8\,385$ for forcings observed at 128 spatial points and higher depths models could not fit in memory) and recall that NCDE is a depth-1 NRDE.

D.2.2 Stochastic Ginzburg-Landau equation

Recall that the stochastic Ginzburg-Landau equations are of the form

$$\begin{aligned}\partial_t u - \Delta u &= 3u - u^3 + G(u)\xi, \\ u(0,x) &= u_0(x), \quad (t,x) \in [0,T] \times [0,1],\end{aligned} \tag{D.6}$$

subject to either Periodic or Dirichlet boundary conditions. Periodic boundary conditions are given by $u(t,0) = u(t,1)$ for all $t \geq 0$ and Dirichlet boundary conditions are given by $u(t,0) = u(t,1) = 0$ for all $t \geq 0$. Initial condition we take as in Sec. 6.5.1 $u_0(x) = x(1-x) + \kappa\eta(x)$ with $\kappa = 0$ or $\kappa = 0.1$ depending on a task. In both Periodic and Dirichlet case we can take $\eta(x)$ as in eq. (6.8) though in Dirichlet case one must take $a_0 = 0$ to ensure u_0 being zero at the boundary.

We first reproduce an experiment Sec. 6.5.1 on the additive stochastic Ginzburg-Landau equation but with Dirichlet boundary conditions instead of the periodic. We compare it to the benchmark of FNO model which was the most successful among all the benchmarks of Sec. 6.5. From Table D.1 we see that even though Neural SPDE model depends on the spectral methods the errors did not increase compared to the periodic equation in Sec. 6.5.1 (see Table 6.1). Our algorithm still outperforms FNO whose relative $L2$ error increased slightly. The fact that Neural SPDE can be applied to non-periodic equations could be perhaps explained by interpolation (L_θ) and projection (Π_θ) neural networks that could correct for non-periodicity of the data.

Table D.1: **Additive stochastic Ginzburg-Landau equation with homogeneous Dirichlet boundary conditions**. The experimental setup is the same as in Chapter 6. We report the relative L2 error on the test set. The symbol x indicates that the model is not applicable. N is fixed to 1 000.

Model	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
FNO	0.132	0.023	x
NSPDE (Ours)	0.135	0.008	0.010

We now take a look at the specific hyperparameter: number of forward iterations in the fixed point solver. We also call this a number of Picard iterations P. Theoretically as P increases Fixed Point Solver should converge to the true solution (see Hairer (2009)). This suggests that higher P should improve the performance of the Neural SPDE algorithms. In practise we observed in both additive Ginsburg Landau equation from Sec. 6.5.1 and in KdV equation from Sec. 6.5.2 that $P = 1$ could already be enough. This could be explained either by dominance of the linear part of the equation or by overfitting in these cases. Thus we present an experiment on the multiplicative stochastic Ginzbug-Landau equation over a longer (compared to Sec. 6.5.1) time interval. In Table D.2 we compare NSPDE with $P \in \{1, 2, 3, 4\}$ and again include FNO benchmark (which performed best in the previous experiments). We see that NSPDE with even $P = 1$ outperforms FNO. Relative $L2$ error for $T = 0.05$ increases for both NSPDE and FNO due to more complicated multiplicative noise. In Table D.2 we present for each P the best result over other hyperparameters obtained by cross validation. One could clearly see an improvement in error as we increase the number of Picard iterations P (with an exception of the case $T = 0.05$ where $P = 3$ outperformed $P = 4$). This improvement becomes more apparent as the time frame T increases. Heuristically (and qualitatively) this is due to the fact that for the short times solution of the SPDE is relatively close to its linearised version and that nonlinearity of the equation starts to play a bigger role for larger T.

Table D.2: **Multiplicative stochastic Ginzburg-Landau equation**. We report the relative L2 error on the test for FNO and NSPDE (Ours) for different number of Picard iterations on the task $\xi \to u$.

Time	FNO	Ours $P = 1$	Ours $P = 2$	Ours $P = 3$	Ours $P = 4$
$T = 0.05$	0.040	0.023	0.018	**0.016**	0.017
$T = 0.10$	0.068	0.042	0.041	**0.040**	**0.040**
$T = 0.25$	0.105	0.079	0.077	0.073	**0.072**

D.2.3 The stochastic wave equation

In this section we consider the following nonlinear wave equation with multiplicative stochastic forcing,

$$\begin{aligned} \partial_t^2 u - \Delta u &= \cos(\pi u) + u^2 + u\xi, \\ u(t,0) &= u(t,1), \\ u(0,x) &= u_0(x), \\ \partial_t u(0,x) &= v_0(x), \quad (t,x) \in [0,T] \times [0,1]. \end{aligned} \tag{D.7}$$

The nonlinear stochastic wave equation arises in relativistic quantum mechanics and is also used in simulations of nonlinear waves that are subject to either noisy observations or random forcing. We refer a reader to Temam (2012) for an overview on the nonlinear wave equation. The above equation can put in a form of eq. (6.1) by rewriting it as a system for $(u, v) = (u, \partial_t u)$. To generate training datasets, we solve the SPDE using a finite difference method with 128 evenly distanced points in space and a time step size $\Delta t = 10^{-3}$. As in Chevyrev et al. (2021, equation (3.5)), we solve the SPDE until $T = 0.5$. We then downsample the temporal resolution by a factor 5, resulting in 100 time points. Here, the initial condition is given by $u_0(x) = \sin(2\pi x) + \kappa\eta(x)$, where η is defined in eq. (6.8) and for simplicity initial velocity v_0 is taken deterministic $v_0(x) = x(1 - x)$. Similarly to Sec. 6.5.1 we either take $\kappa = 0$ or $\kappa = 1$ to generate datasets where the initial condition is either fixed or varies across samples. Each dataset consists of $N = 1\,000$ training observations.

Table D.3: **Stochastic Wave equation**. We report the relative L2 error on the test set. The symbol x indicates that the model is not applicable. N is fixed to 1 000.

Model	$u_0 \mapsto u$	$\xi \mapsto u$	$(u_0, \xi) \mapsto u$
NCDE	x	0.142	0.432
NRDE	x	0.146	0.445
NCDE-FNO	x	0.029	0.037
DeepONet	0.190	0.143	x
FNO	0.151	0.026	x
NSPDE (Ours)	0.150	**0.023**	**0.026**

Bibliography

Clement John Adkins and Clement John Adkins. *Equilibrium Thermodynamics*. Cambridge University Press, 1983.

Ahmed M Alaa and Mihaela van der Schaar. Attentive state-space modeling of disease progression. *Advances in Neural Information Processing Systems*, 32, 2019.

D. J. Aldous. Weak convergence and general theory of processes. *Unpublished draft of monograph*, 1981.

Sh A Alimov, RR Ashurov, and AK Pulatov. Multiple fourier series and fourier integrals. In *Commutative Harmonic Analysis IV*, pages 1–95. Springer, 1992.

Mauricio Alvarez, David Luengo, and Neil D Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pages 9–16. PMLR, 2009.

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.

Imanol Perez Arribas, Cristopher Salvi, and Lukasz Szpruch. Sig-sdes model for quantitative finance. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

J Backhoff-Veraguas, D Bartl, M Beiglböck, and M Eder. Adapted wasserstein distances and stability in mathematical finance. *Finance and Stochastics*, 2019.

J Backhoff-Veraguas, D Bartl, M Beiglböck, and J Wiesel. Estimating processes in adapted wasserstein distance. *Annals of Applied Probability*, 2021.

Julio Backhoff-Veraguas, Daniel Bartl, Mathias Beiglböck, and Manu Eder. All adapted topologies are equal. *Probability Theory and Related Fields*, 178(3): 1125–1172, 2020.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32:690–701, 2019.

Ronald Balvers, Yangru Wu, and Erik Gilliland. Mean reversion across national stock markets and parametric contrarian investment strategies. *The Journal of Finance*, 55(2):745–772, 2000.

Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.

Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. Deep optimal stopping. *Journal of Machine Learning Research*, 20:74, 2019.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112 (518):859–877, 2017.

Patric Bonnier, Patrick Kidger, I Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep signature transforms. *Advances in Neural Information Processing Systems*, 2019.

Patric Bonnier, Chong Liu, and Harald Oberhauser. Adapted topologies and higher rank signatures. *arXiv preprint arXiv:2005.08897*, 2020.

William L Briggs and Van Emden Henson. *The DFT: an owner's manual for the discrete Fourier transform.* SIAM, 1995.

David R Burt, Carl Edward Rasmussen, and Mark van der Wilk. Convergence of sparse variational inference in gaussian processes regression. *The Journal of Machine Learning Research*, 21:1–63, 2020a.

David R Burt, Carl Edward Rasmussen, and Mark van der Wilk. Variational orthogonal features. *arXiv preprint arXiv:2006.13170*, 2020b.

Zurab Bzhalava, Ardi Tampuu, Piotr Bała, Raul Vicente, and Joakim Dillner. Machine learning for detection of viral sequences in human metagenomic datasets. *BMC bioinformatics*, 19(1):1–11, 2018.

Thomas Cass, Bruce K Driver, Nengli Lim, and Christian Litterer. On the integration of weakly geometric rough paths. *Journal of the Mathematical Society of Japan*, 68(4):1505–1524, 2016.

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.

Jeffrey Chang. Simulating an ideal gas to verify statistical mechanics, 2015. `http://stanford.edu/~jeffjar/files/simulating-ideal-gas.pdf`.

Kuo-Tsai Chen. Integration of paths, geometric invariants and a generalized baker-hausdorff formula. *Annals of Mathematics*, pages 163–178, 1957.

Kuo-Tsai Chen. Integration of paths–a faithful representation of paths by non-commutative formal power series. *Transactions of the American Mathematical Society*, 89(2):395–407, 1958.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6572–6583, 2018.

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

Ilya Chevyrev and Andrey Kormilitzin. A primer on the signature method in machine learning. *arXiv:1603.03788*, 2016.

Ilya Chevyrev and Harald Oberhauser. Signature moments to characterize laws of stochastic processes. *arXiv preprint arXiv:1810.10971*, 2018.

Ilya Chevyrev, Terry Lyons, et al. Characteristic functions of measures on geometric rough paths. *The Annals of Probability*, 44(6):4049–4082, 2016.

Ilya Chevyrev, Andris Gerasimovics, and Hendrik Weber. Feature engineering with regularity structures. *arXiv preprint arXiv:2108.05879*, 2021.

Andreas Christmann and Ingo Steinwart. *Support Vector Machines*. Springer verlag, 2008.

Andreas Christmann and Ingo Steinwart. Universal kernels on non-standard input spaces. In *Advances in Neural Information Processing Systems*, pages 406–414, 2010.

John B Conway. *A course in functional analysis*, volume 96. Springer, 2019.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2015.

Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. *arXiv preprint arXiv:1703.01541*, 2017.

Marco Cuturi and Arnaud Doucet. Autoregressive kernels for time series. *arXiv preprint arXiv:1101.0673*, 2011.

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.

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.

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.

Laurent Decreusefond et al. Stochastic analysis of the fractional brownian motion. *Potential analysis*, 10(2):177–214, 1999.

Joscha Diehl and Jeremy Reizenstein. Invariants of multidimensional time series based on their iterated-integral signature. *Acta Applicandae Mathematicae*, 164(1):83–122, 2019.

Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *Proceedings of the 4th International Conference on Learning Representations*, pages 1–4, 2016.

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.

Tyrone E Duncan, Yaozhong Hu, and Bozenna Pasik-Duncan. Stochastic calculus for fractional brownian motion i. theory. *SIAM Journal on Control and Optimization*, 38(2):582–612, 2000.

Vincent Dutordoir, Nicolas Durrande, and James Hensman. Sparse gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*, pages 2793–2802. PMLR, 2020.

Thomas Fawcett. *Problems in stochastic analysis: Connections between rough paths and non-commutative harmonic analysis.* PhD thesis, University of Oxford, 2002.

Adeline Fermanian. Embedding and learning with signatures. *arXiv preprint arXiv:1911.13211*, 2019.

Seth Flaxman, Dino Sejdinovic, John P Cunningham, and Sarah Filippi. Bayesian learning of kernel embeddings. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 182–191, 2016.

Seth R Flaxman. *Machine learning in space and time.* PhD thesis, Carnegie Mellon University, 2015.

Peter K Friz and Martin Hairer. *A course on rough paths.* Springer, 2020.

Peter K Friz and Nicolas B Victoir. *Multidimensional stochastic processes as rough paths: theory and applications*, volume 120. Cambridge University Press, 2010.

Kenji Fukumizu, Francis R Bach, and Michael I Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan):73–99, 2004.

Kenji Fukumizu, Arthur Gretton, Xiaohai Sun, and Bernhard Schölkopf. Kernel measures of conditional dependence. In *Neural Information Processing Systems*, volume 20, pages 489–496, 2007.

Kenji Fukumizu, Le Song, and Arthur Gretton. Kernel bayes' rule: Bayesian inference with positive definite kernels. *The Journal of Machine Learning Research*, 14(1):3753–3783, 2013.

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.

Paul Gassiat et al. On the martingale property in the rough bergomi model. *Electronic Communications in Probability*, 24, 2019.

Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative Finance*, 18(6):933–949, 2018.

Chad Giusti and Darrick Lee. Signatures, lipschitz-free spaces, and paths of persistence diagrams. *arXiv preprint arXiv:2108.02727*, 2021.

Ian Goodfellow, Honglak Lee, Quoc Le, Andrew Saxe, and Andrew Ng. Measuring invariances in deep networks. *Advances in Neural Information Processing Systems*, 22, 2009.

Benjamin Graham. Sparse arrays of signatures for online character recognition. *arXiv preprint arXiv:1308.0371*, 2013.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012a.

Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems*, pages 1205–1213. Citeseer, 2012b.

Martin Hairer. An introduction to stochastic pdes. *arXiv preprint arXiv:0907.4178*, 2009.

Martin Hairer. Solving the kpz equation. *Annals of mathematics*, pages 559–664, 2013.

Martin Hairer and Étienne Pardoux. A wong-zakai theorem for stochastic pdes. *Journal of the Mathematical Society of Japan*, 67(4):1551–1604, 2015.

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.

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.

James Hensman, Nicolas Durrande, and Arno Solin. Variational fourier features for gaussian processes. *The Journal of Machine Learning Research*, 18(1): 5537–5588, 2017.

Calypso Herrera, Florian Krach, Pierre Ruyssen, and Josef Teichmann. Optimal stopping via randomized neural networks. *arXiv preprint arXiv:2104.13669*, 2021.

Terrell L Hill. *An introduction to statistical thermodynamics.* Courier Corporation, 1986.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.

Helge Holden, Bernt Øksendal, Jan Ubøe, and Tusheng Zhang. Stochastic partial differential equations. In *Stochastic Partial Differential Equations*, pages 141–191. Springer, 1996.

D. Hoover and J. Keisler. Adapted probability distributions. *Trans. Amer. Math. Soc.*, 1984.

Kelvin Hsu, Richard Nock, and Fabio Ramos. Hyperparameter learning for conditional kernel mean embeddings with rademacher complexity bounds. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 227–242. Springer, 2018.

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.

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.

Tommi Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1-2):95–114, 2000.

Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844, 2004.

Jasdeep Kalsi, Terry Lyons, and Imanol Perez Arribas. Optimal execution with rough path signatures. *SIAM Journal on Financial Mathematics*, 11(2): 470–493, 2020.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

Patrick Kidger and Terry Lyons. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. *arXiv:2001.00706*, 2020. URL `https://github.com/patrick-kidger/signatory`.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926*, 2020.

Patrick Kidger, James Foster, Xuechen Li, Harald Oberhauser, and Terry Lyons. Neural sdes as infinite-dimensional gans. *arXiv preprint arXiv:2102.03657*, 2021.

Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20, 2019.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, pages 2004–2013, 2016.

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, 2018a.

Ho Chung Leon Law, Dougal Sutherland, Dino Sejdinovic, and Seth Flaxman. Bayesian approaches to distribution regression. In *International Conference on Artificial Intelligence and Statistics*, pages 1167–1176. PMLR, 2018b.

Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. *Advances in Neural Information Processing Systems*, pages 1441–1448, 2003.

Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. *Advances in Neural Information Processing Systems*, 33, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33, 2020c.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020d.

Feng Liu, Wenkai Xu, Jie Lu, Guangquan Zhang, Arthur Gretton, and Danica J Sutherland. Learning deep kernels for non-parametric two-sample tests. In *International Conference on Machine Learning*, pages 6316–6326. PMLR, 2020.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444, 2002.

Francis A Longstaff and Eduardo S Schwartz. Valuing american options by simulation: a simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001.

Gabriel J Lord, Catherine E Powell, and Tony Shardlow. *An introduction to computational stochastic PDEs*, volume 50. Cambridge University Press, 2014.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229, 2021.

Anton Rask Lundborg, Rajen D Shah, and Jonas Peters. Conditional independence testing in hilbert spaces with applications to functional data analysis. *arXiv preprint arXiv:2101.07108*, 2021.

Terry Lyons. Rough paths, signatures and the modelling of functions on streams. *Proceedings of the International Congress of Mathematicians, Korea*, 2014.

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.

Terry et al Lyons. Coropa computational rough paths (software library). 2010. URL `http://coropa.sourceforge.net/`.

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

Terry J Lyons, Michael Caruana, and Thierry Lévy. *Differential equations driven by rough paths*. Springer, 2007.

Alexander Graeme de Garis Matthews. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

Remigijus Mikulevicius and Boris L Rozovskii. Stochastic navier–stokes equations for turbulent flows. *SIAM Journal on Mathematical Analysis*, 35(5): 1250–1310, 2004.

Jovana Mitrovic, Dino Sejdinovic, and Yee Whye Teh. Causal inference via kernel deviance measures. *arXiv preprint arXiv:1804.04622*, 2018.

PJ Moore, TJ Lyons, J Gallacher, Alzheimer's Disease Neuroimaging Initiative, et al. Using path signatures to predict a diagnosis of alzheimer's disease. *PLoS ONE*, 14(9), 2019.

James Morrill, Patrick Kidger, Lingyi Yang, and Terry Lyons. Neural controlled differential equations for online prediction tasks. *arXiv preprint arXiv:2106.11028*, 2021.

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.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint arXiv:1605.09522*, 2016.

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.

Hao Ni. *The expected signature of a stochastic process.* PhD thesis, Oxford University, UK, 2012.

Junier Oliva, Willie Neiswanger, Barnabás Póczos, Jeff Schneider, and Eric Xing. Fast distribution to real regression. In *Artificial Intelligence and Statistics*, pages 706–714. PMLR, 2014.

Sudhanshu Sekhar Panda, Daniel P Ames, and Suranjan Panigrahi. Application of vegetation indices for agricultural crop yield prediction using neural network techniques. *Remote Sensing*, 2(3):673–696, 2010.

Anastasia Papavasiliou, Christophe Ladroue, et al. Parameter estimation for rough differential equations. *The Annals of Statistics*, 39(4):2047–2073, 2011.

J Park and K Muandet. Regularised least–squares regression with infinite–dimensional output space. *arXiv preprint arXiv:2010.10973*, 2021.

Junhyung Park and Krikamol Muandet. A measure-theoretic approach to kernel conditional mean embeddings. *Advances in Neural Information Processing Systems*, 33, 2020.

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.

François Petitjean, Jordi Inglada, and Pierre Gançarski. Satellite image time series analysis under time warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8):3081–3095, 2012.

G. C. Pflug and A. Pichler. A distance for multistage stochastic optimization models. *SIAM J. Optim.*, 22(1):1–23, 2012.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 239–247, 2013.

Barnabás Póczos, Aarti Singh, Alessandro Rinaldo, and Larry Wasserman. Distribution-free distribution regression. In *Artificial Intelligence and Statistics*, pages 507–515. PMLR, 2013.

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.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20, 2007.

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.

J. Ramsay, J. Ramsay, B.W. Silverman, Springer Science+Business Media, and H.O.W.P.M.B.W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, 2005. ISBN 9780387400808. URL `https://books.google.co.uk/books?id=mU3dop5wY_4C`.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

Linda E Reichl. A modern course in statistical physics, 1999.

Jeremy Reizenstein and Benjamin Graham. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv preprint arXiv:1802.08252*, 2018.

Bastian Rieck, Tristan Yates, Christian Bock, Karsten Borgwardt, Guy Wolf, Nicholas Turk-Browne, and Smita Krishnaswamy. Uncovering the topology of time-varying fmri data using cubical persistence. *Advances in Neural Information Processing Systems*, 33:6900–6912, 2020.

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. doi: 10.1175/BAMS-85-3-381. URL `https://doi.org/10.1175/BAMS-85-3-381`.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

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

Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426. Springer, 2001.

Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

Erwin Schrödinger. *Statistical thermodynamics.* Courier Corporation, 1989.

Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, pages 2263–2291, 2013.

Rajen D Shah and Jonas Peters. The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48 (3):1514–1538, 2020.

Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. Rep the set: Neural networks for learning set representations. In *International Conference on Artificial Intelligence and Statistics*, pages 1410–1420. PMLR, 2020.

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.

Le Song, Jonathan Huang, Alex Smola, and Kenji Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968, 2009.

Le Song, Arthur Gretton, and Carlos Guestrin. Nonparametric tree graphical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 765–772. JMLR Workshop and Conference Proceedings, 2010.

Le Song, Kenji Fukumizu, and Arthur Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

Bharath K Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert RG Lanckriet. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11: 1517–1561, 2010.

Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

Xiaohai Sun, Dominik Janzing, Bernhard Schölkopf, and Kenji Fukumizu. A kernel-based causal learning algorithm. In *Proceedings of the 24th International Conference on Machine Learning*, pages 855–862, 2007.

Danica J. Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alexander J. Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *5th International Conference on Learning Representations, ICLR*, 2017.

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.

Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, 35 (3):1032–1060, 2021.

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. URL `http://jmlr.org/papers/v21/20-091.html`.

Roger Temam. *Infinite-dimensional dynamical systems in mechanics and physics*, volume 68. Springer Science & Business Media, 2012.

Robert E Tillman, Arthur Gretton, and Peter Spirtes. Nonlinear directed acyclic structure learning with weakly additive noise models. In *Neural Information Processing Systems*, pages 1847–1855, 2009.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

Csaba Toth and Harald Oberhauser. Bayesian learning from sequential data using gaussian processes with signature covariances. In *International Conference on Machine Learning*, pages 9548–9560. PMLR, 2020.

Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. *Advances in Neural Information Processing Systems*, 31, 2018.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Edward Wagstaff, Fabian B Fuchs, Martin Engelcke, Ingmar Posner, and Michael Osborne. On the limitations of representing functions on sets. *arXiv preprint arXiv:1901.09006*, 2019.

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.

Charles Walkden. Ergodic theory. *Lecture Notes University of Manchester*, 2014.

Abdul-Majid Wazwaz. Solitary waves theory. In *Partial Differential Equations and Solitary Waves Theory*, pages 479–502. Springer, 2009.

Veit Wild and George Wynne. Variational gaussian processes: A functional analysis view. *arXiv preprint arXiv:2110.12798*, 2021.

Yuesheng Xu and Haizhang Zhang. Refinable kernels. *Journal of Machine Learning Research*, 8(9), 2007.

Weixin Yang, Terry Lyons, Hao Ni, Cordelia Schmid, Lianwen Jin, and Jiawei Chang. Leveraging the path signature for skeleton-based human action recognition. *arXiv preprint arXiv:1707.03993*, 2017.

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.

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.

Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. *arXiv preprint arXiv:1202.3775*, 2012.